Division of Research
School of Business Administration
The University of Michigan

December 1991

AN ALTERNATIVE APPROACH TO MODELING PERFECT SEQUENCING FLEXIBILITY

Working Paper #669

Thomas J. Schriber
The University of Michigan

ABSTRACT

The concept of sequencing flexibility and its potential importance in such scheduling environments as those of manufacturing systems are reviewed briefly. The description of a manufacturing system 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). An approach to modeling perfect sequencing flexibility is then described and a GPSS/H model illustrating the implementation of this approach in the setting of the given manufacturing system is presented.

This paper and model are modified versions of a similar paper and model by Schriber (1991b). The present paper illustrates a modeling approach in which watchdogs are used to dispatch jobs to machines. In the approach taken in the 1991b paper, watchdogs are not employed; instead, jobs themselves are responsible for dispatching other jobs to machines. No matter how the dispatching is accomplished, the freeing up of a machine brings about a *two*fold dispatching responsibility when there is sequencing flexibility: another job can be dispatched to that machine; and some other job can be dispatched to some other machine as well, if conditions permit. (Particulars are described here on page 4 under point 5.) The 1991b model does not provide for the second of these two dispatching activities, whereas the model presented in this paper does.

1. Sequencing Flexibility

Suppose that manufacturing a particular product requires that one step be performed on the corresponding unit of work-in-process (WIP) 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 the 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 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 a 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) additionally 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 building and 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. An Exercise for Modeling Perfect Sequencing Flexibility

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 chronological 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 (with the system residence time measure based on all jobs, whether they are tardy or not). Assume that 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. (For an example of the requested simulation report, see Figure 2 on page 19 here.)

3. An Approach to Building the Model

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

1. When a job arrives, it needs to record its time of arrival (to support the later measurement of its system residence time) and then sample to determine how many steps are to be performed on it, which machines are to be involved, and what the individual 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 conveniently 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 job only involves one step, then the original creates no clones and the original is the one and only "sub-job" in this case.) Each clone will then be a one-step sub-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 and the values of the corresponding attribute of other 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 be accomplished:

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 temporarily their candidacy for machine use. (Remember that only one step at a time can be performed on an job.)

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 for the overall job" attribute on all associated sub-jobs.

The a, b and c messaging can be supported by having the sub-jobs making up an overall 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* as well, to support the messaging needs described above. If operational sets of this type are not provided in the modeling language used to implement the simulation, then the modeler will have to work with other language elements in an attempt 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 simply 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 and depart the model.

5. When a sub-job finishes using its machine, and apart from the needs described in step 4, there is a need to dispatch another sub-job to the now-idle machine (assuming that there is at least one qualified sub-job waiting for that machine; note that such a sub-job would be associated with *another* overall job, not with the overall job for which a sub-job has just finished in use of the machine). There is *also* a need to check whether another associated sub-job can now capture the machine for which it is waiting. Why? Because an associated sub-job could be waiting for a machine that is *idle*, with the need to wait dictated not by a lack of machine availability, but dictated instead by the fact that a kindred sub-job was in the process of being machined.

When a sub-job finishes using a machine, the sub-job itself can assume the twofold responsibility described above; alternatively, a watchdog not connected with any jobs at all can be designated for each machine and take the responsibility of dispatching a job to its machine whenever conditions permit. These responsibilities are left to the sub-job itself in Schriber 1991b, but are handled by watchdogs in the present paper, one watchdog per machine. At the philosophical level, it can be argued that the watchdog approach may be the better approach. Why? Because it is conceptually cleaner and more modular to explicitly identify the job-dispatching responsibility and isolate it in its own model section than to embed it piecewise within the model sections whose purpose is to provide the logic of using a machine after a job has been dispatched to it.

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. *Note that 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 sub-jobs waiting for the machine may change while the sub-jobs wait for the machine's current user to finish. This means 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 that need be taken into account in modeling *perfect* sequencing flexibility for the *fewest-steps-remaining* dispatching rule. The logical considerations 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 the messaging requirement described in 3b does not arise.) The logical considerations may be more demanding, however, in cases of *partial* sequencing flexibility, irrespective of the dispatching rule involved. For the extreme of *no* sequencing flexibility, relatively minimal logical considerations are involved.

4. Implementation of the Approach in GPSS/H

The logical considerations outlined in section 3 are implemented in the GPSS/H (Henriksen and Crain 1989) model displayed in Figure 1 (pages 9 through 18), and output from the simulation requested in section 2 is given in Figure 2 (page 19). In addition to showing the model itself, Figure 1 provides an appended column of block numbers (labeled BLOCK#) at the left, and of column labels (LOCATION, OPERATION, etc.) across the top of each part of the figure.

Only brief discussion of the GPSS/H model is given here, but the comments embedded liberally in the model itself (the Figure 1 comments are in lower case for the most part 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. Briefly, parts 1, 2 and 3 of Figure 1 (pages 10, 11 and 12) specify the background against which the model is set. 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 4 and 5 of Figure 1 (pages 12 and 13). Blocks 8 and 9 (ADVANCE and SPLIT) structure the job arrival process, Blocks 13 through 20 determine how many and which machines a job needs, and Blocks 21 through 26 handle step times while Block 27 sets the due date.

2. Block 31 (SPLIT) in part 5 of Figure 1 provides sub-jobs (clones) that inherit the attributes of the original job.

3. Block 33 (JOIN) in part 5 of Figure 1 puts an overall job's sub-jobs into a unique set (a GPSS/H Group) to support the messaging needs described earlier. Block 35 (TRANSFER) then sends each sub-job to the waiting line for its designated machine.

The use of each machine is modeled with a stack of Blocks specific to that machine. Hence, Blocks 36 through 46 in part 6 (page 14) are for machine one; Blocks 47 through 57 in part 7 (page 15) are for machine two; and so on. Because each of these stacks of Blocks reflects identical logic, only the stack specific to machine one will be discussed here.

When a sub-job arrives at its machine, it joins the set of other sub-jobs (if any) waiting for the machine. (Sub-jobs wait on a User Chain. Sub-jobs coming to machine one, for example, are put onto the machine one User Chain via Block 36, LINK.) Note that the sub-job does *not* test the machine's status to determine if it can capture the machine. (In the model presented here, watchdogs have the responsibility of dispatching sub-jobs to machines. This responsibility is not exercised by the sub-jobs themselves. See point 5 in this section for watchdog particulars.)

In the machine-one stack of Blocks, Block 38 (ALTER) handles the need for a sub-job to let all other sub-jobs in its set know that a step is starting on a member of their set. Block 43 (ALTER) accomplishes the reverse effect. Block 42 (ALTER) is used to update the number-of-remaining-steps attribute of set members.

4. A completed sub-job at machine one uses Block 44 (REMOVE) to remove itself from its sub-job set, then uses Block 45 (TEST) to determine if it is survived by other sub-jobs in the set. If so, it leaves the model (Block 46, TERMINATE); otherwise, it transfers to Block 91 (TABULATE, page 16) to record statistics on the now-finished overall job, and then leaves the model.

5. In this model, watchdogs are used to dispatch jobs to machines. Each machine has its own watchdog. Watchdog logic is provided for all the watchdogs with Blocks 95 through 106 in part 9 of Figure 1 (page 17). Each watchdog moves repeatedly over time through this cycle:

a. The watchdog waits at a TEST Block (Block 99) until its machine is idle and at least one job is queued up for that machine.

b. The watchdog then (at one and the same simulated time) executes the following logic. First, it unlinks (Block 100, UNLINK) from their place of waiting (a User Chain) all sub-jobs waiting for its machine and targets them to be relinked in their place of waiting, ranked ascending on the number of remaining steps for the associated overall jobs. At Block 101 (BUFFER), the watchdog pauses while the unlinked sub-jobs are relinked (at their LINK Block), ordered ascending on their remaining-steps attribute. At Block 102, the watchdog then unlinks the most highly qualified waiting sub-job and dispatches it to capture the machine. At Block 103 (BUFFER), the watchdog pauses again to let the dispatched sub-job carry out the machine capture. The watchdog then takes note of the simulated time of its ongoing movement (ASSIGN, Block 104) and waits (TEST, Block 105) until the next clock advance has taken place before it resumes its machine-monitoring role at the aforementioned TEST Block (Block 99).

The preceding logic is subtle, and merits careful study. The following points should be considered. First, the watchdogs have lower priority than the sub-jobs. (This means that at each clock time, arrival of new jobs and/or the freeing up of machines will take place before the watchdogs do their monitoring. It also means that when a watchdog executes a BUFFER Block, the higher-priority sub-jobs it has unlinked from a User Chain will be processed before further processing of the watchdog itself takes place.) Second, each watchdog makes its entire set of moves (if any are called for) before any of the other watchdogs make their moves. (If watchdog movement were intermixed, two or more

watchdogs might try to dispatch associated sub-jobs to the respective watchdog-machines, but this would be invalid because only one machining step at a time can be carried out on an overall job.) Third, it is essential that when a watchdog dispatches a sub-job to a machine, the sub-job be permitted to capture the machine *and message its associated sub-jobs that the overall job is now active* before any other watchdog starts to make its set of moves. (Otherwise, another watchdog might dispatch to its machine a sub-job that shouldn't capture a machine at this time because its overall job is already active.)

6. Part 10 of Figure 1 (page 18) provides for run control and the reporting logic which leads to the simulation report shown in Figure 2 (page 19).

```
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,40000,_    number of bytes in COMMON
                     GRP,4000       number of 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 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 10)

```
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
 TRUE      SYN        1              code for testing Boolean expressions
*
*       ...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 OKTOTRY expression is true if this watchdog's
*          machine is idle and at least one sub-job is
*          waiting for the machine...
 OKTOTRY   BVARIABLE  FNU(PF(MYMAC))AND(CH(PF(MACLINE))>0)
*
*       ...the QUALIFY expression is true if this sub-job's
*          overall job is inactive (idle)...
 QUALIFY   BVARIABLE  PF(STATUS)=IDLE
*
*       ...labels for the various machine SEIZE Blocks...
 GETMAC    FUNCTION   PF(MYMAC),D5
MAC1,GETMAC1/MAC2,GETMAC2/MAC3,GETMAC3/MAC4,GETMAC4/MAC5,GETMAC5
*
*       ...labels for the various waiting-line LINK Blocks...
 INTOLINE FUNCTION    PF(MYMAC),D5
MAC1,LINEFOR1/MAC2,LINEFOR2/MAC3,LINEFOR3/MAC4,LINEFOR4/MAC5,LINEFOR5
*
*       ...identifiers for each machine's User Chain...
 LINEID    FUNCTION   PF(MYMAC),D5
MAC1,MAC1LINE/MAC2,MAC2LINE/MAC3,MAC3LINE/MAC4,MAC4LINE/MAC5,MAC5LINE
*
*       ...identifiers for each watchdog's machine...
 MACID     FUNCTION   PF(SERIALNO),D5
1,MAC1/2,MAC2/3,MAC3/4,MAC4/5,MAC5
```

Figure 1: The GPSS/H Model (part 2 of 10)

```
BLOCK#  LOCATION OPERATION  OPERANDS          COMMENTS
*
*       ...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: The GPSS/H Model (part 3 of 10)

```
BLOCK#  LOCATION OPERATION  OPERANDS          COMMENTS
        *
        ************************************************************************
        *         Job Creation and Specification Segment                       *
        ************************************************************************
        *
        *       ...seed the segment with a master job-Xact (Transaction)...
      1            GENERATE   0,,,1,25,11PF,9PL
        *
        *       ...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
        *
        *       ...job is idle at time of eventual arrival...
      7            ASSIGN     STATUS,IDLE,PF
        *
        *       ...interarrival time elapses...
      8  NEXTJOB   ADVANCE    RVEXPO(1,&JOBIAT)
        *
        *       ...create successor job-Xact;
        *          route it to experience it's interarrival time...
      9            SPLIT      1,NEXTJOB
        *
        *       ...record time of job's arrival at the system...
     10            ASSIGN     TIMEIN,AC1,PL
        *
        *       ...update 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
        *
     14  SHUFFLE   BLET       &INDEX1=RN2@5+1       &INDEX1 = sample from U(1,5)
     15            BLET       &INDEX2=RN3@5+1       &INDEX2 = sample from U(1,5)
     16            BLET       &TEMPSTOR=PF(&INDEX1)     swap the two
     17            ASSIGN     &INDEX1,PF(&INDEX2),PF    randomly-chosen
     18            ASSIGN     &INDEX2,&TEMPSTOR,PF      machine id's
        *
        *       ...update loop count and repeat as needed...
     19            BLET       &LUPCOUNT=&LUPCOUNT-1
     20            TEST E     &LUPCOUNT,0,SHUFFLE
        *
        ********************** End of Permutation Logic ************************
        *
```

Figure 1: The GPSS/H Model (part 4 of 10)

```
BLOCK#  LOCATION OPERATION  OPERANDS          COMMENTS
*       ...set 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 sub-job for each step
*          (create no clones for a one-step job),
*          numbering the sub-jobs 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 sub-job joins a Group unique to this job...
33         JOIN       PF(JOBID)
*
*       ...pause while each other sub-job for this job
*          joins this group...
34         PRIORITY   PR,BUFFER
*
35         TRANSFER   ,FN(INTOLINE)
*
```

Figure 1: The GPSS/H Model (part 5 of 10)

```
BLOCK#  LOCATION OPERATION  OPERANDS          COMMENTS
        *
        ************************************************************************
        *         Use of Machine 1                                             *
        ************************************************************************
        *
        *       ...go into waiting line (onto User Chain) ranked
        *          in order of increasing number of remaining steps...
    36   LINEFOR1 LINK       MAC1LINE,(STEPS2GO)PF
        *
        *       ...capture machine 1...
    37   GETMAC1  SEIZE      MAC1
        *
        *       ...message other sub-jobs making up this overall job
        *          that this job is now becoming active...
    38           ALTER      PF(JOBID),ALL,(STATUS)PF,BUSY
        *
        *       ...use the machine, free it, and update the
        *          remaining number of steps this job requires...
    39           ADVANCE    PL(PF(SERIALNO))
    40           RELEASE    MAC1
    41           ASSIGN     STEPS2GO-,1,PF
        *
        *       ...message updated number of remaining steps to
        *          other sub-jobs making up this overall job...
    42           ALTER      PF(JOBID),ALL,(STEPS2GO)PF,PF(STEPS2GO)
        *
        *       ...message other sub-jobs making up this overall job
        *          that this job has now become inactive...
    43           ALTER      PF(JOBID),ALL,(STATUS)PF,IDLE
        *
        *       ...remove this sub-job from this job-group...
    44           REMOVE     PF(JOBID)
        *
        *       ...if this was the last step for this job, branch to JOBDONE;
        *          else, simply destroy this sub-job...
    45           TEST NE    G(PF(JOBID)),0,JOBDONE
    46           TERMINATE
        *
```

Figure 1: The GPSS/H Model (part 6 of 10)

```
BLOCK#  LOCATION OPERATION  OPERANDS          COMMENTS
        *
        ********************************************************************
        *         Use of Machine 2                                         *
        *         (Comments for Machine 1 Also Apply to Machines 2-5)      *
        ********************************************************************
        *
   47    LINEFOR2 LINK       MAC2LINE,(STEPS2GO)PF
        *
   48    GETMAC2  SEIZE      MAC2
   49             ALTER      PF(JOBID),ALL,(STATUS)PF,BUSY
   50             ADVANCE    PL(PF(SERIALNO))
   51             RELEASE    MAC2
   52             ASSIGN     STEPS2GO-,1,PF
   53             ALTER      PF(JOBID),ALL,(STEPS2GO)PF,PF(STEPS2GO)
   54             ALTER      PF(JOBID),ALL,(STATUS)PF,IDLE
   55             REMOVE     PF(JOBID)
        *
   56             TEST NE    G(PF(JOBID)),0,JOBDONE
   57             TERMINATE
        *
        ********************************************************************
        *         Use of Machine 3                                         *
        *         (Comments for Machine 1 Also Apply to Machines 2-5)      *
        ********************************************************************
        *
   58    LINEFOR3 LINK       MAC3LINE,(STEPS2GO)PF
        *
   59    GETMAC3  SEIZE      MAC3
   60             ALTER      PF(JOBID),ALL,(STATUS)PF,BUSY
   61             ADVANCE    PL(PF(SERIALNO))
   62             RELEASE    MAC3
   63             ASSIGN     STEPS2GO-,1,PF
   64             ALTER      PF(JOBID),ALL,(STEPS2GO)PF,PF(STEPS2GO)
   65             ALTER      PF(JOBID),ALL,(STATUS)PF,IDLE
   66             REMOVE     PF(JOBID)
        *
   67             TEST NE    G(PF(JOBID)),0,JOBDONE
   68             TERMINATE
        *
```

Figure 1: The GPSS/H Model (part 7 of 10)

```
BLOCK#  LOCATION OPERATION  OPERANDS          COMMENTS
        *
        ************************************************************************
        *         Use of Machine 4                                             *
        *         (Comments for Machine 1 Also Apply to Machines 2-5)          *
        ************************************************************************
        *
   69    LINEFOR4 LINK       MAC4LINE,(STEPS2GO)PF
        *
   70    GETMAC4  SEIZE      MAC4
   71             ALTER      PF(JOBID),ALL,(STATUS)PF,BUSY
   72             ADVANCE    PL(PF(SERIALNO))
   73             RELEASE    MAC4
   74             ASSIGN     STEPS2GO-,1,PF
   75             ALTER      PF(JOBID),ALL,(STEPS2GO)PF,PF(STEPS2GO)
   76             ALTER      PF(JOBID),ALL,(STATUS)PF,IDLE
   77             REMOVE     PF(JOBID)
        *
   78             TEST NE    G(PF(JOBID)),0,JOBDONE
   79             TERMINATE
        *
        ************************************************************************
        *         Use of Machine 5                                             *
        *         (Comments for Machine 1 Also Apply to Machines 2-5)          *
        ************************************************************************
        *
   80    LINEFOR5 LINK       MAC5LINE,(STEPS2GO)PF
        *
   81    GETMAC5  SEIZE      MAC5
   82             ALTER      PF(JOBID),ALL,(STATUS)PF,BUSY
   83             ADVANCE    PL(PF(SERIALNO))
   84             RELEASE    MAC5
   85             ASSIGN     STEPS2GO-,1,PF
   86             ALTER      PF(JOBID),ALL,(STEPS2GO)PF,PF(STEPS2GO)
   87             ALTER      PF(JOBID),ALL,(STATUS)PF,IDLE
   88             REMOVE     PF(JOBID)
        *
   89             TEST NE    G(PF(JOBID)),0,JOBDONE
   90             TERMINATE
        *
        ************************************************************************
        *         Wrapup for Finished Jobs                                     *
        ************************************************************************
        *
        *      ...tabulate finished job's time in the system...
   91    JOBDONE  TABULATE   SYSTYMS
        *
        *      ...branch if not tardy;
        *         else, tabulate finished job's tardy time...
   92             TEST G     AC1,PL(DUEDATE),NOTLATE
   93             TABULATE   TARDTYMS tardy job lateness
        *
        *      ...count down on finished jobs leaving the system...
   94    NOTLATE  TERMINATE  1
        *
```

Figure 1: The GPSS/H Model (part 8 of 10)

```
BLOCK#  LOCATION OPERATION  OPERANDS          COMMENTS
        *
        ********************************************************************
        *         Machine Watchdog Transactions                            *
        ********************************************************************
        *
        *       ...create a master watchdog...
   95             GENERATE   0,,,1,1,11PF,9PL
        *
        *       ...bring in more watchdogs (one watchdog per machine)...
   96             SPLIT      4,NEXTONE,(SERIALNO)PF
        *
        *       ...pick up the identifier for this watchdog's machine...
   97   NEXTONE   ASSIGN     MYMAC,FN(MACID),PF
        *
        *       ...pick up the identifier for the machine's User Chain...
   98             ASSIGN     MACLINE,FN(LINEID),PF
        *
        *       ...wait until the machine is idle and at least one
        *          sub-job is waiting for the machine...
   99   AGAIN     TEST E     BV(OKTOTRY),TRUE
        *
        *       ...unlink all sub-jobs from the User Chain
        *          of sub-jobs waiting for this machine...
  100             UNLINK     PF(MACLINE),FN(INTOLINE),ALL
        *
        *       ...pause to put these sub-jobs back onto their User Chain
        *          ranked ascending by the remaining number of steps
        *          in their job-group...
  101             BUFFER
        *
        *       ...now unlink the first qualifying sub-job (if any) from
        *          the front of the User Chain and send it to the machine...
  102             UNLINK     PF(MACLINE),FN(GETMAC),1,BV(QUALIFY)
        *
        *       ...pause to let the unlinked sub-job capture its machine
        *          and update the status of it's job-group...
  103             BUFFER
        *
        *       ...copy current clock time onto this watchdog...
  104             ASSIGN     TYMMOVED,AC1,PL
        *
        *       ...the watchdog now waits until the clock has been advanced...
  105             TEST G     AC1,PL(TYMMOVED)
        *
        *       ...now the watchdog resumes its monitoring role...
  106             TRANSFER   ,AGAIN
        *
```

Figure 1: The GPSS/H Model (part 9 of 10)

```
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: The GPSS/H Model (part 10 of 10)

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)

No. of Jobs Done	TARDY JOBS Total	TARDY JOBS Pct	TARDY TIME, HRS Avg.	TARDY TIME, HRS Std. Dev.	TIME IN SYSTEM, HRS (All Jobs) Avg.	TIME IN SYSTEM, HRS (All Jobs) Std. Dev.
100	33	33.0	3.3	2.7	8.8	6.2
200	70	35.0	4.0	3.1	9.2	7.2
300	84	28.0	4.2	3.1	8.5	6.9
400	87	21.7	4.0	3.1	7.2	6.4
500	90	18.0	3.9	3.1	6.3	6.1
600	91	15.2	3.9	3.1	5.8	5.8
700	94	13.4	3.8	3.1	5.5	5.5
800	96	12.0	3.7	3.1	5.2	5.2
900	99	11.0	3.7	3.1	5.0	5.0
1000	99	9.9	3.7	3.1	4.8	4.8
1100	105	9.5	3.5	3.1	4.7	4.7
1200	110	9.2	3.4	3.0	4.8	4.6
1300	112	8.6	3.3	3.0	4.6	4.5
1400	116	8.3	3.3	3.0	4.6	4.3
1500	117	7.8	3.2	3.0	4.5	4.3
1600	120	7.5	3.2	3.0	4.5	4.2
1700	128	7.5	3.1	3.0	4.4	4.1
1800	129	7.2	3.1	3.0	4.4	4.1
1900	132	6.9	3.1	2.9	4.4	4.0
2000	132	6.6	3.1	2.9	4.3	4.0
2100	138	6.6	3.0	2.9	4.3	3.9
2200	145	6.6	2.9	2.9	4.3	3.9
2300	171	7.4	2.9	2.7	4.5	4.1
2400	197	8.2	2.8	2.7	4.6	4.2
2500	212	8.5	2.8	2.7	4.6	4.2

Figure 2: The Report Produced when the GPSS/H Model of Figure 1 is Executed

5. Summary

The concept of sequencing flexibility in a scheduling environment has been described briefly, and the advantages potentially provided by sequencing flexibility have been commented upon. The need to use simulation modeling to quantify the potential advantages of sequencing flexibility has been indicated. The logical requirements involved in modeling perfect sequencing flexibility have been outlined, and the implementation of these requirements in the GPSS/H modeling language has been shown. Others using GPSS/H to investigate the characteristics of perfect sequencing flexibility can use as a starting point the model presented here, or can apply techniques illustrated in this model to construct similar models for situations of interest to them. And those using other modeling languages to investigate perfect sequencing flexibility can take as guidelines for their work the logical considerations identified and outlined here.

REFERENCES

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

Rachamadugu, R. and T.J. Schriber. 1990a. "Performance of Dispatching Rules Under Perfect Sequencing Flexibility." *Proceedings of the 1990 Winter Simulation Conference.* Society for Computer Simulation, San Diego, CA.

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. 1991a. *An Introduction to Simulation Using GPSS/H,* John Wiley & Sons, Inc., New York, NY.

Schriber, T.J. 1991b. "The Modeling of Perfect Sequencing Flexibility in a Scheduling Environment." *Proceedings of the 1991 Winter Simulation Conference.* Society for Computer Simulation, San Diego, CA.