1981 Winter Simulation Conference Proceedings
T.I. Oren, C.M. Delfosse, C.M. Shub (Eds.)

GPSS - FINDING THE APPROPRIATE WORLD-VIEW

James O. Henriksen
Wolverine Software Corporation
P.O. Box 1251
Falls Church, VA 22041

Every simulation language embodies a world-view which heavily influences approaches taken in building models in the language. In most applications for which a given language is used, the world-view of the language enforces a discipline of programming which results in models which are time- and space-efficient, reflecting the usefulness of the language and the appropriateness of language choice by the programmer. For some applications, however, the programming style encouraged by the world-view of a language can lead to programs which are time- and space-inefficient, even though the programs are natural, straightforward solutions to the problem at hand. In such cases, one may be forced to consider alternative languages or to alter one's approach in application of a given language. This paper briefly summarizes the world-view of the GPSS language and gives two examples of systems which, when modelled with conventional GPSS approaches, result in inefficient programs. For each system, two GPSS models are presented: a straightforward model which is inefficient, and a clever model which is efficient. In both cases, the clever models are easily programmed in GPSS and require only marginally more skill on the part of the programmer than do the straightforward models. Once an appropriate alternative to the obvious GPSS world-view is found, the rest is easy. A working knowledge of GPSS is required to read this paper.

1. THE GPSS WORLD-VIEW

The world-view of GPSS (Schriber 1974) is that of Transaction flow; i.e., that of motion of dynamic elements (Transactions) through a flowchart-inspired program specifying the rules of operation of the system. In GPSS, the resources for which Transactions compete are usually modelled as Facilities (single server entities) or Storages (multiple servers). From a programming viewpoint, GPSS resources are passive entities: their behavior patterns are the result of handling requests made by active model elements (Transactions) in accordance with the predefined, built-in rules of operation of the GPSS simulator program. Other languages, such as Simula (Franta 1977), offer world-views in which resources are programmed as active entities.

The Transaction-flow world-view of GPSS is applicable to a wide range of systems. The examples given in this paper are from manufacturing systems. For such systems, the Transaction-flow world-view of GPSS often results in easily written, straightforward, highly readable models. For example, an assembly line may be modelled by representing the parts flowing through the system as Transactions, and the resources for which the parts contend may be represented as GPSS Facilities or Storages.

2. A FIRST HYPOTHETICAL EXAMPLE

A hypothetical manufacturing system to be considered is shown in Figure 1. The systems operates as follows:

1. The first machine in the system is preceded by an infinite supply of unfinished parts; i.e., whenever the first machine is ready to machine another part, the unfinished part is assumed to be instantaneously available.
2. The third machine in the system is followed by an infinite output bin; i.e., each time the

third machine finishes a part, the part is instantaneously removed from the machine and exits the system.

3. The first and second machines are connected by a bin of fixed capacity which serves as an output bin for the first machine and an input bin for the second machine. The second and third machines share a similar bin.

4. The three machines operate continuously (without breakdowns), subject only to two kinds of blockage: input bin empty or output bin full. Note that the first machine will never experience an empty input bin, and the third machine will never experience a full output bin.

5. Machining times for this example are really irrelevent. They have been chosen as 100 +- 90 seconds, uniformly distributed. This distribution contains enough variance to make the results moderately interesting.

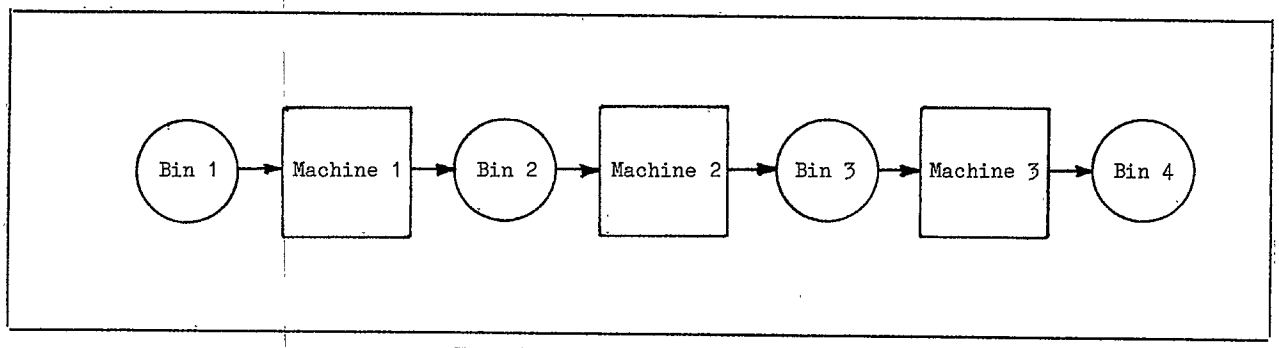

Fig. 1 - First Hypothetical System

2.1 Naive Approach to the First Hypothetical Example

A naively coded model of the first hypothetical system is shown in Figure 2.

Structure of the Model - The naive approach to modelling our hypothetical system is a classic GPSS passive server approach. Transactions are used to represent parts flowing through the system. Bins 1, 2, and 3 are represented as Storage entities (multiple servers), and machines 1, 2, and 3 are represented as User Chains (single servers). (If an extremely naive approach were to be taken, machines would be represented as GPSS Facilities, and the results would be quite disastrous. The use of User Chains enables the best possible implementation of our (admittedly poor) approach. A tutorial on the use of User Chains is well beyond the scope of this paper. Interested readers should see references (Schriber 1974) and (Ingerman 1981).) The operation of the model is as follows:

1. An infinite supply of unfinished parts is simulated by the first GENERATE Block in the model. Parts are GENERATEd at an infinite rate until Storage BIN1 (the input bin for the first machine) fills up. After BIN1 has initially been filled, whenever the first machine removes a part from BIN1, storage BIN1 becomes "not full," and another part Transaction is allowed to escape from the first GENERATE Block.

2. Machines 1, 2, and 3 are modelled as User Chains MACH1, MACH2, and MACH3, respectively. The conditional form of the LINK Block is used, so that only one part Transaction is allowed to control a machine at any given time.

3. The most critical aspect of model implementation is the ENTER-UNLINK Block sequence at the conclusion of the first and second machine operations. This sequence guarantees that a part must be placed in the machine's output bin before the part Transaction can proceed to the UNLINK Block, allowing a successor part Transaction to have access to the current machine.

4. The modelling of input starvation is implicit. Since a Transaction is used to represent a part, and a User Chain is used to represent a machine, input starvation at a machine corresponds to a situation where no part Transactions are currently attempting to pass through a LINK Block for the User Chain, nor are any Transactions currently on the User Chain as a result of a previously "unsuccessful" execution of the LINK Block.

Comments on this Approach - The approach outlined above is very straightforward, but as we will see below, it results in a very inefficent model. Some reflection on our approach will make the reasons clear. Since Transactions have been used to represent parts, the number of Transactions active in the model at any given point is (approximately, for purists) equal to the number of parts currently in the system. In this example, BIN1, BIN2, and BIN3 have capacities of 100 parts. Thus, in the worst case, 300 Transactions could be simultaneously active in the model. If the capacities of these bins were altered to 1000, in the worst case, the model could contain 3000 active Transactions. The pattern is apparent: as the size and traffic of the system increase, so do the size and complexity of the computer run-time representation of the system. The approach which is given below alleviates this problem.

```
GPSS/H  VP/CSS RELEASE 1.0 (UN261)       29 JUL 81   8:30:56    FILE: WIDGETUC

LINE# STMT#  IF DO  BLOCK#  *LOC   OPERATION  A,B,C,D,E,F,G   COMMENTS

00010     1                       SIMULATE
00020     2                       REALLOCATE COM,20000       SUFFICIENT COMMON

00040     4                  ********************************************************************************
00050     5                  *                                                                              *
00060     6                  *      WIDGET PRODUCTION LINE MODEL                                            *
00070     7                  *      (PASSIVE SERVER, ACTIVE WIDGET APPROACH)                                *
00080     8                  *                                                                              *
00090     9                  ********************************************************************************

00110    11                       STORAGE    S$BIN1,100/S$BIN2,100/S$BIN3,100 INTERMEDIATE BINS

00130    13                  PROC1  FUNCTION   RN1,C2          MACHINE 1 PROCESSING TIME
00140    14                  0,10/1,191
00150    15                  PROC2  FUNCTION   RN2,C2          MACHINE 2 PROCESSING TIME
00160    16                  0,10/1,191
00170    17                  PROC3  FUNCTION   RN3,C2          MACHINE 3 PROCESSING TIME
00180    18                  0,10/1,191

00200    20             1         GENERATE                     SIMULATE AN INFINITE SUPPLY
00210    21             2         ENTER      BIN1              ONLY CONSTRAINT: BIN 1 CAPACITY
00220    22             3         LINK       MACH1,FIFO,GOT1   GRAB FIRST MACHINE
00230    23             4   GOT1  LEAVE      BIN1              LEAVE FIRST BIN WHEN FIRST MACH FREE
00240    24             5         ADVANCE    FN$PROC1          MACHINE 1 PROCESSING TIME
00250    25             6         ENTER      BIN2              DEPOSIT PART IN OUTPUT BIN
00260    26             7         UNLINK     MACH1,GOT1,1      ALLOW NEXT PART TO HAVE MACH 1
00270    27             8         LINK       MACH2,FIFO,GOT2   GRAB SECOND MACHINE
00280    28             9   GOT2  LEAVE      BIN2              REMOVE PART FROM INPUT BIN
00290    29            10         ADVANCE    FN$PROC2          MACHINE 2 PROCESSING TIME
00300    30            11         ENTER      BIN3              DEPOSIT PART IN OUTPUT BIN
00310    31            12         UNLINK     MACH2,GOT2,1      ALLOW NEXT PART TO HAVE MACH 2
00320    32            13         LINK       MACH3,FIFO,GOT3   GET THIRD MACHINE
00330    33            14   GOT3  LEAVE      BIN3              REMOVE PART FROM INPUT BIN
00340    34            15         ADVANCE    FN$PROC3          MACHINE 3 PROCESSING TIME
00350    35            16         UNLINK     MACH3,GOT3,1      ALLOW NEXT PART TO HAVE MACH 3
00360    36            17         TERMINATE  1                 PART COMPLETED

00380    38                       RMULT      11111,33333,55555 MAKE RN1, RN2, RN3 INDEPENDENT
00390    39                       START      5000              SIMULATE PRODUCTION OF 5000 PARTS
00400    40                       END
```

Fig. 2 - Naive Model of First Hypothetical System

2.2 Sophisticated Approach to the First Hypothetical Example

A sophisticated model of the first hypothetical system is shown in Figure 3.

Structure of the Model - The sophisticated approach to modelling our first hypothetical system uses an "active server" approach. While this approach is an obvious approach in languages such as Simula, its use in GPSS requires a bit of extra thought. In GPSS, simultaneous operations are almost always modelled by simultaneously active Transactions; i.e., Transactions embody the capability for representing parallelism in a system. In our hypothetical system, there are at most three machining operations going on at any given time. Accordingly, it is proper to consider whether three Transactions can be used to model the operation of the system, one for each machine.

The model shown in Figure 3 uses this approach. It operates as follows:

1. Machine 1 is represented by a single Transaction which traverses an infinite loop. In the loop, machining time is modelled by an appropriate ADVANCE Block. The output bin for machine 1 is modelled by a Storage named BIN2. Within its infinite loop, the Transaction representing machine 1 is delayed only when BIN2 is full. There is no delay for input starvation, since machine 1 is assumed to have an infinite supply of unfinished parts.
2. Machine 2 is also represented by a single Transaction which traverses an infinite loop. The loop is similar to the loop for machine 1, except that in addition to modelling output congestion,

input starvation must be accounted for. This is handled by including a GATE SNE BIN2 Block. Inclusion of this Block forces the machine 2 Transaction to wait until its input bin becomes non-empty.

3. Machine 3 is represented by a single Transaction which traverses an infinite loop. This loop is similar to the loop for machine 2, with two exceptions: first, there is no need to provide for output congestion (by definition of the hypothetical system), and second, logic has been included to terminate model execution after 5000 parts have been machined.

Comments on this Approach - The approach outlined above is readily implemented in GPSS, using standard language constructs; however this approach is almost certainly not the first approach that would occur to a beginning GPSS modeller. Since the model contains only three simultaneously active Transactions, one can readily anticipate comparative results vis-a-vis the naive approach.

```
GPSS/H  VP/CSS RELEASE 1.0 (UN261)       29 JUL 81   8:32:35    FILE: WIDGETAS

LINE# STMT#  IF DO  BLOCK#  *LOC   OPERATION  A,B,C,D,E,F,G   COMMENTS

00010     1                       SIMULATE
00020     2                       REALLOCATE COM,15000       SUFFICIENT COMMON

00040     4                  ********************************************************************************
00050     5                  *                                                                              *
00060     6                  *      WIDGET PRODUCTION LINE MODEL                                            *
00070     7                  *      (ACTIVE SERVER, PASSIVE WIDGET APPROACH)                                *
00080     8                  *                                                                              *
00090     9                  ********************************************************************************

00110    11                  PROC1  FUNCTION   RN1,C2          MACHINE 1 PROCESSING TIME
00120    12                  0,10/1,191
00130    13                  PROC2  FUNCTION   RN2,C2          MACHINE 2 PROCESSING TIME
00140    14                  0,10/1,191
00150    15                  PROC3  FUNCTION   RN3,C2          MACHINE 3 PROCESSING TIME
00160    16                  0,10/1,191

00180    18             1        GENERATE   ,,,1             MACHINE 1 XACT
00190    19             2  MLUP1 ADVANCE    FN$PROC1         MACHINE 1 PROCESSING TIME
00200    20             3        ENTER      BIN2             PLACE PART IN OUTPUT BIN
00210    21             4        TRANSFER   ,MLUP1           PROCESS PARTS FOREVER

00230    23             5        GENERATE   ,,,1             MACHINE 2 XACT
00240    24             6  MLUP2 GATE SNE   BIN2             WAIT FOR PART IN INPUT BIN
00250    25             7        LEAVE      BIN2             REMOVE PART FROM BIN
00260    26             8        ADVANCE    FN$PROC2         MACHINE 2 PROCESSING TIME
00270    27             9        ENTER      BIN3             PLACE PART IN OUTPUT BIN
00280    28            10        TRANSFER   ,MLUP2           PROCESS PARTS FOREVER

00300    30            11        GENERATE   ,,,1             MACHINE 3 XACT
00310    31            12  MLUP3 GATE SNE   BIN3             WAIT FOR PART IN INPUT BIN
00320    32            13        LEAVE      BIN3             REMOVE PART FROM INPUT BIN
00330    33            14  LASTM ADVANCE    FN$PROC3         MACHINE 3 PROCESSING TIME
00340    34            15        TEST E     N$LASTM,5000,MLUP3  PROCESS 5000 PARTS
00350    35            16        TERMINATE  1                AND STOP.

00370    37                       RMULT      11111,33333,55555 MAKE RN1, RN2, RN3 INDEPENDENT
00380    38                       START      1                SIMULATE PRODUCTION OF 5000 PARTS
00390    39                       END
```

Fig. 3 - Sophisticated Model of First Hypothetical System

2.3 First Hypothetical Example - Comparison of Results

Due to space limitations, results of running the two models cannot be included herein; rather, comparative results will be briefly summarized. Although differences in approach dictated that the two models collected different statistics, it was immediately apparent from common statistics that the two models were functionally identical. The time- and space-efficiencies of the two models, however, differed dramatically. The naive model took twice as much execution time and 28 times as much COMMON storage as the sophisticated approach. (COMMON storage is used for Transactions, among other things.) The cost savings achieved in this example are fairly modest, but this is an extremely simple example. In larger, more complex systems, even greater cost savings could be achieved. In addition, the

sophisticated approach offers a highly desirable form of modularity and readability, by localizing the logic for operation of a machine. For all machines in the model, the rules by which the machine operates are contained in the infinite loop for the machine. This is in contrast to the naive approach, in which the same rules of operation are imbedded in the flowchart-inspired description of overall part flow. In larger, more complex models, with more complex contention for resources, the rules of operation for resources can easily become dispersed throughout the model, making the model more difficult to read and debug.

3. A SECOND HYPOTHETICAL EXAMPLE

A second hypothetical example, a sparkplug packaging line, is depicted in Figure 4. The system operates as follows:

1. At time zero, a stream of sparkplugs begins flowing into the packing area on a conveyor belt at a rate of 1000 plugs per minute. The stream of sparkplugs is assumed to be continuous. It takes 0.3 minutes for the initial flow of sparkplugs to reach the first of several packing machines.
2. Each packing machine operates as follows:
   A. The machine is initially idle.
   B. When spark plugs reach the position of the packing machine along the conveyor, the packing machine begins packing plugs at a rate of up to 333 plugs per minute.
   C. If the rate of flow on the conveyor at any given machine exceeds 333 plugs per minute, the excess flows downstream to the next machine.
   D. Machines are susceptible to failure. For every 400 +- 200 (uniformly distributed) plugs a machine packs, a jam requiring operator intervention occurs. The operator requires 15 +- 9 seconds to unjam the machine. When a machine fails, the flow of plugs that the machine was packing begins to flow past the machine; i.e., a downstream surge is created. When a machine has been repaired, it continues from step B, above; i.e., if at the time it is becomes available, a non-zero flow exists on the conveyor at the point of the machine, the machine will resume packing plugs at a rate up to its maximum. The resumption of a machine will cause a decrease in flow downstream, i.e., a "negative surge."
3. The time for a spark plug to travel from one machine to the next is 0.15 minutes.
4. The system is to be configured with a fixed number of packing machines. If enough of the packing machines fail with sufficient simultaneity, spark plugs flow past the last machine, off the end of the conveyor, into a barrel. Plugs which flow into the barrel must subsequently be manually reloaded onto the conveyor, upstream from the first packing machine. (Reintoduction of dumped plugs is ignored here.) The purpose of the model is to enable management to make a cost/benefit analysis of the number of packing machines to be installed in the packing subsystem.

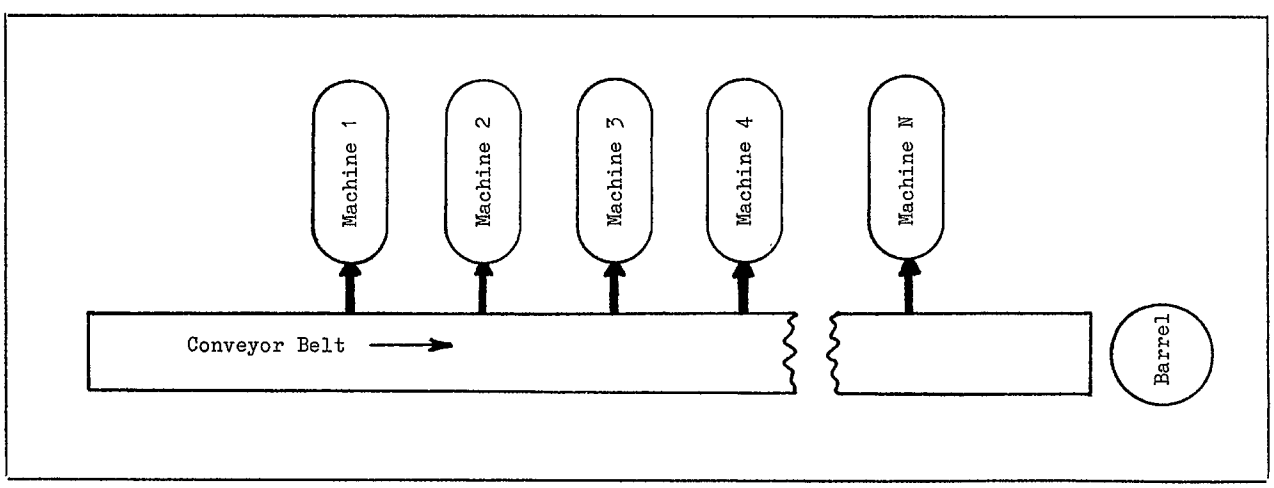

Fig. 4 - Second Hypothetical Example

3.1 Naive Approach to the Second Hypothetical Example

A naively coded model of the second hypothetical system is shown in Figure 5.

Structure of the Model - The naive approach to modelling the sparkplug packing line is (again) a classic GPSS approach, representing sparkplugs as Transactions and machines as Facilities (with associated counters). The operation of the model is as follows:

1. At time zero, random samples are drawn to determine the number of plugs until the first failure of each packing machine in the system. The Transaction which performs this initialization also serves as a timer Transaction, shutting off the model after 10 minutes of simulated operation.

2. The time unit of the simulation is .001 minute. Since the initial surge of sparkplugs takes 0.3 minutes to reach the first machine, a GENERATE 1,,300 Block accomplishes the purpose of introducing a flow of 1000 plugs per minute at the first machine, beginning at time 0.3 minutes.
3. A sparkplug flows through the model by looking at each machine in succession until it encounters a packing machine which can pack the plug or until it flows off the end of the conveyor, into the barrel.
4. When a plug finds a machine that can pack it, it SEIZEs the Facility representing the machine, ADVANCES for .003 minutes, and executes the logic corresponding to machine failure.
5. Machine failures are assumed to occur at the conclusion of packing of a sparkplug. Each time a plug is packed, a failure counter is decremented. When the counter goes to zero, a failure is simulated by making the machine Facility unavailable until it has been repaired. When the machine has been repaired, the Facility is once again made available, and a new random sample is drawn to determine the number of plugs that will be processed until the next failure occurs.

Comments on this Approach - The approach outlined above is very straightforward, but leads to a very inefficient model. The reader who has carefully read through the first hypothetical example should cringe at the mere mention of phrases like "1000 plugs per minute." The numbers chosen for this example are reasonably realistic; however, it should be obvious that were the size and traffic level of this system to be increased, correspondingly large numbers of Transactions would be required for model execution.

```
GPSS/H  VP/CSS RELEASE 1.0 (UN261)       26 JUL 81  15:39:18    FILE: SPARKEZ

LINE# STMT#  IF DO  BLOCK#  *LOC   OPERATION  A,B,C,D,E,F,G   COMMENTS

00010     1                       SIMULATE
00020     2                       REALLOCATE COM,50000        LOTS OF COMMON

00040     4                 ********************************************************************************
00050     5                 *                                                                              *
00060     6                 *      SPARKPLUG PACKING LINE MODEL                                            *
00070     7                 *      NAIVE GPSS APPROACH                                                     *
00080     8                 *                                                                              *
00090     9                 *      TIME UNIT = .001 MINUTES                                                *
00100    10                 *                                                                              *
00110    11                 ********************************************************************************

00130    13                 *      FAILURE AND REPAIR RANDOM VARIABLES

00150    15                 NTILX  FUNCTION   RN(PF$MACHNO),C2 NUMBER OF PLUGS 'TIL FAILURE
00160    16                 0,200/1,600

00180    18                 REPAIR FUNCTION   RN(PF$MACHNO),C2 REPAIR TIME (MEAN = 15 SEC)
00190    19                 0,100/1,400

00210    21                 *      FAILURE TRACE INFO (VERIFICATION).

00230    23                 TFAIL  MATRIX     MX,50,2          TRACE FAILURES (VERIFICATION)
00240    24                 FMACH  SYN        1                COLUMN 1: MACHINE ID
00250    25                 FTIME  SYN        2                COLUMN 2: FAILURE TIME

00270    27                 *      CONFIGURATION DEFINITION

00290    29                 MACH   EQU        1(10),F          MAX OF 10 MACHINES
00300    30                 FAIL   EQU        1(10),X          DITTO
00310    31                        INITIAL    X$LASTM,4        RUN WITH 4 PACKING MACHINES
```

Fig. 5 - Naive Model of Second Hypothetical System

```
GPSS/H  VP/CSS RELEASE 1.0 (UN261)       26 JUL 81  15:39:18    FILE: SPARKEZ

LINE# STMT#  IF DO  BLOCK#  *LOC   OPERATION  A,B,C,D,E,F,G   COMMENTS

00330    33                ********************************************************************************
00340    34                *                                                                              *
00350    35                *      PLUG FLOW LOGIC                                                         *
00360    36                *                                                                              *
00370    37                ********************************************************************************

00390    39             1          GENERATE   1,,300,,,1PF    1000 PLUGS/MIN
00400    40             2          ASSIGN     MACHNO,1,PF     START AT MACHINE NO 1

00420    42             3  MLOOP   GATE NU    PF$MACHNO,NEXTM BUSY MACHINE => TRY NEXT
00430    43             4          GATE FNV   PF$MACHNO,PACKIT  AVAIL => GO PACK PLUG
00440    44             5  NEXTM   ADVANCE    150             ELSE, PROCEED TO NEXT MACHINE
00450    45             6          ASSIGN     MACHNO+,1,PF    NEXT MACHINE NUMBER
00460    46             7          TEST G     PF$MACHNO,X$LASTM,MLOOP LOOP THRU ALL MACHINES

00480    48             8          SAVEVALUE  OFFEND+,1       FALL-THRU => RAN OFF THE END
00490    49             9          TERMINATE  0

00510    51            10  PACKIT  SEIZE      PF$MACHNO       GRAB MACHINE
00520    52            11          PRIORITY   1               DEPARTURES HIGHER THAN ARRIVALS
00530    53            12          ADVANCE    3               PACK AT 333/MIN
00540    54            13          RELEASE    PF$MACHNO       FREE MACHINE
00550    55            14          SAVEVALUE  PF$MACHNO-,1     DECREMENT FAILURE COUNTER
00560    56            15          TEST LE    X(PF$MACHNO),0,EXIT SKIP AHEAD IF NO FAILURE
00570    57            16          SAVEVALUE  NFAIL+,1        ONE MORE FAILURE
00580    58            17          MSAVEVALUE TFAIL,X$NFAIL,FMACH,PF$MACHNO RECORD MACH NO
00590    59            18          MSAVEVALUE TFAIL,X$NFAIL,FTIME,AC1  RECORD FAILURE TIME
00600    60            19          FUNAVAIL   PF$MACHNO       SIGNAL FAILURE
00610    61            20          ADVANCE    FN$REPAIR       REPAIR TIME ELAPSES
00620    62            21          FAVAIL     PF$MACHNO       MACHINE BACK ONLINE
00630    63            22          SAVEVALUE  PF$MACHNO,FN$NTILX  RESET NEW RANDOM FAILURE CTR
00640    64            23  EXIT    TERMINATE  0               EXIT SYSTEM

00660    66                ********************************************************************************
00670    67                *                                                                              *
00680    68                *      RUN CONTROL                                                             *
00690    69                *                                                                              *
00700    70                ********************************************************************************

00720    72            24          GENERATE   ,,,1,,1PF       CONTROL XACT
00730    73            25          ASSIGN     MACHNO,X$LASTM,PF NUMBER OF MACHINES
00740    74            26  ILUPE   SAVEVALUE  PF$MACHNO,FN$NTILX  INIT RANDOM FAILURE COUNTER
00750    75            27          LOOP       MACHNO$PF,ILUPE

00770    77            28          ADVANCE    10000           RUN FOR 10 MINUTES

00790    79            29          TERMINATE  1               SHUT DOWN


00810    81                        RMULT      111,333,555,777
00820    82                        START      1               RUN WITH 4 MACHINES
00830    83                        INITIAL    X$LASTM,5       FIVE MACHINES
00840    84                        RMULT      111,333,555,777,999
00850    85                        CLEAR      X$LASTM         CLEAR ALL BUT CONFIG COUNT
00860    86                        START      1               RUN WITH 5 MACHINES
00870    87                        END
```

Fig. 5 - Naive Model of Second Hypothetical System (Cont.)

3.2 Sophisticated Approach to the Second Hypothetical Example

A sophisticated model of the second hypothetical system is shown in Figure 6.

Structure of the Model - The sophisticated approach to modelling the second hypothetical system is motivated by combined discrete/continuous simulation techniques available in such languages as SLAM II

(Pritsker & Associates 1981). The essence of this approach is that we need only to concern ourselves with the dynamics of aggregate sparkplug flow, rather than with the flow of each and every sparkplug. The model shown in Figure 6 implements this approach. It operates as follows:

1. The model contains three segments, a surge tracking segment, a failure scheduling segment, and a timer segment.
2. The surge tracking segment operates as follows:
   A. A single Transaction is used to represent the initial surge of plug flow into the system. Subsequent surges within the system, which result from machine failures and repairs, are also modelled with a single Transaction per surge.
   B. A surge Transaction executes a loop, tracking each surge from its origin through each successive machine in the system. For each machine not currently in a state of failure, a new rate of operation is calculated, based on the magnitude of the surge and the capacity of the machine. If the rate of operation of a machine is changed from its previous rate, its estimated time of failure must be updated. This is accomplished by PREEMPTing and immediately RETURNing a Facility unique to each machine, which has been SEIZEd by a Transaction used to model failures in the failure scheduling segment of the model. When PREEMPTed, the failure Transaction is routed from the ADVANCE Block in which it currently resides (where estimated time until failure is elapsing) to a Block at which a new estimated time of failure is computed.
   C. For each change in the rate of operation of a machine, a corresponding adjustment is made to the magnitude of the surge represented by the surge tracking Transaction. If the system configuration is sufficient, nearly all surges will die out prior to reaching the last machine.
   D. When the surge Transaction reaches the end of the conveyor, statistics for non-zero surges off the end of the conveyor (into the barrel) are collected.
3. The failure scheduling segment operates as follows:
   A. At time zero, a failure scheduling Transaction is GENERATEd for each machine in the system. For each machine, a random sample is drawn, corresponding to the number of plugs until the next failure occurs. The failure scheduling Transaction SEIZEs a Facility, so it may subsequently be signalled of machine rate changes (via PREEMPT). Each failure Transaction then goes into an ADVANCE Block with an extremely large ADVANCE time, corresponding to "infinite" wait.
   B. When a surge tracking Transaction causes a change in the rate of operation of a machine, it PREEMPTs the corresponding failure Transaction out of the infinite wait ADVANCE Block and routes it to a Block called SCHEDF, at which an updated estimate is made of the time at which the machine will fail.
   C. At SCHEDF, an the number of plugs until the next failure of the machine is updated by subtracting the current rate of machine operation times the length of time the machine has been operating at this rate from the previous value of the failure counter.
   D. If a failure Transaction has been routed to SCHEDF because a machine has become idle, the failure Transaction returns to the infinite wait ADVANCE Block. Otherwise, the remaining number of plugs until failure are converted into an estimated time until failure, and the failure Transaction enters an ADVANCE Block, assuming that the proper time has been calculated.
   E. If no other changes take place, the failure Transaction will exit the ADVANCE Block and model machine failure. When a machine fails, a surge Transaction is SPLIT off to model increased downstream flow, and when the machine is repaired, a surge Transaction is SPLIT off to model decreased downstream flow.
   F. If rate changes take place while a failure Transaction is in the ADVANCE Block corresponding to the elapsing of estimated time until failure, the Transaction is PREEMPTed out of the ADVANCE Block and routed back to SCHEDF, where an updated estimated time until failure is calculated.
4. The timer segment generates a single Transaction after 10 minutes of simulated operation. The timer Transaction updates statistics for flows off the end of the conveyor and shuts down the model.

3.3 Second Hypothetical example - Comparison of Results

Due to space limitations, actual results from running the two models cannot be presented herein; however, comparative results will be briefly summarized. Results for the two models agreed within acceptable bounds. Slight differences were due to the effects of truncation in integer arithmetic in the sophisticated model. Run-time performance of the two models differed significantly. The naive solution required 13.5 times the COMMON storage and 105.8 times the CPU time required by the sophisticated solution. Selecting the proper view of the problem really paid off here.

4. GENERALITY OF SOPHISTICATED TECHNIQUES ILLUSTRATED

The techniques that have been illustrated in the sophisticated models for the two hypothetical systems are applicable to a wide class of problems. However, they do have one shortcoming, namely that as coded, they apply only to systems which have contain flows of homogeneous elements. The parts machined

in the first example are all identical, as are the sparkplugs in the second example. In many systems, components flowing through the system are heterogeneous. For such systems, two possible techniques exist. It is possible that the unique characteristics of components flowing through a system can be determined by random sampling at critical points. For example, we may know that 35 percent of the parts flowing through a system have a particular characteristic. If this is the case, sampling from a uniform distribution may enable us to avoid having to explicitly carry a unique attribute for each part. The examples shown can rather easily be extended to include such sampling.

In cases where attributes must be carried for each moving component, the techniques illustrated must be modified significantly. In GPSS, the easiest vehicle for implementing dynamic elements with unique attributes is, of course, the Transaction, with its associated Parameters. Much of the time and space efficiency in the techniques illustrated was achieved by relieving the overburdened GPSS simulator from having to manage excessive numbers of Transactions. What can one do? What is really needed here is a class of objects which have user-specified, user-accessible attributes, and which can be created, destroyed, and collected into sets, all under user program control. This description begins to sound very much like Simscript II.5 temporary entities and sets (Russell 1981). Temporary entities look very

```
GPSS/H  VP/CSS RELEASE 1.0 (UN261)       29 JUL 81   8:25:16    FILE: SPARKCD

LINE# STMT# IF DO  BLOCK# *LOC   OPERATION  A,B,C,D,E,F,G   COMMENTS

00010     1                      SIMULATE

00030     3                ********************************************************************************
00040     4                *                                                                              *
00050     5                *      SPARKPLUG PACKING LINE MODEL                                            *
00060     6                *      "COMBINED DISCRETE/CONTINUOUS APPROACH"                                 *
00070     7                *                                                                              *
00080     8                *      TIME UNIT = .001 MINUTES                                                *
00090     9                *                                                                              *
00100    10                ********************************************************************************

00120    12                *      PARAMETER DICTIONARY

00140    14                MACHNO EQU       1,PF           PACKING MACHINE NUMBER
00150    15                SURGE  EQU       2,PF           SURGE RATE (PLUGS/MINUTE)
00160    16                TLAST  EQU       3,PF           TIME OF LAST UPDATE

00180    18                *      MACHINE FAILURE MATRICES

00200    20                FAIL   MATRIX    MX,10,3
00210    21                INTO   SYN       1              COLUMN 1: RATE INTO MACH (PLUGS/MIN)
00220    22                MRATE  SYN       2              COLUMN 2: CURRENT MACHINE RATE
00230    23                NTILF  SYN       3              COLUMN 3: REMAINING PLUGS 'TIL FAILUR

00250    25                TFAIL  MATRIX    MX,50,2        TRACE FAILURES (VERIFICATION)
00260    26                FMACH  SYN       1              COLUMN 1: MACHINE ID
00270    27                FTIME  SYN       2              COLUMN 2: FAILURE TIME

00290    29                *      FAILURE AND REPAIR RANDOM VARIABLES

00310    31                NTILX  FUNCTION  RN(PF$MACHNO),C2 NUMBER OF PLUGS 'TIL FAILURE
00320    32                0,200/1,600

00340    34                REPAIR FUNCTION  RN(PF$MACHNO),C2 REPAIR TIME (MEAN = 15 SEC)
00350    35                0,100/1,400

00370    37                *      TABLE TO TABULATE FLOW OFF END OF MAIN CONVEYOR

00390    39                OFFEND TABLE     ((AC1-X$TLASTO)*X$ORATE/1000),50,50,10


00410    41                *      CONFIGURATION DEFINITION

00430    43                       INITIAL   X$LASTM,4      RUN WITH 4 PACKING MACHINES
00440    44                       STORAGE   S1-S10,333     MACHINES RUN AT 333 PLUGS/MIN
```

Fig. 6 - Sophisticated Model of Second Hypothetical System

```
    GPSS/H  VP/CSS RELEASE 1.0 (UN261)       29 JUL 81   8:25:16    FILE: SPARKCD

    LINE# STMT#  IF DO  BLOCK#  *LOC   OPERATION  A,B,C,D,E,F,G   COMMENTS

00460     46                ********************************************************************************
00470     47                *                                                                              *
00480     48                *      SURGE TRACKING SEGMENT                                                  *
00490     49                *                                                                              *
00500     50                ********************************************************************************

00520     52             1          GENERATE   ,,300,1,,3PF    XACT FOR INITIAL SURGE INTO SYSTEM
00530     53             2          ASSIGN     MACHNO,1,PF     START WITH MACHINE NO 1
00540     54             3          ASSIGN     SURGE,1000,PF   1000 PLUGS/MIN INTO SYSTEM

00560     56             4  MLOOP   MSAVEVALUE FAIL,PF$MACHNO,INTO,PF$SURGE  RATE INTO MACH
00570     57             5          GATE SV    PF$MACHNO,NEXTM SKIP IF MACHINE UNAVAIL

00590     59             6  BAKUP   TEST NE    PF$SURGE,S(PF$MACHNO),NODIF SKIP IF NO RATE CHANGE
00600     60             7          SAVEVALUE  NEWRATE,S(PF$MACHNO)+R(PF$MACHNO) MAX RATE
00610     61             8          TEST G     X$NEWRATE,PF$SURGE,*+2 SKIP IF SURGE >= MAX
00620     62             9          SAVEVALUE  NEWRATE,PF$SURGE SURGE < MAX
00630     63            10          LEAVE      PF$MACHNO,S(PF$MACHNO) EMPTY STORAGE
00640     64            11          ENTER      PF$MACHNO,X$NEWRATE NOW SET NEW RATE
00650     65            12          PREEMPT    PF$MACHNO,,SCHEDF  WAKE UP FAILURE XACT
00660     66            13          RETURN     PF$MACHNO       IMMEDIATELY RETURN MACHINE FACIL

00680     68            14  NODIF   ASSIGN     SURGE-,S(PF$MACHNO),PF REDUCE DOWNSTREAM SURGE

00700     70            15  NEXTM   ADVANCE    150             TIME TO GET TO NEXT MACHINE
00710     71            16          ASSIGN     MACHNO+,1,PF    NEXT MACHINE NO
00720     72            17          TEST G     PF$MACHNO,X$LASTM,MLOOP LOOP THRU ALL MACHINES

00740     74            18          TEST NE    X$ORATE,0,NOTAB SKIP TABULATION IF CURRENT RATE = 0
00750     75            19          TABULATE   OFFEND          TABULATE PREV SURGE OFF END
00760     76            20  NOTAB   SAVEVALUE  TLASTO,AC1      TIME OF LAST RATE CHANGE = NOW
00770     77            21          SAVEVALUE  ORATE,PF$SURGE  NEW RATE OFF END
00780     78            22          TERMINATE  0               END OF SURGE

00800     80                ********************************************************************************
00810     81                *                                                                              *
00820     82                *      MACHINE FAILURE SCHEDULING MECHANISM                                    *
00830     83                *                                                                              *
00840     84                ********************************************************************************

00860     86            23          GENERATE   ,,,X$LASTM,,3PF ONE XACT PER MACHINE
00870     87            24  HERE    ASSIGN     MACHNO,N$HERE+1,PF ASSIGN MACHINE ID
00880     88            25          SEIZE      PF$MACHNO       USE FACILITY SO WE CAN PREEMPT
00890     89            26          MSAVEVALUE FAIL,PF$MACHNO,NTILF,FN$NTILX NO 'TIL 1ST FAIL
00900     90            27  INFWT   ADVANCE    1000000         "INFINITE" WAIT
00910     91            28          SEIZE      X$ERROR         ERROR IF WE GET HERE

00930     93            29  SCHEDF MSAVEVALUE FAIL-,PF$MACHNO,NTILF,
00940     94            29                     MX$FAIL(PF$MACHNO,MRATE)*MP$TLAST$PF/1000
00950     95                *      ABOVE LINE UPDATES REMAINING NO OF PLUGS UNTIL FAILURE

00970     97            30          MSAVEVALUE FAIL,PF$MACHNO,MRATE,S(PF$MACHNO) CURRENT RATE
00980     98            31          MARK       TLAST$PF        TIME OF LAST RATE CHANGE = NOW
00990     99            32          TEST G     S(PF$MACHNO),0,INFWT  GO WAIT IF MACHINE NOW IDLE

01010    101                *      THE FOLLOWING ADVANCE BLOCK CORRESPONDS TO THE TIME
01020    102                *      UNTIL A MACHINE FAILURE OCCURS, ASSUMING NO CHANGES IN
01030    103                *      RATE OF OPERATION.  IF A RATE CHANGE TAKES PLACE, AN XACT
01040    104                *      IN THE ADVANCE BLOCK WILL BE PREEMPTED OFF THE FUTURE EVENTS
01050    105                *      CHAIN AND ROUTED TO THE "SCHEDF" BLOCK.

01070    107            33          ADVANCE    MX$FAIL(PF$MACHNO,NTILF)*1000/S(PF$MACHNO)
```

Fig. 6 - Sophisticated Model of Second System (Cont.)

```
    GPSS/H  VP/CSS RELEASE 1.0 (UN261)       29 JUL 81   8:25:16    FILE: SPARKCD

    LINE# STMT#  IF DO  BLOCK#  *LOC   OPERATION  A,B,C,D,E,F,G   COMMENTS

01090   109                 *       FAILURE HAS OCCURRED.

01110   111              34         SAVEVALUE  NFAIL+,1          ONE MORE FAILURE
01120   112              35         MSAVEVALUE TFAIL,X$NFAIL,FMACH,PF$MACHNO RECORD MACH NO
01130   113              36         MSAVEVALUE TFAIL,X$NFAIL,FTIME,AC1  RECORD FAILURE TIME
01140   114              37         ASSIGN     SURGE,MX$FAIL(PF$MACHNO,INTO),PF CURRENT RATE INTO
01150   115              38         SPLIT      1,NEXTM           PROPAGATE DOWNSTREAM SURGE
01160   116              39         LEAVE      PF$MACHNO,S(PF$MACHNO)  EMPTY STORAGE => DOWN
01170   117              40         MSAVEVALUE FAIL,PF$MACHNO,MRATE,0  DOWN => RATE=0
01180   118              41         SUNAVAIL   PF$MACHNO         MAKE MACHINE UNAVAIL
01190   119              42         ADVANCE    FN$REPAIR         REPAIR TIME
01200   120              43         SAVAIL     PF$MACHNO         MACHINE NOW AVAIL AGAIN
01210   121              44         ASSIGN     SURGE,MX$FAIL(PF$MACHNO,INTO),PF CURRENT RATE INTO
01220   122              45         SPLIT      1,BAKUP           ROUTE SURGE TO THIS MACHINE
01230   123              46         MSAVEVALUE FAIL,PF$MACHNO,NTILF,FN$NTILX NEXT FAIL
01240   124              47         TRANSFER   ,INFWT            GO WAIT FOR NEXT "SIGNAL"

01260   126                 ********************************************************************************
01270   127                 *                                                                              *
01280   128                 *       RUN CONTROL                                                            *
01290   129                 *                                                                              *
01300   130                 ********************************************************************************

01320   132              48         GENERATE   ,,10000,1,,3PF    RUN FOR 10 MINUTES
01330   133              49         TEST NE    X$ORATE,0,*+2     TABULATE IF FLOWING OFF END
01340   134              50         TABULATE   OFFEND            LAST UPDATE OF "OFF END"
01350   135              51         TERMINATE  1                 SHUT DOWN

01370   137                         RMULT      111,333,555,777
01380   138                         START      1                 RUN WITH 4 MACHINES
01390   139                         INITIAL    X$LASTM,5         FIVE MACHINES
01400   140                         RMULT      111,333,555,777,999
01410   141                         CLEAR      X$LASTM           CLEAR ALL BUT CONFIG COUNT
01420   142                         START      1                 RUN WITH 5 MACHINES
01430   143                         END
```

Fig. 6 - Sophisticated Model of Second Hypothetical System (Cont.)

attractive both from a time and space efficiency standpoint. In GPSS/H (Henriksen 1978), for example, a Transaction requires 56 bytes of storage for simulator internal data. This overhead is above and beyond the storage required for representation of (user-requested) Transaction Parameters. Thus, the storage overhead for creating a Transaction object is substantial. As the comparative results for naive and sophisticated models shown above indicate, the time overhead in requiring the GPSS simulator to manage large numbers of Transactions can also be quite large.

Barring major additions to the GPSS language, one can often make use of existing capabilities. For example, Matrix Savevalues can be used to record the attributes of a collection of entities, with one row per entity and columns for each required attribute. The number of rows required can often be determined prior to running a model, because it may relate to some real constraint on the number of entities in a part of the system. Of course, the use of Matrix Savevalues requires user-provided code for "allocating" and "releasing" rows in a matrix, corresponding to the creation and destruction of an object.

5. CAVEATS

The sample programs which have been shown are intended only to illustrate basic programming style. Systems of the type illustrated are fraught with problems of random sampling, validation, and analysis of output. Because of the high variance in system performance, sophisticated techniques such as batch means, multiple replications, autoregressive analysis, etc. would almost certainly be required in "real-world" models. For a discussion of such topics, see reference (Law 1981).

The sample programs were run under GPSS/H on the National CSS network, where execution times are measured in ARU's (application resource units). Readers who attempt to reproduce results of the programs will doubtless experience different timings on other systems. In addition, certain GPSS/H

features, such as symbolic names for Parameters, have been used to improve program readability. Other versions of GPSS may not contain the extended features of GPSS/H.

6. SUMMARY

This paper has illustrated two interesting techniques for application of GPSS to problems encountered in manufacturing systems. In both cases, naive solutions to the problems at hand are readily suggested by the GPSS world-view; however the world-view steers us in the wrong direction. Fortunately, more sophisticated techniques, which are far more time- and space-efficient, can be achieved with nearly equal ease in GPSS. The message should be clear both to the novice and expert GPSS programmer: don't let yourself get into a conceptual rut, solving all your modelling problems with techniques you have seen or applied in the past. You may be disastrously off the mark.

ACKNOWLEDGEMENTS

The "surge" technique used in the second example was first brought to my attention by Dr. Ed Russell of CACI. My thoughts on the combined discrete/continuous approach to the second example were sharpened by a conversation with Dr. Charles Standridge of Pritsker & Associates.

REFERENCES

Franta, W. R. (1977), The Process View of Simulation, North-Holland, New York

Henriksen, J. O. (1978), GPSS/H User's Manual, Wolverine Software Corporation, P.O. Box 1251, Falls Church, VA 22041

Ingerman, D. (1981), Using Chains and Groups to Make GPSS More Efficient, Proceedings of the 1981 Winter Simulation Conference

Law, A. M. and Kelton, D. W. (1981), Simulation Modeling and Analysis, (Scheduled for publication in Fall, 1981)

Pritsker & Associates (1981), The SLAM II User's Manual

Russell, E. C. (1981), Building Simulation Models with Simscript II.5, CACI, Inc., Los Angeles

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