Research Support
School of Business Administration

Revised May 1994

MODELING PERFECT SEQUENCING FLEXIBILITY WHEN "FEWEST REMAINING STEPS" IS USED AS THE DISPATCHING RULE

Working Paper #9403-16

Thomas J. Schriber
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. The emphasis in this paper is on the model itself. The impact of both perfect and partial sequencing flexibility on measures of manufacturing system performance are reported separately in a literature article for which a reference is given.

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) proposed a sequencing 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 the Rachamadugu and Schriber metric, an SFM value of 0.6 results for this situation. (See the referenced papers for quantitative details. These details are also given in Rachamadugu, Nandkeolyar and Schriber 1993.)

There are potential benefits in 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) and Rachamadugu, Schriber and Nandkeolyar (1993) 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 Rachamadugu and Schriber (1990b) found that 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 pointed 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. An Exercise for Modeling Perfect Sequencing Flexibility

A particular flexible manufacturing system consists of nine 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 9 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 multiplying its total step time by 7.5, then adding this product to its time of arrival.)

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.

The particular machines a job is to visit should be determined by sampling without replacement from the set of nine machines. Whenever a job is dispatched to a machine, first-come first-served should be used to resolve ties for the fewest remaining steps.

Perform a single simulation with the model. Initialize the model by simulating until 2500 jobs have been completed. Then continue the simulation until another 10000 jobs have been completed. For each 500 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 18 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 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. Each clone will then be a one-step sub-job associated with the overall job. When all sub-jobs are finished, 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-remaining-steps 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, with first-come, first-served used to break ties for fewest remaining steps.

3. Before the sub-jobs forming 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, 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 until just now 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 dispatching 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 handled by watchdogs in the approach presented here, one watchdog per machine. It can be argued that the watchdog approach is the better approach, 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 within the model section whose purpose is to provide the logic of using a machine after a job has been dispatched to it.

In either dispatching approach, qualifying sub-jobs waiting for the machine must be ranked on their number-of-remaining-steps (for their overall job) attribute *just before* the decision is made about which sub-job is the next to get the machine, with ties resolved first-come, first-served. (A *qualifying* sub-job is one which is actively contending for the machine, as contrasted with a sub-job in a state of temporary suspension because an associated sub-job is currently being machined.) *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 remaining steps for one or more waiting sub-jobs may change before the machine's current user finishes with the machine. 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-remaining-steps* 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 modest 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 10 through 17), and output from the simulation requested in section 2 is given in Figure 2 (page 18). 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. (See Schriber 1991 for more particulars about the composition of GPSS/H statements.)

Only brief discussion of the GPSS/H model is given here, but the comments embedded liberally in the model itself should make it quite easy (for a person familiar with any discrete-event modeling *language*, and certainly for anyone familiar with GPSS/H itself, or GPSS in general) 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 21, parts 4 and 5 of Figure 1 (pages 13 and 14). Blocks 3 and 4 (ADVANCE and SPLIT, page 13) structure the job arrival process. Block 11 (page 14) determines how many steps need to be carried out on a job. Blocks 12 through 20 (page 14) determine which machines a job needs (and what the corresponding step times are), using a sampling-without-replacement scheme. Finally, Block 21 (page 14) sets the due date.

Note how the logic in Blocks 11 through 20 is flexibly specified in terms of the number of machines (&MACHINES) in the system being modeled. Although the exercise of Section 2 describes the system as consisting of *nine* machines, the model is flexible enough to handle systems consisting of from 2 to 9 machines. (A one-machine system would not be of interest in a study of perfect sequencing flexibility.) The number of machines in the system is set with the "LET &MACHINES=9" control statement in part 2 (page 11) of Figure 1. (If the system consisted of *more* than nine machines, only several changes would be needed in the model to handle this case, too. In particular, no Blocks would have to be changed at all to increase the number of machines in the system.)

Note similarly how the job interarrival time is flexibly specified in terms of overall machine utilization and the number of machines in the system. See the "LET &SYSUTIL=0.90" and the "LET &JOBIAT=..." statements in part 2 (page 11) of Figure 1.

2. Block 24 (SPLIT) in part 5 of Figure 1 (page 14) provides sub-jobs (clones) that inherit the attributes of the original job. Each sub-job is delegated to one of the particular machines that will be used to carry out the overall job. This delegation involves tagging each sub-job with the identifier for its particular machine. (The SPLIT Block's serialization option, expressed through the SPLIT Block's C Operand, is used to serially number an overall job's sub-jobs 1, 2, 3, ..., and then the Block 26 ASSIGN relates sub-job serial number to the particular machine to which the sub-job is delegated.)

3. Block 27 (JOIN) at the bottom of part 5 (page 14) of Figure 1 puts an overall job's sub-jobs into a unique set (a GPSS/H Group) to support the eventual messaging needs previously outlined. Block 28 (part 6 of Figure 1, page 15) computes a unique index for a job such that when the various sub-jobs waiting for a machine are ranked by that index in ascending order, the rank corresponds to first-come, first-served within fewest-remaining-steps.

The use of all machines is modeled with a single stack of Blocks (Blocks 30 through 39 in part 6 of Figure 1, page 15). 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 29, LINK. The Block 29 LINK addresses User Chains at one level of indirection and so is flexible enough to accommodate each machine's User Chain.) Note that sub-jobs do *not* test the status of their machine to determine if they 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-usage stack of Blocks, Block 31 (ALTER, page 15) 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 38 (ALTER) accomplishes the reverse effect. Block 37 (ALTER) is used to update the number-of-remaining-steps attribute of set members.

4. A completed sub-job uses Block 36 (TEST) to determine if more steps remain before its overall job will be finished. If more steps remain, it accomplishes the messaging described in Section 3 under points 3b and 3c (using Blocks 37 and 38, ALTER, for this purpose) and leaves the model (Block 39, TERMINATE); otherwise, it simply transfers to the JOBDONE Block (Block 40, TABULATE, part 7 of Figure 1, 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 44 through 53 in part 7 of Figure 1 (page 16). Each watchdog moves repeatedly over time through this cycle:

a. The watchdog waits at a TEST Block (Block 46) until its machine is idle and at least one job is queued up for that machine.

b. The watchdog then executes the following logic (at one and the same simulated time). First, it unlinks (Block 47, UNLINK) from their place of waiting (a User Chain) all *qualifying* sub-jobs waiting for its machine and targets them to be re-linked in their place of waiting, ranked ascending on the number of remaining steps for the associated overall jobs, with first-come, first-served used to break ties. (A sub-job qualifies for this unlink/relink process if its overall job is inactive, which means the sub-job is a candidate for capturing its machine now.) If there are *no* qualifying sub-jobs, then the watchdog's machine must remain idle for the time being and so the watchdog skips to Block 51 (see below). But if there is at least one qualifying sub-job, the watchdog pauses (Block 48, BUFFER) while the one or more qualifying sub-jobs have their ranking criterion updated (Block 28, ASSIGN) and then are relinked (at Block 29, LINK) in their updated order. At Block 49, the watchdog then unlinks the most highly qualified waiting sub-job and dispatches it to capture the machine. At Block 50 (BUFFER), the watchdog pauses again to let the dispatched sub-job capture the machine and message associated sub-jobs that the overall job is now active. The watchdog then takes note of the simulated time of its ongoing movement (ASSIGN, Block 51) and waits (TEST, Block 52) until the next clock advance has taken place before it resumes its machine-monitoring role at the Block 46 TEST.

The logic under point 5 above 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 a new job and/or the freeing up of one or more 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.) It is also important to note the need for the Block 52 TEST; in its absence, a watchdog would loop endlessly if the Block 46 TEST were true, that is, if the watchdog's machine were idle and all sub-jobs waiting for the machine were inactive.

Part 8 of Figure 1 (page 17) provides for run control and the reporting logic which leads to the simulation report shown in Figure 2 (page 18).

```
BLOCK#  LOCATION OPERATION  OPERANDS              COMMENTS
*********************************************************************
*                                                                   *
          SIMULATE                                Base Time Unit: 1 Minute *
*    Sequencing Flexibility Measure:  1.0                           *
*    Service Order: Fewest Remaining Steps                          *
*    Arrival Process:  Poisson                                      *
*    Number of Machines in the System:  &MACHINES (9 maximum)       *
*    Number of Machines Used per Job:  From 1 to &MACHINES (random) *
*    Mean Machining Time:  &MSTEPTYM Minutes, 2-Erlang Distributed  *
*    Expected Machine Utilization:  &ESYSUTIL                       *
*           (see the "LET Statements" below for                     *
*            the setting of model-parameter values)                 *
*                                                                   *
*********************************************************************
*         Compiler Directives                                       *
*********************************************************************
*
*       ...reallocate default maximum quantities of entities...
          REALLOCATE COM,75000,_    number of bytes in COMMON
                     GRP,15000      number of Xact Groups
*
*       ...selected correspondences between
*          symbolic and numeric identifiers...
*
*       ...Facilities and User Chains (machines and their lines)...
 MAC1     EQU        1,F,C    MAC1 is Facility 1 and User Chain 1
 MAC2     EQU        2,F,C             ...and so on...
 MAC3     EQU        3,F,C
 MAC4     EQU        4,F,C
 MAC5     EQU        5,F,C
 MAC6     EQU        6,F,C
 MAC7     EQU        7,F,C
 MAC8     EQU        8,F,C
 MAC9     EQU        9,F,C
*
*       ...Fullword Integer Parameters (variables local to Xacts)...
 MACID1   EQU        1,PF   MACID1 is (in) Fullword Parameter 1
 MACID2   EQU        2,PF          ...and so on...
 MACID3   EQU        3,PF
 MACID4   EQU        4,PF
 MACID5   EQU        5,PF
 MACID6   EQU        6,PF
 MACID7   EQU        7,PF
 MACID8   EQU        8,PF
 MACID9   EQU        9,PF
*
*       ...Real Parameters (variables local to Xacts)...
 STEPTYM1 EQU        1,PL  STEPTYM1 is (in) Real Parameter 1
 STEPTYM2 EQU        2,PL      ...and so on...
 STEPTYM3 EQU        3,PL
 STEPTYM4 EQU        4,PL
 STEPTYM5 EQU        5,PL
 STEPTYM6 EQU        6,PL
 STEPTYM7 EQU        7,PL
 STEPTYM8 EQU        8,PL
 STEPTYM9 EQU        9,PL
*
```

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

```
BLOCK#  LOCATION OPERATION  OPERANDS            COMMENTS
*       ...Integer Variables (global variables)...
           INTEGER    &CHOICE         points to random machine choice
           INTEGER    &I              DO-loop counter
           INTEGER    &INITJOBS       number of initialization jobs
           INTEGER    &JOBCOUNT       jobs-arrived counter
           INTEGER    &LISTSIZE       used in machine-selection scheme
           INTEGER    &MACHINES       number of machines in system
           INTEGER    &MACLIST(9)     list of identifiers for
*                                     up to 9 machines in the system
           INTEGER    &STEPS          number of steps to do this job
*
*       ...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
*
*************************************************************************
*         Control Statements                                            *
*************************************************************************
*
************************** Boolean Expressions ***********************
*
*       ...the OKTOTRY expression is true if the active watchdog's
*          machine is idle and at least one sub-job is waiting
*          for the machine...
 OKTOTRY   BVARIABLE  FNU(PF(MYMAC))AND(CH(PF(MYMAC))>0)
*
*************************** LET Statements ***************************
*
*       ...assignment of values to selected global variables...
           LET        &ESYSUTIL=0.90    expected machine utilization
           LET        &FLOFAKTR=7.5     flow allowance factor
           LET        &INITJOBS=2500    number of initialization jobs
           LET        &MACHINES=9       9 machines
           LET        &MSTEPTYM=30.0    mean step time, minutes
*
           LET        &JOBIAT=_         job interarrival time, minutes
               (FLT(&MACHINES+1)/2)*&MSTEPTYM/(&MACHINES*&ESYSUTIL)
*
```

Figure 1: The GPSS/H Model (part 2 of 8)

```
BLOCK#  LOCATION OPERATION  OPERANDS              COMMENTS
        ************************* RMULT Statements ****************************
        *
        *       ...Initial Positions of U(0,1) random-number generators...
        *          (antithetic random numbers result if these RMULT Operands
        *           are prefixed with a minus sign; see Schriber 1991a)
                   RMULT      100000,_   RN1 (random interarrival times)
                              200000,_   RN2 (random number of job steps)
                              300000,_   RN3 (random pointer into mac-id list)
                              400000,_   RN4 (first piece of 2-Erlang)
                              500000     RN5 (second piece of 2-Erlang)
        *
        ************************* TABLE Statements ****************************
        *
        *       ...system residence time, all jobs, hours...
         SYSTYMS  TABLE       (AC1-PL(TIMEIN))/60.0,0,1,2
        *
        *       ...tardy time for tardy jobs, hours...
         TARDTYMS TABLE       (AC1-PL(DUEDATE))/60.0,0,1,2
        *
        ******************************* UNLIST ********************************
        *
        *       ...suppression of Control-Statement echoes...
                   UNLIST     CSECHO
```

Figure 1: The GPSS/H Model (part 3 of 8)

```
BLOCK#  LOCATION OPERATION  OPERANDS              COMMENTS
        *
        ************************************************************************
        *         Job Creation and Specification Segment                        *
        ************************************************************************
        *
        *       ...seed the segment with a master job-Xact (Transaction)...
      1            GENERATE   0,,,1,25,17PF,13PL
        *
        *       ...job is idle at time of its arrival...
      2            ASSIGN     STATUS,IDLE,PF
        *
        *       ...interarrival time elapses...
      3  NEXTJOB   ADVANCE    RVEXPO(1,&JOBIAT)
        *
        *       ...create successor job-Xact;
        *          route it to experience its interarrival time...
      4            SPLIT      1,NEXTJOB
        *
        *       ...record time of job's arrival at the system...
      5            ASSIGN     TIMEIN,AC1,PL
        *
        *       ...update job count, then give this job a unique id number
      6            BLET       &JOBCOUNT=&JOBCOUNT+1
      7            ASSIGN     JOBID,&JOBCOUNT,PF
        *
        *       ...initialize the list of machine id's
        *          prior to selecting machine id's at random...
        *          (a count-down loop is used)
      8            ASSIGN     POINTER,&MACHINES,PF
      9  NEXTID    BLET       &MACLIST(PF(POINTER))=PF(POINTER)
     10            LOOP       (POINTER)PF,NEXTID
        *
```

Figure 1: The GPSS/H Model (part 4 of 8)

```
BLOCK#  LOCATION OPERATION  OPERANDS              COMMENTS
        *       ...set the number of steps required by this job...
     11            BLET       &STEPS=RN2@&MACHINES+1
        *
        ********** Pick Machine id's and Step Times for This Job ************
        *        (the machine id's are stored in PF(1)...PF(&STEPS), and
        *         corresponding step times are stored in PL(1)...PL(&STEPS);
        *         a count-down loop is used, so the storing starts with
        *         PF(&STEPS) and PL(&STEPS) and ends with PF(1) and PL(1))
        *
        *       ...set the starting length of the machine-id list
        *          and the step whose specs (machine id and step time)
        *          are to be determined first...
     12            BLET       &LISTSIZE=&MACHINES
     13            ASSIGN     STEPNOW,&STEPS,PF
        *
        *       ...choose a random entry in the (remaining) machine-id list...
     14  REPEAT    BLET       &CHOICE=RN3@&LISTSIZE+1
        *
        *       ...assign that machine id for the current step,
        *          and sample and assign a corresponding step time...
     15            ASSIGN     PF(STEPNOW),&MACLIST(&CHOICE),PF
     16            ASSIGN     PF(STEPNOW),_
     16                       RVEXPO(4,&MSTEPTYM/2)+RVEXPO(5,&MSTEPTYM/2),PL
        *
        *       ...update the total step time for this job...
     17            ASSIGN     TOTSTIME+,PL(PF(STEPNOW)),PL
        *
        *       ...copy the last machine id in the current machine-id
        *          list over the machine id just chosen...
     18            BLET       &MACLIST(&CHOICE)=&MACLIST(&LISTSIZE)
        *
        *       ...shorten the size of the remaining machine-id list by 1...
     19            BLET       &LISTSIZE=&LISTSIZE-1
        *
        *       ...repeat for each of this job's steps...
     20            LOOP       (STEPNOW)PF,REPEAT
        *
        ******** End of Logic for Picking Machine id's and Step Times *******
        *
        *       ...assign this job's due date...
     21  TEST      ASSIGN     DUEDATE,AC1+&FLOFAKTR*PL(TOTSTIME),PL
        *
        *       ...initialize the number of steps to go for this job...
     22            ASSIGN     STEPS2GO,&STEPS,PF
        *
        *       ...create clones so there is one sub-job for each step,
        *          numbering the clones (sub-jobs) serially
        *          in a Fullword Parameter...
        *
     23            ASSIGN     SERIALNO,-1,PF
        *
     24            SPLIT      PF(STEPS2GO),JUMP,(SERIALNO)PF
     25            TERMINATE  0
        *
     26  JUMP      ASSIGN     MYMAC,PF(PF(SERIALNO)),PF
        *
        *       ...each sub-job joins a Group unique to this job...
     27            JOIN       PF(JOBID)
```

Figure 1: The GPSS/H Model (part 5 of 8)

```
BLOCK#  LOCATION OPERATION  OPERANDS              COMMENTS
        *
        ********************************************************************
        *         Use of Machines                                          *
        ********************************************************************
        *
        *       ...assign criterion for User-Chain ordering...
        *          (objective:  form a single criterion to rank waiting
        *           jobs by JOBID within STEPS2GO, thereby using FCFS
        *           to resolve STEPS2GO ties)
    28   INTOLINE ASSIGN     RANKINDX,1000000*PF(STEPS2GO)+PF(JOBID),PF
        *
        *       ...go into appropriate waiting line (onto User Chain)
        *          ranked in order of increasing ranking index...
    29             LINK       PF(MYMAC),(RANKINDX)PF
        *
        *       ...capture the appropriate machine...
        *          (note: the following Block is reached by routing
        *           to it Xacts that have been UNLINKed to be sent
        *           to capture their machine)
    30   GETMAC    SEIZE      PF(MYMAC)
        *
        *       ...message other sub-jobs in this job-group
        *          that this overall job is now becoming active...
    31             ALTER      PF(JOBID),ALL,(STATUS)PF,BUSY
        *
        *       ...use the machine, free it, remove this sub-job from
        *          its job-group, and update the remaining number of steps
        *          this overall job requires...
    32             ADVANCE    PL(PF(SERIALNO))
    33             RELEASE    PF(MYMAC)
    34             REMOVE     PF(JOBID)
    35             ASSIGN     STEPS2GO-,1,PF
        *
        *       ...if the overall job is not done, do the needed messaging
        *          and then destroy this sub-job;
        *          else, branch to JOBDONE...
    36             TEST NE    PF(STEPS2GO),0,JOBDONE
        *
        *       ...message updated number of remaining steps to
        *          other sub-jobs in this job-group...
    37             ALTER      PF(JOBID),ALL,(STEPS2GO)PF,PF(STEPS2GO)
        *
        *       ...message other sub-jobs in this job-group
        *          that this overall job has now become inactive...
    38             ALTER      PF(JOBID),ALL,(STATUS)PF,IDLE
        *
        *       ...destroy this sub-job
    39             TERMINATE  0
```

Figure 1: The GPSS/H Model (part 6 of 8)

```
BLOCK#  LOCATION OPERATION  OPERANDS              COMMENTS
        *
        ************************************************************************
        *         Wrapup for Finished Jobs                                      *
        ************************************************************************
        *
        *       ...tabulate finished job's time in the system...
    40   JOBDONE  TABULATE   SYSTYMS
        *
        *       ...branch if not tardy;
        *          else, tabulate finished job's tardy time...
    41             TEST G     AC1,PL(DUEDATE),NOTLATE
    42             TABULATE   TARDTYMS
        *
        *       ...count down on finished jobs leaving the system...
    43   NOTLATE  TERMINATE  1
        *
        ************************************************************************
        *         Machine Watchdog Transactions                                 *
        ************************************************************************
        *
        *       ...create a low-priority master watchdog...
    44             GENERATE   0,,,1,-25,17PF,13PL
        *
        *       ...bring in more watchdogs (one watchdog per machine)...
        *          with serialization used to assign to each watchdog
        *          the identifier for its machine...
    45             SPLIT      &MACHINES-1,AGAIN,(MYMAC)PF
        *
        *       ...wait until the machine is idle and at least one
        *          sub-job is waiting for the machine...
    46   AGAIN     TEST E     BV(OKTOTRY),TRUE
        *
        *       ...unlink all qualifying sub-jobs (if any) from the
        *          User Chain of sub-jobs waiting for this machine;
        *          skip to NONE if none qualify...
    47             UNLINK     PF(MYMAC),INTOLINE,ALL,(STATUS)PF,IDLE,NONE
        *
        *       ...pause to put these sub-jobs back onto their User Chain
        *          ranked ascending by their updated ranking index...
    48             BUFFER
        *
        *       ...now unlink the first qualifying sub-job (if any) from
        *          the front of the User Chain and send it to the machine...
    49             UNLINK     PF(MYMAC),GETMAC,1,(STATUS)PF,IDLE
        *
        *       ...pause to let the unlinked sub-job capture its machine
        *          and update the status of its job-group...
    50             BUFFER
        *
        *       ...copy current clock time onto this watchdog...
    51   NONE      ASSIGN     TYMMOVED,AC1,PL
        *
        *       ...the watchdog now waits until the next clock advance...
    52             TEST G     AC1,PL(TYMMOVED)
        *
        *       ...now the watchdog resumes its monitoring role...
    53             TRANSFER   ,AGAIN
        *
```

Figure 1: The GPSS/H Model (part 7 of 8)

```
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 20 sets of jobs, 500 jobs per set...
        *
                  DO        &I=1,20
        *
                     START   500,NP
        *
        *       ...for each job set, write out cumulative statistics...
        *
                  PUTPIC     LINES=1,FILE=SYSPRINT,_
              (500*&I,TC(TARDTYMS),100.*TC(TARDTYMS)/(500*&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 8 of 8)

Performance Report for FRS Dispatching Rule
Under Conditions of Perfect Sequencing Flexibility

(FRS: Fewest Remaining Steps)

Number of Initialization Jobs: 2500

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

No. of Jobs Done	TARDY JOBS Total	Pct	TARDY TIME, HRS Avg.	Std. Dev.	TIME IN SYSTEM, HRS (All Jobs) Avg.	Std. Dev.
500	24	4.8	3.6	2.8	5.1	3.2
1000	80	8.0	5.3	4.7	7.1	5.9
1500	96	6.4	4.9	4.5	6.7	5.4
2000	109	5.4	4.5	4.4	6.5	5.2
2500	123	4.9	4.1	4.3	6.2	4.9
3000	143	4.8	4.0	4.1	6.1	4.7
3500	193	5.5	4.2	4.1	6.4	5.1
4000	239	6.0	4.1	3.9	6.8	5.3
4500	334	7.4	4.9	4.4	7.4	6.0
5000	426	8.5	5.2	4.6	7.7	6.3
5500	477	8.7	5.1	4.5	7.8	6.3
6000	492	8.2	5.0	4.5	7.6	6.2
6500	514	7.9	4.9	4.4	7.5	6.1
7000	535	7.6	4.8	4.4	7.4	6.0
7500	602	8.0	5.0	4.4	7.6	6.2
8000	654	8.2	5.1	4.5	7.6	6.2
8500	674	7.9	5.0	4.5	7.4	6.1
9000	685	7.6	5.0	4.5	7.3	6.0
9500	697	7.3	5.0	4.4	7.2	5.9
10000	711	7.1	4.9	4.4	7.1	5.9

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

5. Results Produced by an Extended Experimental Study of Sequencing Flexibility

Figure 2 above is limited to sample results produced by the GPSS/H model documented in this paper for the case of perfect sequencing flexibility (SFM = 1.0) and the use of fewest remaining steps as the dispatching rule (with a flow allowance factor of 7.5). Results for full factorial designs covering a range of six SFM values (ranging from 0.0 to 1.0 in steps of 0.2) and eleven scheduling rules (with five alternative flow allowance factors) are reported in Rachamadugu, Nandkeolyar and Schriber (1993).

6. 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.

Rachamadugu, R., U. Nandkeolyar and T. J. Schriber. 1993. "Scheduling with Sequencing Flexibility." *Decision Sciences*, Vol. 24, No. 2, pp. 315-341.

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