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

New advanced features of GPSS/H

Robert C. Crain
Daniel T. Brunner
Wolverine Software Corporation
7630 Little River Turnpike, Suite 208
Annandale, VA 22003-2653

ABSTRACT

Although most of the features that distinguish GPSS/H from other tools are widely used and have been documented in earlier papers, use of certain features remains low among some practitioners, particularly academics, who are accustomed to older and/or less functional versions of GPSS. Meanwhile, GPSS/H continues to acquire new features and capabilities. This paper documents three 1988 developments.

In support of the discussion of one of these developments, this paper also includes a discussion of techniques for modeling changeovers in a job shop.

1. INTRODUCTION

GPSS/H is a simulation language and model development system that was originally created to run models developed for IBM's widely used GPSS implementations, GPSS/360 and GPSS V, which have been available for over 15 years. Because of its speed, GPSS/H was quickly accepted by the GPSS user community.

Soon the advanced functionality of GPSS/H gave it a personality of its own. GPSS/H offered improvements in three major areas of concern to the modeler: designing and executing simulation experiments, creating enhanced user interfaces, and building and debugging models. An earlier Winter Simulation Conference paper (Crain, Brunner and Henriksen 1987) describes many of these improvements.

The three new enhancements that are presented in this paper are (1) increased model transportability, thanks in part to new hardware platforms for GPSS/H; (2) better management of in-service units of traffic through the new RESCHEDULE Block; and (3) the long-awaited incorporation of built-in probability distributions and mathematical functions.

As a means of illustrating the RESCHEDULE Block, this paper analyzes the problem of modeling a random occurrence that, whenever it occurs, has a sweeping impact on the system being studied. An example of such an event is an unscheduled changeover in a manufacturing process. A sample model is presented that addresses the simpler problem of modeling scheduled changeovers. The same model is also suitable for modeling the more difficult case of unscheduled changeovers.

2. MODEL TRANSPORTABILITY

GPSS/H has the ability to run models on a variety of hardware types. For a system as fast as GPSS/H to be able to do this is not an easy task, because GPSS/H is a compiler, and many separate code generators must be maintained. Several different operating systems must be supported, and there are two completely different underlying implementations of GPSS/H underneath it all.

Despite these difficulties, the goal of model transportability has been met. A GPSS/H model can be moved among the following environments.

Mainframes
- *IBM 370 and compatible*
  - MVS
  - CMS
  - TSO
  - VM/PC

Minicomputers
- *Digital VAX*
  - VMS
  - Ultrix

Workstations
- *Sun-3 (Unix)*
- *Apollo DN (Unix)*
- *Integrated Solutions (Unix)*
- *NCR Tower 32 (Unix)*
- *Silicon Graphics IRIS 3xxx and 4D (Unix)*

Personal Computers
- *IBM PC (MS-DOS)*

GPSS/H model transportability took a big step forward in 1988 with the introduction of two GPSS/H implementations for IBM PC and compatible microcomputers running MS-DOS. The full commercial version contains all of the functionality of the other versions except that model size is limited to moderate-sized problems by the 640K MS-DOS address space limit. Transactions, data, and compiled model code can each extend beyond one 64K segment, which at least allows for maximum flexibility in utilizing the constrained memory.

Wolverine Software has also introduced a low-priced student version of GPSS/H. Student GPSS/H, coupled with the new *Learning Simulation With GPSS/H* tutorial text (scheduled for publication in early 1989), provides a powerful, full-function environment for learning simulation and GPSS/H. The only limitations of this inexpensive software are that it will process only student-sized models and that it does not support external routines.

3. USING THE RESCHEDULE BLOCK TO MODEL CHANGEOVERS

3.1 The Need for RESCHEDULE

Any system in which: (1) discrete units of traffic (2) compete for scarce resources (3) over a period of time (4) in the presence of randomness...is a candidate for discrete event simulation.

How does a simulationist model random events? Software tools for discrete event simulation typically have data structures and built-in algorithms that are well adapted for modeling random events that affect individual units of traffic or scarce resources.

In GPSS/H, the ADVANCE Block is often used to model random intervals for service or delay times. Model logic typically translates these isolated sources of random delay into a system-wide context, making it possible to model virtually any type of small or large scale randomness. However, in cases where some other random occurrence causes sweeping, system-wide effects, the simulation model becomes more difficult to construct.

Particular difficulties arise when work in process must be interrupted and/or rescheduled. This is because a unit of traffic (a GPSS/H Transaction) that represents work in process typically resides on the Future Events Chain (FEC), where it is oblivious to outside events until its scheduled completion time is reached. Forcing such "sleeping" units of traffic to respond to unexpected events is a problem in most discrete event simulation languages.

A typical example of this problem occurs with changeovers in a manufacturing system. Changeovers, both scheduled and unscheduled, are similar to machine downtime. Most simulation tools have good capabilities for modeling isolated downtimes. GPSS/H handles downtime easily via the FUNAVAIL and FAVAIL Blocks, which toggle the status of a single server from "one unit available" to and from "zero units available." Most GPSS/H models use this scheme for modeling downtime. It's effective because execution of the FUNAVAIL Block can actually affect the status of Transactions on the Future Events Chain.

In contrast to simple, isolated downtimes, changeovers can have much more sweeping effects on the system, and are thus more complicated to model.

Through careful management of User Chains, a GPSS/H simulationist can model virtually any type of random changeovers. However, this typically takes the model away from the simple SEIZE-ADVANCE-RELEASE paradigm. The problem is that at the time of the changeover, the units of work in progress are resting, oblivious, on the Future Events Chain, waiting for their predetermined ADVANCE Block time to expire. With random changeovers, FUNAVAIL and FAVAIL are often less convenient constructs. Unfortunately, before 1988, FUNAVAIL and PREEMPT (used for wresting control of a Facility away from its current user) were the only Blocks that could affect a Transaction on the Future Events Chain. (SUNAVAIL cannot affect FEC Transactions because no record of "ownership" is kept for Transactions that ENTER or LEAVE a Storage.)

Now there is a new GPSS/H Block: RESCHEDULE. The explicit purpose of the RESCHEDULE Block is to allow direct manipulation of selected Transactions that are resting on the Future Events Chain. With the implementation of RESCHEDULE came an immediate need to uniquely identify Transactions on the Future Events Chain so that they could be manipulated, so the XID1 (unique Transaction identifier) Standard Numerical Attribute came into the language at the same time as RESCHEDULE.

3.2 The Changeover Modeling Example

3.2.1. Problem Statement. In the sample system, a job shop operates around the clock with scheduled shift-boundary changeovers. At three shift boundaries each day, the capacities of each of three stations change due to changes in the number of operators. When the capacities change in a downward direction, work in progress must be interrupted and possibly rescheduled.

The capacities of the stations are as follows:

	Station 1	Station 2	Station 3
Shift 1	3	2	1
Shift 2	2	2	1
Shift 3	1	0	1

Note that the stations can have capacity greater than one. This precludes straightforward use of the GPSS/H FUNAVAIL Block unless the natural Storage construct for multiple servers is replaced by an array of Facilities.

The sample GPSS/H model presented here shows *scheduled* changeovers taking place. However, this model is flexible enough to handle different changeover schedules or even random, *unscheduled* changeovers.

3.2.2. The Look-Ahead Approach. Among the several approaches available in GPSS/H for modeling scheduled changeovers, the most simple-minded approach would be to "look ahead" before placing a particular Transaction on the Future Events Chain, comparing its Move Time with the time of the next scheduled changeover. In this way, the Transaction can be scheduled to come off the FEC in anticipation of the changeover. For scheduled changeovers, this would solve the sample problem quite nicely. A non-GPSS/H illustration of the Look-Ahead approach to solving the sample problem appears in a software vendor newsletter (Systems Modeling Corporation 1987).

The Look-Ahead approach becomes complex in the case of randomly occurring (unscheduled) changeovers. If the randomness of the changeover times results from complicated combinations of circumstances elsewhere in the model – not an unlikely scenario – then the Look-Ahead method fails, because there is no known time to which to look ahead.

3.2.3. The Look-Out Approach. A more sophisticated, albeit slightly more complex, approach would be a "clone" method. In this method, one "watchful" Transaction is used to monitor each "oblivious" Future Events Chain Transaction. This "Look-Out" Approach could be implemented without a RESCHEDULE Block as follows.

Each Transaction about to be placed on the FEC SPLITs off a "clone" that does the actual waiting on the FEC. The original Transaction then waits on a User Chain, where it can more easily be forced to respond to an unpredicted event. If no changeover occurs, then the clone Transaction eventually comes off the FEC, UNLINKs the original Transaction from the User Chain for further processing, and destroys itself. If a random changeover caused the original Transaction to be UNLINKed before its originally scheduled time, then the original Transaction can proceed through the model earlier

than it was originally scheduled to do so. The clone Transaction will eventually exit the FEC, attempt to UNLINK its clone (this will have no effect), and quietly destroy itself.

The "Look-Out" approach is effective, even in the case of an unscheduled changeover, but sometimes the model gets too complex for all this artful handshaking to be comprehensible. That's where the RESCHEDULE Block comes in.

RESCHEDULE allows GPSS/H to remove immediately the original Transaction from the FEC, or to recalculate its Move Time while leaving it on the FEC.

RESCHEDULE does have some limitations. In particular, RESCHEDULE can affect only one FEC Transaction at a time. This forces the simulationist to maintain a list of Transactions that would need to be RESCHEDULEd in the event of a changeover. The underlying problem is that in an efficiently executing model, a Transaction cannot wait and watch at the same time.

A simple way to maintain such a list is to resort again to a clone method, where this time the clones are waiting on the User Chain and the original Transactions are on the FEC. Although this may seem like the non-RESCHEDULE method with a twist, RESCHEDULE is somewhat simpler to use than the method described above, because the original Transaction waits where expected – on the FEC. It seems likely that GPSS/H users will discover cases where RESCHEDULE brings dramatic simplification, especially in complex models.

3.2.4. The GPSS/H Sample Model. A GPSS/H model of the sample problem is presented in Figure 1. This model uses the Look-Out approach for handling the shift transitions and the RESCHEDULE Block for notifying Future Events Chain Transactions of each shift change. Standard statistical output for this model appears in Figure 2. For the simple system presented, a GPSS/H Look-Out model that does not use RESCHEDULE is also possible, and would require about the same number of statements as the model in Figure 1.

The problem statement for the scheduled changeover example was adapted from an article that appeared in the newsletter of another simulation vendor (Systems Modeling Corporation 1987). The same article contains a solution for this scheduled changeover problem that uses the less flexible Look-Ahead method, which is not easily adaptable to an unscheduled changeover problem. That model, coded in the SIMAN language with FORTRAN User Code, contains 20 lines of Model code, 27 lines of Experiment code, and 53 lines of FORTRAN code, or a total of 100 lines of code.

The GPSS/H model presented in Figure 1, which uses the more flexible Look-Out method, contains 28 Block Statements and 11 lines of Compiler Directives and Control Statements, and does not require the use of an external programming language. There are 39 total lines of code. This is a concrete example of the superior modeling flexibility that is often claimed for GPSS/H.

4. NEW MATH AND RANDOM VARIATE SNAs

Prior to 1988, doing complicated mathematics and calling mathematically defined random variate (probability distribution) generators in GPSS/H required the use of external routines.

People often ask whether GPSS/H supports calling external routines during a simulation run The answer is yes. On IBM mainframes, the language must be FORTRAN or Assembler; under VAX/VMS, the language can be any that follows VMS calling conventions (most do); under Unix, the language must be C (but other languages can often be glued in via a dummy C routine); and under MS-DOS, at least C and FORTRAN will be supported, although external routines were not available under MS-DOS GPSS/H as of this writing.

Despite this extensive support, Wolverine Software Corporation has recently stated that calling external (e.g. FORTRAN) routines for such mundane things as I/O, math, statistical routines, and complicated model logic should be completely abandoned. Simulation languages are complicated enough without requiring the simulationist to deal with another compiler and another set of syntax.

This way of thinking is evident in the I/O capabilities that have been part of GPSS/H almost from its inception. Historically, GPSS simulationists often used FORTRAN solely for writing custom output, for file I/O, or for custom user interfaces. Some users still do this, although GPSS/H eliminated that need years ago.

GPSS has also always had the flexibility to deal with arbitrarily complex logic. The only thing that prevented earlier adoption of the "avoid-FORTRAN" position was the long-delayed introduction of built-in random variate generators and sophisticated math functions into the GPSS/H language.

In 1988 these capabilities became part of the language:

Math	Distributions
ACOS	RVEXPO (Exponential)
ASIN	RVNORM (Normal)
ATAN	RVTRI (Triangular)
COS	
EXP	
LOG	
SIN	
SQRT	
TAN	

Other closed-form distributions can be built using the ones provided and coded directly into a model.

Although these new features are implemented and classified as Standard Numerical Attributes, each one behaves syntactically like a programming language function that returns a value. The sample model from the previous section shows one of these features in use.

GPSS/H already has a partial built-in language for addressing programming needs through its Control Statement Programming Language (Crain, Brunner and Henriksen, 1987); see also (Henriksen and Crain 1983). The new math and probability distribution SNAs add strength to this language. As this built-in language is expanded and refined, GPSS/H simulationists should find that the need to call routines written in FORTRAN or another external language continues to diminish.

5. SUMMARY

GPSS/H has emerged as a leading tool for people who regularly model complex systems. This acceptance comes despite

```
***
***     GPSS/H MODEL OF A SIMPLE JOB SHOP WITH THREE STATIONS
***
***     SCHEDULED SHIFT CHANGES TAKE PLACE EVERY EIGHT HOURS
***     THIS MODEL LOGIC WOULD ALSO WORK IF THE CHANGEOVERS
***     OCCURRED AT UNPREDICTABLE TIMES
***
***     THIS MODEL ILLUSTRATES SEVERAL ADVANCED FEATURES OF GPSS/H,
***     INCLUDING THE NEW RESCHEDULE BLOCK AND BUILT-IN PROBABILITY
***     DISTRIBUTIONS
***
        SIMULATE
*
        INTEGER         &SHIFT                          DATA DECLARATION (INTEGER AMPERVARIABLE)
CAPS    MATRIX          MH,3,3                          DATA DECLARATION (HALFWORD MATRIX SAVEVALUE)
*
PMEAN   FUNCTION        PH(STATION),M3                  FUNCTION TO RETURN MEAN PROCESSING TIME, WHICH
1,0.3/2,0.6/3,0.9                                         DEPENDS ON WHICH STATION THE UNIT IS VISITING
*
******* MODEL BLOCKS FOR UNITS OF WORK PASSING THROUGH THE THREE STATIONS
*
        GENERATE        RVEXPO(1,1),,,,,4PH,2PL         EXPONENTIAL ARRIVALS, RANDOM STREAM 1, MEAN IAT = 1
JOBSIN  ASSIGN          STATION,1,PH                    FIRST STOP IS STATION 1
*
NEXTSTN ASSIGN          PROCTIME,RVEXPO(2,FN(PMEAN)),PL EXPONENTIAL SERVICE TIMES, MEAN IS FUNCTION OF STATION
INQ     QUEUE           PH(STATION)                     RECORDING ENTRANCE TO STATION FOR STATISTICS
STN     ENTER           PH(STATION)                     ATTEMPT TO GRAB ONE MACHINE AT CURRENT STATION
        ASSIGN          DONETIME,PL(PROCTIME)+AC1,PL    CALCULATE AND KEEP TRACK OF COMPLETION TIME
        ASSIGN          MYBRO,XID1,PH                   STAMP SELF WITH UNIQUE NAME, WHICH CLONE WILL ALSO GET
        SPLIT           1,CLONE                         SEND CLONE OFF TO WATCH FOR SHIFT CHANGE
        ADVANCE         PL(PROCTIME)                    WAIT ON FUTURE EVENTS CHAIN
        LEAVE           PH(STATION)                     RELINQUISH MACHINE
OUTQ    DEPART          PH(STATION)                     RECORD EXIT FROM STATION FOR STATISTICS
*
        ASSIGN          STATION,PH(STATION)+1,PH        INCREMENT CURRENT STATION NUMBER
        TEST G          PH(STATION),3,NEXTSTN           TRANSFER TO NEXT STATION UNLESS PAST STATION 3
*
JOBSOUT TERMINATE                                       EXIT THE MODEL
*
******* MODEL BLOCKS WHERE UNITS OF W-I-P WAIT (ZERO TIME) FOR SHIFT CHANGE TO COMPLETE
*
SUSPEND LEAVE           PH(STATION)                     ALL XACTS DRAINED FROM STORAGE COME HERE
        GATE LR         SCHANGE                         WAIT FOR CAPACITIES TO BE READJUSTED
        ASSIGN          PROCTIME,PL(DONETIME)-AC1,PL    REFIGURE SERVICE TIME
        TRANSFER        ,STN                            TRY TO GO BACK IN SERVICE
*
******* MODEL BLOCKS WHERE CLONE TRANSACTIONS WAIT FOR SHIFT CHANGE TO OCCUR
*
CLONE   GATE LS         SCHANGE                         CLONE DOES NOTHING UNTIL A SHIFT CHANGE OCCURS
        RESCHEDULE      PH(MYBRO),AC1,SUSPEND,OK        SIBLING, IF STILL ALIVE, GOES NOW TO BLOCK NAMED SUSPEND
OK      TERMINATE                                       CLONE DIES PEACEFULLY
*
******* MODEL BLOCKS FOR SCHEDULING SHIFT CHANGES
*
        GENERATE        8,,,,-1                         IN THIS EXAMPLE, SHIFT CHANGEOVERS OCCUR EVERY 8 HOURS
NEWDAY  LOGIC S         SCHANGE                         SET 'SHIFT CHANGE IN PROGRESS' FLAG
        BUFFER                                          PAUSE TO ALLOW W-I-P TO HALT
        BLET            &SHIFT=(N(NEWDAY))@3+1          FIGURE OUT IF THIS IS SHIFT 1, 2, OR 3
        BSTORAGE        S1,MH(CAPS,1,&SHIFT)/S2,MH(CAPS,2,&SHIFT)/S3,MH(CAPS,3,&SHIFT)         MODIFY CAPACITIES
        LOGIC R         SCHANGE                         CLEAR 'SHIFT CHANGE IN PROGRESS' FLAG
        TERMINATE       1                               TELL EXPERIMENT CONTROL THAT SHIFT IS DONE
*
******* DATA INITIALIZATION AND EXPERIMENT CONTROL
*
        INITIAL         MH$CAPS(1,1),3/MH$CAPS(1,2),2/MH$CAPS(1,3),1
        INITIAL         MH$CAPS(2,1),2/MH$CAPS(2,2),2/MH$CAPS(2,3),0
        INITIAL         MH$CAPS(3,1),1/MH$CAPS(3,2),1/MH$CAPS(3,3),1
*
        STORAGE  S1,MH(CAPS,1,1)/S2,MH(CAPS,2,1)/S3,MH(CAPS,3,1)        SET INITIAL CAPACITIES
*
        START           20*3                                            RUN 20 DAYS (60 SHIFTS)
        END
```

Figure 1. The GPSS/H Model

RELATIVE CLOCK: 480.0000 ABSOLUTE CLOCK: 480.0000

BLOCK CURRENT	TOTAL	BLOCK CURRENT	TOTAL	BLOCK CURRENT	TOTAL
1	483	OUTQ	1412	OK	1986
JOBSIN	483	12	1412	22	60
NEXTSTN	1439	13	1412	NEWDAY	60
INQ 8	1439	JOBSOUT	456	24	60
STN	1986	SUSPEND 19	574	25	60
6	1986	16	555	26	60
7	1986	17	555	27	60
8	3972	18	555	28	60
9	1986	CLONE	1986		
10	1412	20	1986		

	--AVG-UTIL-DURING--								
STORAGE	TOTAL TIME	AVAIL TIME	UNAVL TIME	ENTRIES	AVERAGE TIME/UNIT	CURRENT STATUS	PERCENT AVAIL	CAPACITY	AVERAGE CONTENTS
1	0.102			503	0.293	AVAIL	100.0	3	0.307
2	0.314			502	0.601	AVAIL	100.0	2	0.629
3	0.909			981	0.445	AVAIL	100.0	1	0.909

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/UNIT	$AVERAGE TIME/UNIT
1	8	0.385	483	0		0.382	0.382
2	14	2.922	482	0		2.910	2.910
3	28	9.098	474	0		9.213	9.213

HALFWORD MATRIX SAVEVALUE CAPS

ROW/COL	1	2	3
1	3	2	1
2	2	2	0
3	1	1	1

RANDOM STREAM	ANTITHETIC VARIATES	INITIAL POSITION	CURRENT POSITION	SAMPLE COUNT	CHI-SQUARE UNIFORMITY
1	OFF	100000	100484	484	0.90
2	OFF	200000	201439	1439	0.22

Figure 2. GPSS/H Model Output

the fact that many of the advanced features of GPSS/H are not well known or (in some cases) well understood. Papers such as this one provide GPSS/H users as well as other observers with the opportunity to keep up with the latest developments in the language.

REFERENCES

Crain, R. C., Brunner, D. T., and Henriksen, J. O. "Advanced Features of GPSS/H," *Proceedings of the 1987 Winter Simulation Conference*, 269-275.

Henriksen, J. O., "Alternatives for Modeling of Preemptive Scheduling," *Proceedings of the 1987 Winter Simulation Conference*, 575-581.

Henriksen, J. O., and Crain, R. C, *GPSS/H User's Manual*, Second Edition. Wolverine Software Corporation, Annandale, Virginia, 1983.

Pegden, C. D., *Introduction to SIMAN*. Systems Modeling Corporation, 1982.

"The Sample Example," *SIMAN SEZ..*, Fall 1987, page 6 (author unknown). Systems Modeling Corporation, State College, Pennsylvania.

AUTHORS' BIOGRAPHIES

ROBERT C. CRAIN has been with Wolverine Software 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. Mr. Crain is a member of SCS, SIGSIM, and ACM, and served as Business Chairman of the 1986 Winter Simulation Conference.

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 Software since 1986. Mr. Brunner is a member of IIE and SCS, and served as Publicity Chair for the 1988 Winter Simulation Conference.