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

THE MODELING OF PERFECT SEQUENCING FLEXIBILITY IN A SCHEDULING ENVIRONMENT

Thomas J. Schriber

Computer and Information Systems
Graduate School of Business Administration
The University of Michigan
Ann Arbor, Michigan 48109-1234

ABSTRACT

The concept of sequencing flexibility and its potential importance in such scheduling environments as those of manufacturing systems is reviewed. The statement of a manufacturing problem is provided, asking that a simulation model be built for use in studying system performance under conditions of perfect sequencing flexibility when the rule used to dispatch jobs to machines is FRS (fewest remaining steps). A workable approach to modeling perfect sequencing flexibility is then described. Finally, a GPSS/H model illustrating the implementation of this approach in the setting of the stated problem is presented.

1 SEQUENCING FLEXIBILITY

Suppose that manufacturing a particular product requires that one step be performed on a unit of material by each of 5 different machines. If these 5 steps must be carried out in a strict sequence, then there is *no* sequencing flexibility in the manufacturing process. If the 5 steps can be carried out in any sequence whatsoever, then there is *perfect* sequencing flexibility in the process. If some of the 5 steps must be carried out in a strict sequence, but others can be performed without adhering to a strict sequence, then the manufacturing process has *partial* sequencing flexibility.

Rachamadugu and Schriber (1990a, 1990b) have proposed a squencing flexibility measure (SFM) that quantifies the degree of sequencing flexibility in a multistep process on a scale from 0 to 1, with measures of 0 and 1 corresponding respectively to the extreme cases of *no* flexibility and *perfect* flexibility, as described above. For an example in which their metric is applied to *partial* sequencing flexibility, consider the following scenario: *Operations 1 through 5 must be carried out to do a job. Operation 1 must precede 2 and 2 must precede 3. Similarly, operation 4 must precede 5. But there are no precedence requirements otherwise.* Using their metric, an SFM value of 0.6 results for this situation. (See either referenced paper for quantitative details.)

There are important potential benefits of exploiting whatever sequencing flexibility might be inherent in a product structure. Suppose for example that the machining resources in a flexible manufacturing system (FMS) are used to make a variety of products concurrently. A given unit of work-in-process (WIP) can wait *logically* at each of two or more alternative types of machines if there is sequencing flexibility at that stage in the manufacturing process for the WIP unit. This increases the likelihood that a next step can be performed on the WIP earlier than if there were no sequencing flexibility. In turn, it is likely that this will improve such system operating characteristics as the elapsed time between release of a job to the system and completion of the job (system residence time).

System residence time and other important measures of system performance depend not just on sequencing flexibility but also on the choice of dispatching rule (the rule used to decide which unit of waiting work to send to a machine when the machine next becomes free). This makes it of interest to evaluate and rank various dispatching rules under conditions of sequencing flexibility with respect to such measures of system performance as system residence time and such tardy-job characteristics as the percentage of jobs that are tardy and the probability distribution followed by the tardy-time response variable. Rachamadugu and Schriber (1990a, 1990b) have shown that the relative goodness of well-known dispatching rules does depend on whether sequencing flexibility is present and exploited. For example, the shortest processing time (SPT) rule minimizes a job's system residence time in a particular simulated setting relative to seven other dispatching rules when there is *no* sequencing flexibility (or when it exists but is not exploited), whereas the least work remaining (LWR) rule minimizes this measure when there is *perfect* sequencing flexibility. (In the SPT rule, that waiting WIP which needs *the machine* for the shortest time is the next to be dispatched to the machine. In the LWR rule, that waiting WIP which requires the least *total* remaining machining time is the next to be dispatched to the machine.)

As Rachamadugu and Schriber (1990b) point out, "there seem to be no known exact or approximate prescriptive models for analyzing the performance of scheduling rules under perfect or near perfect sequencing flexibility scenarios." This motivates the use of simulation models to study the performance of dispatching rules under conditions of sequencing flexibility.

It is the *modeling* of perfect sequencing flexibility on which our attention centers here. The logical issues involved in building a model for perfect sequencing flexibility are brought into focus by the request to model the system described in the next section.

2 STATEMENT OF A PROBLEM

A particular flexible manufacturing system consists of five different machines. These machines are used to build to order a multiplicity of products that are manufactured concurrently. Each order (job) is for one unit of product and visits from 1 to 5 machines (uniformly distributed), visiting no machine more than one time. The particular machines visited by a job are determined at random. Operation times for all jobs at all machines are assumed to be identically 2-Erlang distributed, with a mean of 30 minutes per operation. Job interarrival times are exponentially distributed and have a mean such that the expected overall machine utilization in the system is 90 percent. The operations required by each job can be performed in any order. The rule used to dispatch jobs to machines is fewest remaining steps (FRS). That is, the waiting job which has the fewest remaining steps to be performed on it before being completed is the next job to be dispatched to a machine that has just become free. The flow allowance factor used in determining a job's due date is 7.5. (A job's due date is determined by adding to its arrival time 7.5 times its total step time.)

Build a model that can be used to simulate the operation of this system. Design the model to measure the percentage of jobs that are tardy, the mean and standard deviation of the tardy-time random variable, and the mean and standard deviation of the system-residence-time random variable. (Base the system residence time measure on all jobs, whether they are tardy or not). Assume only one step at a time can be performed on a given job.

Perform a single simulation with the model. Initialize the model by simulating until 1000 jobs have been completed. Then continue the simulation until another 2500 jobs have been completed. For each 100 post-initialization jobs completed, report the cumulative number of such jobs completed, the cumulative number of tardy jobs, the cumulative percentage of jobs tardy, and the cumulative means and standard deviations described above. (Figure 2 shows such a report.)

3 A SOLUTION APPROACH

The following points make up the main logical considerations that need be taken into account in building a model for the type of problem described in section 2.

1. When a job arrives, it needs to record its time of arrival and then sample to determine how many steps are to be performed on it, which machines are to be used, and what the step times are to be. Total step time and the due date follow from the individual step times and the flow allowance factor. All of these individual pieces of information can take the form of attributes carried by the job itself.
2. The job can then create enough identical copies (clones) of itself to provide one sub-job for each step that must be performed on the overall job. The clones should inherit from the original the attributes described above. (The original job itself can serve as one of these sub-jobs. If the original job only requires one step, then the original creates no clones and the original is the one and only "sub-job" in this case.) Each sub-job will then be a one-step job associated with the overall job. When all the sub-jobs have been finished, then the overall job is finished.

   Note that an attribute of each sub-job will be the total number of steps remaining for the *overall* job. In the fewest-steps-remaining dispatching rule, the value of this attribute for all sub-jobs waiting for a given machine will be used to determine which sub-job is the next to get the machine.
3. Before the sub-jobs making up an overall job are sent to their individual machines, a mechanism must be established so that they can send messages among themselves at appropriate times. The following messaging needs to take place:
   a. When a step is about to start on a sub-job, the sub-job must signal this fact to all associated sub-jobs to suspend their candidacy for machine use.
   b. When the processing of a sub-job is finished, it must let all associated sub-jobs know that they are candidates once again for machine use.
   c. When the processing of a sub-job is finished, it must update the "number of steps remaining" attribute on all associated sub-jobs.

   The needed messaging can be supported by having the sub-jobs making up a job become members of a set unique to that job. Of course these sub-jobs already are members of such a set *conceptually*; but they must also be made members of such a set *operationally*, to support the required messaging. If operational sets of this type are not provided in the modeling language used to implement the simulation, the modeler will have to work with other language elements to achieve the needed messaging capability.
4. When a sub-job finishes using its machine, it can test to determine whether it is the last member of its set. If so, the job is finished and this last surviving sub-job needs to record its system residence time and its tardiness (if it is tardy) before leaving the model. But if this sub-job is survived by others in its set, then it needs to send the messages corresponding to 3b and 3c above before leaving the model.

5. When a sub-job finishes using its machine, there is the need to dispatch a next waiting sub-job to the now-idle machine. The departing sub-job itself can take this responsibility or a "watchdog" not connected with any orders at all can take this responsibility. In either approach, sub-jobs waiting for the machine must be ranked on their number-of-steps-remaining (for their overall job) attribute *just before* the decision is made about which sub-job is the next to get the machine. (A sub-job's correct position in the ranking *cannot be determined at the time the sub-job arrives at a machine*, because the number-of-steps-remaining for one or more waiting sub-jobs may change before the machine's current user finishes. This is why the correct ranking must be determined just prior to dispatching a next waiting sub-job to the machine.)

The foregoing points make up the main logical considerations in modeling *perfect* sequencing flexibility for the *fewest-steps-remaining* dispatching rule. The logic required may be simpler for perfect sequencing flexibility with *other* dispatching rules. (If SPT is the dispatching rule, for example, sub-jobs can be ranked *at the time of their arrival at their machine*, and neither the messaging described in 3c nor the requirement described in 5 arise.) The logic may be more demanding, however, in cases of *partial* sequencing flexibility, irrespective of the dispatching rule involved. For the extreme of *no* sequencing flexibility, the logic is very straightforward.

4 A GPSS/H MODEL FOR THE PROBLEM

The logic outlined in section 3 and applied to the section 2 problem is implemented in the GPSS/H model displayed in Figure 1, with the resulting output shown in Figure 2. In addition to showing the model itself, Figure 1 provides an appended column of block numbers (labeled BLOCK#) at the left, to support discussion, and of column labels (LOCATION, OPERATION, etc.) across the top of each part of the figure. Figure 1 shows the logic for use of the first of the five machines; similar logic applies to the other four machines, and has been excluded from Figure 1 to save space.

Space restrictions permit no more than brief discussion of the GPSS/H model, but the comments embedded liberally in the model itself (the Figure 1 comments are in lower case and begin with an asterisk) should make it quite easy (for a person familiar with any discrete-event modeling *language*, and certainly for anyone familiar with GPSS) to understand the underlying details. Parts 1 and 2 of Figure 1 provide the setting of the model. Handling of considerations 1 through 5 of section 3 is commented on below under corresponding numbers.

1. The considerations under 1 are handled in Blocks 1 through 27, parts 3 and 4 of Figure 1. Blocks 8 and 9 (ADVANCE and SPLIT) structure the job arrival process, Blocks 13 through 21 determine how many and which machines a job needs, Blocks 22 through 26 handle step times, and Block 27 sets the due date.
2. Block 31 (SPLIT) in part 4 of Figure 1 provides for sub-jobs (clones) that inherit the attributes of the original job.
3. Block 33 (JOIN) in part 4 of Figure 1 puts a job's sub-jobs into a unique set (a GPSS Group) to support the messaging needs described earlier. Block 39 (ALTER) in part 5 handles the need to let sub-jobs in the set know that a step is starting on a member of their set. Block 44 (ALTER) accomplishes the reverse effect. Block 43 (ALTER) is used to update the number-of-remaining-steps attribute of set members.
4. A completed sub-job uses Block 45 (REMOVE) in part 5 to remove itself from its sub-job set, then uses Block 49 (TEST) to determine if it is survived by other sub-jobs in the set. If so, it leaves the model (Block 50, TERMINATE); otherwise, it transfers to Block 111 (TABULATE) to record statistics on the now-finished overall job, and then leaves the model.
5. In this model, a finished sub-job sends another waiting sub-job (if any) to use the machine it just made free. This is accomplished with Blocks 46, 47 and 48 (UNLINK; PRIORITY; UNLINK) in part 5 of Figure 1. At Block 46, the finished sub-job unlinks all sub-jobs waiting for that machine from their place of waiting (a User Chain) and targets them to be relinked in their place of waiting, ranked ascending on the number of remaining steps for the associated overall orders. At Block 47, the finished sub-job pauses while the unlinked sub-jobs are relinked at Block 37 (LINK). At Block 48, the finished sub-job unlinks the most highly qualified waiting sub-job and routes it to capture the machine.

If a machine is busy when a sub-job arrives, or if an associated sub-job is already using a machine, then the arriving sub-job should begin waiting for the machine. Otherwise, it should capture the machine. These logical requirements are handled by Blocks 36 (TEST) and 37 (LINK). Block 36 refers to a Boolean expression (named DELAY) in which the compound condition described above (is the machine busy or is an associated sub-job currently using a machine?) is evaluated. If the Boolean expression is true, the sub-job begins waiting for the machine at Block 37 (LINK); otherwise, it captures the machine without delay at Block 38 (SEIZE).

Part 7 of Figure 1 shows run control and the steps for producing the simulation report given in Figure 2.

(The paper concludes with References and the Author's Biography given four pages hence.)

```
BLOCK# | LOCATION OPERATION  OPERANDS           COMMENTS
       |*********************************************************************
       |         SIMULATE                          Base Time Unit: 1 Minute
       |*        Sequencing Flexibility Measure:  1.0                       *
       |*        Service Order: Fewest Remaining Steps                      *
       |*        Number of Machines in the System:  5                       *
       |*        Number of Machines Used per Job:  From 1 to 5 (Random)     *
       |*        Mean Machining Time:  30 Minutes, 2-Erlang Distributed     *
       |*        Arrival Process:  Poisson                                  *
       |*        Expected Machine Utilization:  90%                         *
       |*********************************************************************
       |*
       |*********************************************************************
       |*        Compiler Directives                                        *
       |*********************************************************************
       |*
       |*      ...reallocate default maximum quantities of entities...
       |         REALLOCATE COM,20000,_    number of bytes in COMMON
       |                    GRP,4000       number of Transaction (Xact) Groups
       |*
       |*      ...selected correspondences between
       |*         symbolic and numeric identifiers...
       |*
       |*      ...Facilities (the machines)...
       | MAC1    EQU        1,F      MAC1 is Facility 1
       | MAC2    EQU        2,F         ...and so on...
       | MAC3    EQU        3,F
       | MAC4    EQU        4,F
       | MAC5    EQU        5,F
       |*
       |*      ...Fullword Integer Parameters (variables local to Xacts)...
       | MACID1  EQU        1,PF   MACID1 is Fullword Integer Parameter 1
       | MACID2  EQU        2,PF       ...and so on...
       | MACID3  EQU        3,PF
       | MACID4  EQU        4,PF
       | MACID5  EQU        5,PF
       |*
       |*      ...Real Parameters (variables local to Xacts)...
       | STEPTYM1 EQU       1,PL  STEPTYM1 is Real Parameter 1
       | STEPTYM2 EQU       2,PL       ...and so on...
       | STEPTYM3 EQU       3,PL
       | STEPTYM4 EQU       4,PL
       | STEPTYM5 EQU       5,PL
       |*
       |*      ...Integer Variables (global variables)...
       |         INTEGER    &I             DO-loop counter
       |         INTEGER    &INDEX1        integer value from U(1,5)
       |         INTEGER    &INDEX2        integer value from U(1,5)
       |         INTEGER    &INITJOBS      number of initialization jobs
       |         INTEGER    &JOBCOUNT      jobs-arrived counter
       |         INTEGER    &LUPCOUNT      Xact-loop counter
       |         INTEGER    &MACHINES      number of machines in system
       |         INTEGER    &STEP          number of job's current step
       |         INTEGER    &STEPS         number of job's total steps
       |         INTEGER    &TEMPSTOR      temporary storage location
```

Figure 1: The GPSS/H Model (part 1 of 7)

```
BLOCK#  LOCATION OPERATION  OPERANDS              COMMENTS
        *      ...Real Variables (global variables)...
                  REAL       &FLOFAKTR     multiplier for due date
                  REAL       &JOBIAT       mean job interarrival time
                  REAL       &MSTEPTYM     mean step time (machining time)
                  REAL       &ESYSUTIL     expected system utilization
        *
        *      ...Synonyms (identifiers for integer constants)...
         BUSY     SYN        1         code for a busy job
         IDLE     SYN        0         code for an idle job
        *
        *      ...suppression of Control-Statement echoes...
                  UNLIST     CSECHO
        *
        *      ...assignment of values to selected global variables...
                  LET        &ESYSUTIL=0.90    expected machine utilization
                  LET        &FLOFAKTR=7.5     flow allowance factor
                  LET        &INITJOBS=1000    number of initialization jobs
                  LET        &MACHINES=5       5 machines
                  LET        &MSTEPTYM=30.0    mean machining time, minutes
        *
                  LET        &JOBIAT=_         job interarrival time, minutes
                      (FLT(&MACHINES+1)/2)*&MSTEPTYM/(&MACHINES*&ESYSUTIL)
        *
        **********************************************************************
        *         Control Statements                                         *
        **********************************************************************
        *
        *      ...Boolean expressions...
        *
        *      ...the DELAY expression is true if this step-Xact
        *         must wait to use this machine...
         DELAY    BVARIABLE  (PF(STATUS)=BUSY)OR(FU(PF(MYMAC)))
        *
        *      ...the QUALIFY expression is true if this step-Xact's
        *         job-group is inactive (idle)...
         QUALIFY  BVARIABLE  PF(STATUS)=IDLE
        *
        *      ...Block labels as a function of machine being checked...
         CHEKMACS FUNCTION   PF(MYMAC),D5
        MAC1,CHEKMAC1/MAC2,CHEKMAC2/MAC3,CHEKMAC3/MAC4,CHEKMAC4/MAC5,CHEKMAC5
        *
        *      ...Initial Positions of U(0,1) random-number generators...
                  RMULT      100000,_   RN1 (interarrival times)
                             200000,_   RN2 (permutation index 1)
                             300000,_   RN3 (permutation index 2)
                             400000,_   RN4 (number of machines a job visits)
                             500000,_   RN5 (first piece of 2-Erlang)
                             600000     RN6 (second piece of 2-Erlang)
        *
        *      ...Table Statements...
        *
        *      ...system residence time,all jobs, minutes...
         SYSTYMS  TABLE      (AC1-PL(TIMEIN))/60.0,0,1,2
        *      ...tardy time for tardy jobs, minutes...
         TARDTYMS TABLE      (AC1-PL(DUEDATE))/60.0,0,1,2
```

Figure 1 (continued): The GPSS/H Model (part 2 of 7)

```
BLOCK#  LOCATION OPERATION  OPERANDS              COMMENTS
        *
        *********************************************************************
        *         Job Creation and Specification Segment                    *
        *********************************************************************
        *
        *      ...seed the segment with a master job-Xact (Transaction)...
   1              GENERATE   0,,,1,25,10PF,8PL
        *
        *      ...put the 5 machine identifiers in 5 Fullword Parameters...
   2              ASSIGN     MACID1,MAC1,PF
   3              ASSIGN     MACID2,MAC2,PF
   4              ASSIGN     MACID3,MAC3,PF
   5              ASSIGN     MACID4,MAC4,PF
   6              ASSIGN     MACID5,MAC5,PF
        *
        *      ...a job is idle at the time of its arrival...
   7              ASSIGN     STATUS,IDLE,PF
        *
        *      ...interarrival time elapses...
   8     NEXTJOB  ADVANCE    RVEXPO(1,&JOBIAT)
        *
        *      ...create a successor job-Xact;
        *         route it to experience it's interarrival time...
   9              SPLIT      1,NEXTJOB
        *
        *      ...record the time of this job's arrival at the system...
  10              ASSIGN     TIMEIN,AC1,PL
        *
        *      ...update the job count, then give this job a unique id number...
  11              BLET       &JOBCOUNT=&JOBCOUNT+1
  12              ASSIGN     JOBID,&JOBCOUNT,PF
        *
        ****** Permute the Sequence of Machine ID's Carried by This Job ******
        *
  13              BLET       &LUPCOUNT=10        set &LUPCOUNT = 10
        *
        *      ...set &INDEX1 and &INDEX2 equal to samples drawn from
        *         uniform distributions of integers ranging from 1 to 5...
  14     SHUFFLE  BLET       &INDEX1=RN2@5+1
  15              BLET       &INDEX2=RN3@5+1
        *
        *      ...now swap the machine id's carried in the Xact Parameters
        *         corresponding to these random indices...
  16              BLET       &TEMPSTOR=PF(&INDEX1)
  17              ASSIGN     &INDEX1,PF(&INDEX2),PF
  18              ASSIGN     &INDEX2,&TEMPSTOR,PF
        *
        *      ...update the loop count and repeat until done 10 times...
  19              BLET       &LUPCOUNT=&LUPCOUNT-1
  20              TEST E     &LUPCOUNT,0,SHUFFLE
        *
        ******************** End of Permutation Logic **********************
```

Figure 1 (continued): The GPSS/H Model (part 3 of 7)

```
BLOCK# | LOCATION OPERATION  OPERANDS                 COMMENTS
       | *     ...set the number of steps (machines) for this job...
  21   |           BLET       &STEPS=RN4@5+1
       | *
       | ******* Loop to Determine the 2-Erlang Step Time for Each Step *******
       | ******** this Job Requires;  also Accumulate Total Step Time *********
       | *
  22   |           BLET       &STEP=1
  23   | GETSTIME  ASSIGN     &STEP,_
  23   |                      RVEXPO(5,&MSTEPTYM/2)+RVEXPO(6,&MSTEPTYM/2),PL
  24   |           ASSIGN     TOTSTIME+,PL(&STEP),PL
  25   |           BLET       &STEP=&STEP+1
  26   |           TEST G     &STEP,&STEPS,GETSTIME
       | *
       | *********************** End of Step-Time Loop ***********************
       | *
       | *     ...assign this job's due date...
  27   |           ASSIGN     DUEDATE,AC1+&FLOFAKTR*PL(TOTSTIME),PL
       | *
       | *     ...create clones so there is one step-Xact for each step
       | *        (create no clones for a one-step job),
       | *        numbering the step-Xacts serially in a Fullword Parameter...
  28   |           ASSIGN     STEPS2GO,&STEPS,PF
       | *
  29   |           TEST E     PF(STEPS2GO),1,GOSPLIT
  30   |           ASSIGN     SERIALNO,1,PF
       | *
  31   | GOSPLIT   SPLIT      PF(STEPS2GO)-1,NEXTBLOK,(SERIALNO)PF
       | *
  32   | NEXTBLOK  ASSIGN     MYMAC,PF(PF(SERIALNO)),PF
       | *
       | *     ...each step-Xact joins a Group unique to this job...
  33   |           JOIN       PF(JOBID)
       | *
       | *     ...pause while each other step-Xact for this job
       | *        joins this group...
  34   |           PRIORITY   PR,BUFFER
       | *
       | *     ...route each step-Xact to machine corresponding to its step...
  35   |           TRANSFER   ,FN(CHEKMACS)
```

Figure 1 (continued): The GPSS/H Model (part 4 of 7)

REFERENCES

Rachamadugu, R. and T.J. Schriber. 1990a. Performance of Dispatching Rules Under Perfect Sequencing Flexibility. In: *Proceedings of the 1990 Winter Simulation Conference*, eds. O. Balci, R.P. Sadowski, and R.E. Nance. The Society for Computer Simulation, San Diego, CA, 653-658.

Rachamadugu, R. and T.J. Schriber. 1990b. Performance of Nondelay Schedules: Generalized Open Shops. Working Paper No. 651, Division of Research, University of Michigan, Ann Arbor MI.

Schriber, T.J. 1991. *An Introduction to Simulation Using GPSS/H* (with Student DOS GPSS/H on an included disk). John Wiley & Sons, Inc., New York.

AUTHOR BIOGRAPHY

THOMAS J. SCHRIBER is Professor and Chairman of Computer and Information Systems in the Graduate School of Business at The University of Michigan. He teaches, does research, and consults in the area of discrete-event simulation. He has authored or co-authored several dozen articles, has authored or edited eleven books, including *An Introduction to Simulation Using GPSS/H* (Wiley, 1991), and regularly teaches intensive courses on GPSS-based simulation. From 1977 to 1986 he was the ACM member of the Board of Directors of the Winter Simulation Conferences, serving as Board Chairman two years. His professional affiliations include ACM, DSI, ORSA, SCS, and TIMS.

```
BLOCK# | LOCATION OPERATION OPERANDS             COMMENTS
       | *
       | ***********************************************************************
       | *        Use of Machine 1                                             *
       | ***********************************************************************
       | *
       | *      ...branch to use Machine 1 without delay if possible;
       | *         else, go into waiting line (User Chain) ranked
       | *         in order of increasing number of remaining steps...
    36 |  CHEKMAC1 TEST E     BV(DELAY),1,GETMAC1
    37 |  LINEFOR1 LINK       MAC1LINE,(STEPS2GO)PF
       | *
       | *      ...capture machine 1...
    38 |  GETMAC1  SEIZE      MAC1
       | *
       | *      ...message other step-Xacts in this job-group
       | *         that this job is now becoming active...
    39 |           ALTER      PF(JOBID),ALL,(STATUS)PF,BUSY
       | *
       | *      ...use the machine, free it, and update the
       | *         remaining number of steps this job requires...
    40 |           ADVANCE    PL(PF(SERIALNO))
    41 |           RELEASE    MAC1
    42 |           ASSIGN     STEPS2GO-,1,PF
       | *
       | *      ...message updated number of remaining steps to
       | *         other step-Xacts in this job-group...
    43 |           ALTER      PF(JOBID),ALL,(STEPS2GO)PF,PF(STEPS2GO)
       | *
       | *      ...message other step-Xacts in this job-group
       | *         that this job has now become inactive...
    44 |           ALTER      PF(JOBID),ALL,(STATUS)PF,IDLE
       | *
       | *      ...remove this step-Xact from this job-group...
    45 |           REMOVE     PF(JOBID)
       | *
       | ******** This Step-Xact Now Sends (tries to send) a Successor ********
       | ********* for Itself to Use Machine 1, then Leaves the Model *********
       | *
       | *      ...unlink all step-Xacts (if any) from the User Chain
       | *         of step-Xacts waiting for Machine 1...
    46 |           UNLINK     MAC1LINE,LINEFOR1,ALL
       | *
       | *      ...pause to put these step-Xacts back on the User Chain,
       | *         ranked ascending by the remaining number of steps
       | *         in their job-group...
    47 |           PRIORITY   PR,BUFFER
       | *
       | *      ...now unlink the first qualifying step-Xact from the front
       | *         of the User Chain and send it to capture Machine 1...
    48 |           UNLINK     MAC1LINE,GETMAC1,1,BV(QUALIFY)
       | *
       | *      ...if this was the last step for this job, branch to JOBDONE;
       | *         else, destroy this step-Xact...
    49 |           TEST NE    G(PF(JOBID)),0,JOBDONE
    50 |           TERMINATE
```

Figure 1 (continued): The GPSS/H Model (part 5 of 7)

```
BLOCK# | LOCATION OPERATION  OPERANDS           COMMENTS
       | *
       | *********************************************************************
       | *    Block Sequences for the Use of Machines 2-5 Are Analogous to   *
       | *    Those for Machine 1 (Blocks 36-50) and so Are Not Shown Here   *
       | *********************************************************************
       | *
       | *********************************************************************
       | *         Wrapup for Finished Jobs                                  *
       | *********************************************************************
       | *
       | *      ...tabulate finished job's time in the system...
  111  | JOBDONE  TABULATE   SYSTYMS
       | *
       | *      ...branch if not tardy;
       | *         else, tabulate finished job's tardy time...
  112  |          TEST G     AC1,PL(DUEDATE),NOTLATE
  113  |          TABULATE   TARDTYMS
       | *
       | *      ...count down on finished jobs leaving the system...
  114  | NOTLATE  TERMINATE  1
```

Figure 1 (continued): The GPSS/H Model (part 6 of 7)

```
              -------------------------------------------------
               Performance Report for FRS Dispatching Rule
              Under Conditions of Perfect Sequencing Flexibility
              -------------------------------------------------
                      (FRS:  Fewest Remaining Steps)

                      Number of Initialization Jobs: 1000

                      All Report Entries Are Cumulative
               (Subsequent to Eliminating Initialization Statistics)

                                                   TIME IN SYSTEM,HRS
 No. of      TARDY JOBS      TARDY TIME, HRS          (All Jobs)
Jobs Done   Total   Pct      Avg.   Std. Dev.      Avg.   Std. Dev.
---------   -----   ---      ----   ---------      ----   ---------
   100        30   30.0       3.7      2.6          8.9      6.3
   300        84   28.0       4.2      3.1          8.5      6.9
   500        91   18.2       3.9      3.2          6.5      6.1
   700        97   13.9       3.7      3.2          5.7      5.5
   900       102   11.3       3.6      3.2          5.2      5.0
  1100       107    9.7       3.4      3.2          4.9      4.7
  1300       116    8.9       3.2      3.1          4.8      4.5
  1500       131    8.7       3.0      3.0          4.8      4.4
  1700       143    8.4       2.8      2.9          4.7      4.2
  1900       148    7.8       2.8      2.9          4.7      4.2
  2100       155    7.4       2.8      2.9          4.6      4.1
  2300       194    8.4       2.7      2.7          4.7      4.2
  2500       236    9.4       2.6      2.7          4.9      4.3
-------------------------------------------------------------------
```

Figure 2: The Report Produced when the GPSS/H Model of Figure 1 is Executed (every other row has been deleted from this report because of space restrictions)

```
BLOCK#  LOCATION OPERATION  OPERANDS              COMMENTS
        *
        ***********************************************************************
        *         Run-Control and Customized Reporting Statements             *
        ***********************************************************************
        *
        ********************* Start of Report Header *************************
        *
                  PUTPIC     LINES=15,FILE=SYSPRINT,(&INITJOBS)
                ---------------------------------------------------
                   Performance Report for FRS Dispatching Rule
                Under Conditions of Perfect Sequencing Flexibility
                ---------------------------------------------------
                            (FRS:  Fewest Remaining Steps)

                       Number of Initialization Jobs: ****

                        All Report Entries Are Cumulative
               (Subsequent to Eliminating Initialization Statistics)

                                                           TIME IN SYSTEM,HRS
           No. of     TARDY JOBS       TARDY TIME, HRS        (All Jobs)
          Jobs Done   Total   Pct      Avg.   Std. Dev.     Avg.  Std. Dev.
          ---------   -----   ---      ----   ---------     ----  ---------
        ********************** End of Report Header *************************
        *
        *      ...process initialization jobs,
        *         then flush the initialization statistics...
        *
                  START     &INITJOBS,NP
                  RESET
        *
        *      ...loop through 25 sets of jobs, 100 jobs per set...
        *
                  DO         &I=1,25
        *
                     START   100,NP
        *
        *      ...for each job set, write out cumulative statistics...
        *
                  PUTPIC     LINES=1,FILE=SYSPRINT,_
              (100*&I,TC(TARDTYMS),FLT(TC(TARDTYMS))/&I,_
           TB(TARDTYMS),TD(TARDTYMS),TB(SYSTYMS),TD(SYSTYMS))
           ****        ****   **.*      **.*     **.*        **.*    **.*
        *
                  ENDDO
        *
        *      ...line out the end of the report...
                  PUTPIC     LINES=1,FILE=SYSPRINT
          ------------------------------------------------------------------

        *      ...that's all, folks...
                  END
```

Figure 1 (concluded): The GPSS/H Model (part 7 of 7)