Proceedings of the 1986 Winter Simulation Conference
J. Wilson, J. Henriksen, S. Roberts (eds.)

SIMPLIFIED APPROACHES TO MODELING ACCUMULATING AND NONACCUMULATING CONVEYOR SYSTEMS

James O. Henriksen
Wolverine Software Corporation
7630 Little River Turnpike
Annandale, VA 22003, U.S.A.

Thomas J. Schriber
Graduate School of Business
The University of Michigan
Ann Arbor, MI 48109, U.S.A.

ABSTRACT

In many systems (e.g., manufacturing systems), transportation of work-in-process is accomplished by the use of conveyors. The modeling of objects as they move along conveyors is logically demanding, especially if an object-by-object, inch-by-inch approach is taken. Two alternative and substantially simplified modeling approaches for representing movement of objects on conveyors are presented in this paper: the "minimum travel time" approach is appropriate for accumulating conveyors, and the "follow-the-leader" approach is of use in modeling nonaccumulating conveyors. These modeling approaches are presented in language-independent fashion. The application of the approaches is illustrated through a series of four hypothetical conveyor systems and several variations on them. Models for these conveyor systems are built, implemented in GPSS/H, displayed, and discussed. In a series of exercises, the reader is challenged to build and implement models for these same conveyor systems in whatever alternative simulation language(s) may be of interest.

1. OVERVIEW OF THE PAPER

In Section 2 of the paper, characteristics of conveyor systems are described, to provide a basis for understanding certain terms which are subsequently used in the paper. The difficulties of representing the movement of objects along a conveyor are discussed in Section 3. The deficiencies of a commonly used representation are discussed, and two alternative techniques for simplifying this representation are introduced. In Sections 4, 5, 6, and 7, a series of conveyor system problems is stated, and models using the simplified representations of Section 3 are presented and explained.. A collection of conveyor exercises is offered in Section 8 as a basis for reinforcing and extending understanding of the approaches which have been described and illustrated. Conclusions and references are then given.

2. SOME CHARACTERISTICS OF CONVEYOR SYSTEMS

Think of a conveyor as consisting of a transport medium onto which objects are placed at one or more points, and from which objects are removed at one or more points. For purposes of this paper, we distinguish among the following characteristics of conveyors:

(1) Number of entry and exit points

In general, objects may be placed on a conveyor at an arbitrary number of entry points and removed from the conveyor at an arbitrary number of exit points. In this paper, our discussion is limited to single-entry, single-exit conveyor systems

(2) Positions of entry and exit points

In general, the positions of entry and exit points may not stationary. For example, if objects are placed onto and/or removed from a conveyor by human workerss, it is highly unlikely that the workers will assume strictly stationary positions along the conveyor. In this paper, our discussion is limited to conveyors with stationary entry and exit points. Furthermore, we assume that entry and exit points are always at the extreme, opposite ends of the conveyor.

(3) Number of simultaneous entries and removals

In general, one or more objects at a time can be placed onto a conveyor at an entry point, and removed from the conveyor at an exit point. An entry and a removal may or may not be able to occur simultaneously. The models here assume there can be only one entry at a time per entry point, and only one removal at a time per exit point, but that an entry and a removal can occur simultaneously.

(4) Continuous motion vs. interrupted motion

The belt on some conveyors moves continuously. On other conveyors, the belt <u>may</u> have to be stopped when one or more of these conditions occurs:

(a) Objects are being placed onto and/or removed from the conveyor.

(b) An object bumps up against a stop at the conveyor exit. (Continuous belt motion would then cause the belt to slide under one or more objects on the belt, which might not be allowed in some systems.)

(c) A conveyor breakdown occurs.

The models presented in this paper include cases of both continuous and interrupted motion, but do not include conveyor breakdowns.

(5) Single conveyors vs. multiple-conveyor systems

A multiple-conveyor system consists of two or more conveyors in series, and/or may involve junctions, with two or more single conveyors feeding a junction or emanating from a junction. Only single-conveyor systems are discussed in this paper.

(6) Segmented vs. nonsegmented conveyors

A segmented conveyor is divided into a series of physical segments. No overlapping of objects between consecutive segments is permitted. (An overhead conveyor taking the form of a moving chain, with objects placeable only on hooks which are spaced at intervals along the chain, is an example of a discretely-segmented conveyor.) In most systems, there is no such partitioning of the conveying medium into physical segments. Only models of nonsegmented conveyors are presented here.

(7) Homogeneous vs. heterogeneous objects

Some conveyors are used to transport identical (homogeneous) objects, whereas others transport nonidentical (heterogeneous) objects. We assume that objects are homogeneous in the models presented in this paper. The assumption that objects are homogeneous allows the use of an attribute-free representation of objects; i.e., properties of individual objects do not have to be modeled.

(8) Unit width vs. multiple-width conveyors

Objects cannot be placed side-by-side on a unit-width conveyor, whereas two or more objects can be placed side-by-side in multiple-width systems. We assume the unit-width case in the models presented here.

(9) Removal order

Objects reaching the exit from a conveyor can be removed in first-on, first-off order, or they can (in general) be removed in random order. We assume first-on, first-off order in this paper.

(10) Accumulating vs. nonaccumulating conveyors

When an object reaches the exit of an accumulating conveyor and stops moving while waiting to be removed, the belt nevertheless continues to move, sliding under the stationary object at the exit. In general, the continued belt motion then eventually brings the following object up against the first object, the third object against the second object, and so on. Hence, objects can accumulate at the exit, one against the other, with no intervening gaps.

In contrast, when an object reaches the exit of a nonaccumulating conveyor, the belt must be stopped at least until the object can be removed from the conveyor. This means that when once an object is placed on the conveyor, its position relative to other objects on the conveyor is fixed.

3. THE CHALLENGE IN CONVEYOR MODELING

Modeling objects as they move from point to point on a conveyor offers a number of interesting logical challenges. Frequently, representing the current whereabouts of each object on the conveyor is accomplished by partitioning physical space into small increments, e.g., inches, and modeling motion on an inch-by-inch basis. Such an approach may result in excessive model complexity and excessive consumption of CPU time. By taking approaches more circumspect than those which require tracking information on an object-by-object, inch-by-inch basis, models can be simplified, and CPU time requirements can be reduced. Representative comparisons and contrasts are drawn at a finer level of detail in the following subsections.

3.1 Accumulating Conveyors

Consider an accumulating conveyor with one entry and one exit. When an object is placed on the conveyor, the object moves continuously forward until it either reaches the exit or, before reaching the exit, butts up against the object in front of it (that is, reaches the tail of the queue of one or more abutting objects positioned between it and the exit). How is the current position of such an object to be tracked as simulated time elapses? Ideally, one would like to compute the simulated time that would be required for an object to reach either (1) the conveyor exit or (2) the end of the queue of objects. If such a time could be computed, then motion could be modeled as a simple time delay. Unfortunately, the mechanism by which objects queue up in front of conveyor exits is too complex to allow modeling motion as a simple time delay. For example, in general, the end-of-queue is a moving target. If objects are not removed from the conveyor rapidly enough, the end-of-queue position will advance upstream as additional objects join the queue. If the end-of-queue advances upstream, then a previously calculated time required to reach the end-of-queue must be shortened.

As a consequence of these difficulties, a modeling approach frequently taken is to partition physical space into small increments, e.g., inches, and to model the progress of objects on an inch-by-inch basis. Each time an object's position is updated, a check can be made to see whether the object has reached either the end-of-queue position or the end of the conveyor. This may be the approach which naturally comes first to mind. But the negative aspects of such a "natural" approach include these:

(1) The logical requirements of the model can be very demanding. For example, if object positions are updated in an unspecified order, the modeler must handle time ties very carefully. If the status of a particular inch of space is tested and/or updated in the same instant of simulated time for more than one object, but in the wrong order, difficulties may arise. For example, the first inch required by an upstream object may not yet have been vacated by a downstream object, if the upstream object's position is updated first. If objects queue up at the conveyor exit, time ties are certain to occur: when the first object in the queue is removed from the conveyor, all objects in the queue will advance in lock-step.

(2) The CPU time requirements of the model can also be significant. An object-by-object, inch-by-inch approach requires scheduling an event for every inch of progress for every moving object.

The discussion which follows presents a better approach for coping with accumulating conveyors. In the discussion, the following assumptions are made:

(1) An object is propelled onto the conveyor by the conveyor itself; i.e., objects are not "lifted" onto the conveyor. In other words, the leading (downstream) edge of the object is brought into contact with the upstream end of the conveyor, and the object moves onto the conveyor at the conveyor's speed.

(2) An object is propelled off the conveyor by the conveyor itself; i.e., objects are not "lifted" off the conveyor. In other words, the object is fully removed from the conveyor when its trailing (upstream) edge clears the exit point.

Let us now reconsider the "moving target" problem. Knowing exactly where (and when) an object reaches the end-of-queue may be unnecessary. Our major objective is to properly model the capacity of the conveyor and to properly model the transit times of objects on the conveyor. If meeting these objectives is sufficient, we can take a more abstract view of conveyor operation and ignore the end-of-queue problem.

If an object encounters no blockages enroute to the conveyor exit, its transit time is determined by the distance between the entry and exit points and the speed of the conveyor. In other words, it experiences the minimum time to travel the length of the conveyor. Objects which encounter blockages experience a transit time equal to the minimum time plus the time required for all predecessor (downstream) objects to be removed from the conveyor. This suggests the following simple model approach:

(1) The entire amount of space required to hold an object on the conveyor is requested prior to placing the object on the conveyor. If the total amount is not available, the object must wait.

(2) The space occupied by an object is relinquished in its entirety when the object begins to exit the conveyor.

(3) Each object placed on the conveyor first experiences the minimum travel time.

(4) The time required to remove objects at the exit point (in FIFO order) is modeled explicitly.

(5) The time-in-queue for objects is implicit in the removal times of all predecessor objects.

Parts (1) and (2) of the approach outlined above are a consequence of the assumptions that objects are propelled onto and off of the conveyor. Releasing the space about to be vacated by an object being removed allows consumption of an equal amount of space at the entry point. If the conveyor is full, as soon as removal of the leading object begins, another object can start its entry onto the conveyor. If the objects are all of the same length, this approach models conveyor space properly. The approach is inaccurate when the conveyor is full, and an object being removed is shorter than an object entering the conveyor. As the smaller object is propelled off the conveyor, the larger object could be propelled partially onto the conveyor, but part (1) above does not allow this. For modeling systems with objects of unequal length, this modeling inaccuracy could be removed by reverting to inch-by-inch modeling of objects' entry to the conveyor.

We now consider the timing aspects of the above modeling approach. Assume that a 1-foot long object is placed onto a conveyor that moves at one foot per second, at a distance of ten feet from the exit. Assume a time unit of seconds, and further assume that the object is placed on the conveyor at time 100. Assume that no blockages occur, and the object reaches the exit at time 110. Assume that another 1-foot object is placed on the conveyor at time 102. The separation between this object and the previous object must be 2 feet. Assume that the first object has to wait until time 119 for removal to start. As the first object is propelled off the conveyor, the second object moves into the space occupied by the first object. The second object reaches the exit at time 120. Knowing that the second object butts up against the first object at time 112 is probably of little value.

The "minimum travel time" approach just described is much simpler than the inch-by-inch approach. The minimum travel time approach is illustrated in this paper in models 1(a), (b), (c), and 2. Note that as described, the approach is language-independent. Although the illustrations provided here are modeled in GPSS/H, the "minimum travel time" technique clearly can be followed in other languages, e.g., SIMAN or SLAM.

3.2 Nonaccumulating Conveyors

Consider a nonaccumulating conveyor with one entry and one exit. When an object is placed on a non-accumulating conveyor, the object cannot move relative to the conveyor; i.e., obstruction of the object's motion necessitates stopping the conveyor. Object motion must be modeled in a fashion which reflects the conveyor halts which occur (1) whenever an object has reached the exit and has to wait to be removed, and (2) during entries and removals, if conveyor motion must be halted during these activities. The object-by-object, inch-by-inch approach described above, if applied in this case, would have disadvantages similar to those already indicated.

A markedly superior approach to modeling a nonaccumulating conveyor is to focus attention entirely on the single object which "leads" the other objects on the conveyor in the sense of being the one closest to the exit. In general, an object has already been on the conveyor for some time before its turn comes to be the "leader." When it becomes the leader, the remaining time needed for it to reach the exit can be computed this way:

(1) Start with the minimum travel time needed for an object to move from its original entry position to its exit position.

(2) Subtract the time it has already spent on the conveyor.

(3) Add any conveyor halt time already experienced by the object as of the time it becomes the new leader.

(4) As the simulation proceeds, add any additional conveyor halt time which the new leader experiences before it reaches the exit.

An object's minimum travel time is known from system data. The time an object has spent on the conveyor is easily determined by recording the time at which an object is placed on the conveyor, and then subtracting this value from the time at which the object becomes the new leader. The conveyor halt time experienced by an object is also easily computed by recording the conveyor's total halt time (accumulated from the start of the simulation) when the object is placed on the conveyor, and then subtracting this value from the total halt time observed at the time at which the object becomes the new leader.

Whereas steps (1), (2), and (3) above are straightforward, the ease (or lack thereof) of implementing step (4) may be language-dependent. Some languages provide the capability of temporarily suspending the elapsing of simulated time being experienced by an object. For example, suppose an object reaches a point in a model (e.g., becomes the leader on a conveyor) at simulated time 500, and is to be held at that point in the model for 25 simulated time units (e.g, is to reach the conveyor exit 25 time units later, assuming there are no conveyor halts in the interim). Further suppose that at simulated time 510, the elapsing of simulated time being experienced by this object is suspended for 5 time units (e.g., the conveyor halts for 5 time units). Then, because of the 5-time-unit halt, the object should not (try to) exit the conveyor until simulated time 530 (e.g., the leader will not reach the exit until time 530). This additional halt time can be taken into account easily in some languages, whereas doing so may require extra effort in other languages.

We term the above four-step approach to modeling nonaccumulating conveyor systems the "follow-the-leader" approach. This approach is straightforward in a language which provides the time-suspension capability corresponding to (4) above, and can be implemented in other languages as well. Concentrating on the timing of the leader allows the placing of upstream objects into a data structure which is time-invariant, i.e., does not require updating over simulated time. The data structure is updated only when new objects are placed on the conveyor and when the leader object is removed from the conveyor. The problem of delaying all objects on the conveyor when a stoppage occurs is reduced to the problem of delaying the leader object.

The implementation language used in this paper, GPSS/H, offers the time-suspension capability through the concept of Facility "availability" and "unavailability," and the corresponding FAVAIL and FUNAVAIL Blocks. GPSS User Chains are used to contain non-leader objects. These language features are used in this paper in three nonaccumulating conveyor models.

4. THREE MODELS OF A SIMPLE ACCUMULATING CONVEYOR SYSTEM

Three models of a simple accumulating conveyor system are presented in this section. The second and third of these models are variations of the first. Each model is described and presented in its own subsection.

Throughout the paper, discussion of GPSS aspects of the models presupposes a reader who is conversant with GPSS. Tutorial information about GPSS is available in textbooks (e.g., Bobillier, Kahan, and Probst (1976); Gordon (1975); Schriber (1974)) and in professional short courses (Schriber (1986)).

Output from simulations performed with models presented in the paper is described and discussed briefly when appropriate, but because of space limitations is not shown.

4.1 Model 1(a): A Simple Accumulating Conveyor System

4.1.1 Statement of the Problem

An accumulating conveyor is used to transport widgets (objects) from a single producer to a single consumer. The conveyor has one entry, one exit, and a capacity for 6 widgets. It takes 20 +/- 5 time units to produce a widget, and 20 +/- 15 time units to consume one. Minimum travel time on the conveyor is 10 time units. It takes 2 time units to place a widget on the conveyor, and 2 time units to remove a widget from the conveyor. An entry and a removal can take place (either partially or entirely) at the same time. The conveyor continues to move while an entry and/or a removal occurs. The producer does not start producing the next widget until after the most recently produced widget has been placed on the conveyor.

Build a model for the producer and consumer and the conveyor which links them. Design the model so that a simulation performed with it will stop when a specified number of widgets (e.g., 1,000 widgets) have been consumed.

4.1.2 Presentation and Discussion of the Model

The listing of a GPSS/H model for this system is shown in Figure 1. In the Producer Segment (statements 26 through 37), a single Transaction (the producer) repeatedly cycles through the steps of producing a widget, waiting (if necessary) for the conveyor space needed to place the widget on the conveyor, splitting off another Transaction to represent the widget on the conveyor, and then returning to start production of another widget. The capacity of the conveyor is modeled by using a Storage of capacity 6 (named ACCUM) to represent the conveyor. The ENTER Block (statement 34) causes the producer to be delayed when attempting to place an object onto the conveyor at a time at which the conveyor (the ACCUM Storage) is full.

The Conveyor Flow Segment implements the "minimum travel time" approach described in Section 3.1, and so consists of a single ADVANCE Block (statement 43, Figure 1).

The Consumer Segment (statements 45 through 54, Figure 1) uses the CONSUMER Facility to represent the consumer. Having experienced the minimum travel time needed to reach the conveyor exit, widgets capture the CONSUMER Facility first-come, first-served (first-on, first-off), go through the consumption cycle, and then release the CONSUMER and leave the model.

The model's Run Controls (statements 56 through 66, Figure 1) specify simulating until 1,000 widgets have been consumed when the conveyor has a capacity for six widgets (statement 19), then repeating the simulation with the conveyor's capacity reduced to five widgets (statement 64).

4.2 Model 1(b): A GPSS-based Variation on Model 1(a)

4.2.1 Statement of the Problem

In conveyor model 1(a), Transactions representing widgets available for consumption remain on the Current Events Chain until they succeed in capturing the consumer. Show how to modify model 1(a) so that such Transactions are kept on a User Chain (rather than on the Current Events Chain). The result may be to improve the model's run-time performance by keeping such blocked Transactions off the Current Events Chain. Note that because of the straightforward nature of the consumer, introduction of a User Chain into the model means that no Facility need be used to simulate the consumer. (See Schriber (1974), Section 7.6.2.)

4.2.2 Presentation and Discussion of the Model

Figure 2 shows statements 45 through 54 (the Consumer Segment) for model 1(b). The other model 1(b) statements are identical one-for-one to those in model 1(a), and are not repeated in Figure 2.

In the Figure 2 Consumer Segment, a LINK Block (statement 49) has replaced a corresponding model 1(a) SEIZE Block (statement 49 in Figure 1), and an UNLINK Block (statement 53) has replaced a corresponding model 1(a) RELEASE Block (statement 53 in Figure 1). The other corresponding Blocks in the Figure 1 and 2 Consumer Segments are identical.

All numeric results from the model 1(b) simulation match those from the model 1(a) simulation (as should be the case, given that the random number generators have the same starting points and that these are pseudo-random simulations).

In simulations performed under GPSS/H, Release 2, on an Amdahl 470/V8 computer, models 1(a) and 1(b) consumed 0.130 and 0.121 seconds of CPU time, respectively. As expected, model 1(b) runs somewhat more efficiently than model 1(a). In conveyor models with larger numbers of objects, the improvement would be more dramatic.

4.3 Model 1(c): A Second GPSS-based Variation on Model 1(a)

4.3.1 Statement of the Problem

In model 1(a), Transactions represent widgets available for consumption, and the consumer is modeled passively with a Facility. Show how to modify model 1(a) so that the current contents of a Storage represents the number of widgets available for consumption (meaning that when widget Transactions reach the conveyor exit, they can simply execute an ENTER Block and then leave the model). Use a Transaction to represent the consumer in the modified model. When the consumer Transaction has finished a consumption cycle, it can then try to move

```
GPSS/H VAX/VMS RELEASE 0.96 (UG176)     15 AUG 1986   07:49:52     FILE: CONVAC1A.GPS

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

   1     1                          SIMULATE
   2     2                          RMULT           111111111,333333333  RN1, RN2 INDEPENDENT

   4     4                  ************************************************************************
   5     5                  *                                                                      *
   6     6                  *       GPSS/H MODEL OF A SIMPLE ACCUMULATING CONVEYOR SYSTEM          *
   7     7                  *                                                                      *
   8     8                  *       *----------*                                   *----------*    *
   9     9                  *       * PRODUCER *--->|----------------------->|---* CONSUMER *    *
  10    10                  *       *----------*                                   *----------*    *
  11    11                  *                                                                      *
  12    12                  *       RULES:  1.  THE ENTIRE LENGTH OF THE CONVEYOR IS AN ACCUM-     *
  13    13                  *                   ULATION ZONE.                                      *
  14    14                  *               2.  THE PRODUCER STOPS ONLY WHEN THE CONVEYOR          *
  15    15                  *                   COMPLETELY FILLS UP.                               *
  16    16                  *                                                                      *
  17    17                  ************************************************************************

  19    19                  ACCUM   STORAGE         6                ROOM FOR 6 WIDGETS

  21    21                  PTIME   FUNCTION        RN1,C2           PRODUCER CYCLE TIME
  22    22                  0,15/1,25                                20 + - 5
  23    23                  CTIME   FUNCTION        RN2,C2           CONSUMER CYCLE TIME
  24    24                  0,5/1,35                                 20 + - 15

  26    26                  ************************************************************************
  27    27                  *       PRODUCER SEGMENT                                               *
  28    28                  ************************************************************************

  30    30              1           GENERATE        ,,,1             SINGLE PRODUCER
  31    31              2   PLOOP   ADVANCE         FN$PTIME         TIME TO PRODUCE A WIDGET
  32    32              3           QUEUE           ACCUM            CONVEYOR QUEUE
  33    33              4           ENTER           ACCUM            MAY BE COMPLETELY FULL
  34    34              5           DEPART          ACCUM            WE CAN GET ON NOW
  35    35              6           ADVANCE         2                LOAD TIME
  36    36              7           SPLIT           1,FLOW           OFFSPRING MODELS FLOW
  37    37              8           TRANSFER        ,PLOOP           CYCLE ENDLESSLY

  39    39                  ************************************************************************
  40    40                  *       CONVEYOR FLOW SEGMENT                                          *
  41    41                  ************************************************************************

  43    43              9   FLOW    ADVANCE         10               MINIMUM TRANSIT TIME

  45    45                  ************************************************************************
  46    46                  *       CONSUMER SEGMENT                                               *
  47    47                  ************************************************************************

  49    49             10           SEIZE           CONSUMER         EXIT ONE-AT-A-TIME
  50    50             11           ADVANCE         2                TIME REQUIRED TO UNLOAD
  51    51             12           LEAVE           ACCUM            ALLOW ENTRY OF ANOTHER WIDGET
  52    52             13           ADVANCE         FN$CTIME         CONSUMER CYCLE TIME
  53    53             14           RELEASE         CONSUMER         THIS WIDGET HAS BEEN CONSUMED
  54    54             15           TERMINATE       1                EXIT THE MODEL

  56    56                  ************************************************************************
  57    57                  *       RUN CONTROLS                                                   *
  58    58                  ************************************************************************

  60    60                          START           1000             CONSUME 1000 WIDGETS

  62    62                          RMULT           111111111,333333333  SAME SEEDS FOR 2ND RUN
  63    63                          CLEAR
  64    64                  ACCUM   STORAGE         5                TRY WITH ROOM FOR 5 WIDGETS
  65    65                          START           1000             CONSUME 1000 WIDGETS
  66    66                          END
```

Figure 1: Model 1(a): A Simple Accumulating Conveyor System

```
45    45              ************************************************************************
46    46              *       CONSUMER SEGMENT                                               *
47    47              ************************************************************************

49    49         10          LINK            CONSUMER,FIFO,NEXTC  EFFICIENT GPSS
50    50         11   NEXTC  ADVANCE         2                TIME REQUIRED TO UNLOAD
51    51         12          LEAVE           ACCUM            ALLOW ENTRY OF ANOTHER WIDGET
52    52         13          ADVANCE         FN$CTIME         CONSUMER CYCLE TIME
53    53         14          UNLINK          CONSUMER,NEXTC,1  EFFICIENT GPSS
54    54         15          TERMINATE       1                EXIT THE MODEL
```

Figure 2: Model 1(b): A GPSS-Based Variation on Model 1(a)

```
40    40              ************************************************************************
41    41              *       CONVEYOR FLOW SEGMENT                                          *
42    42              ************************************************************************

44    44          9   FLOW   ADVANCE         10               MINIMUM TRAVEL TIME
45    45         10          ENTER           AVAIL            WIDGET AVAIL FOR PROCESSING
46    46         11          TERMINATE       0                EXIT THE MODEL

48    48              ************************************************************************
49    49              *       CONSUMER SEGMENT                                               *
50    50              ************************************************************************

52    52         12          GENERATE        ,,,1             SINGLE CONSUMER
53    53         13          ASSIGN          1,1000,PH        CONSUME 1000 WIDGETS
54    54         14   CLOOP  GATE SNE        AVAIL            WAIT FOR SOMETHING TO DO
55    55         15          ADVANCE         2                TIME REQUIRED TO UNLOAD
56    56         16          LEAVE           ACCUM            ALLOW ENTRY OF ANOTHER WIDGET
57    57         17          LEAVE           AVAIL            INDICATE THIS ONE USED
58    58         18          ADVANCE         FN$CTIME         CONSUMER CYCLE TIME
59    59         19          LOOP            1PH,CLOOP        CONSUME N WIDGETS
60    60         20          TERMINATE       1                SHUT DOWN THE MODEL
```

Figure 3: Model 1(c): A Second GPSS-Based Variation on Model 1(a)

into a refusal-mode GATE to test for the presence of one or more additional consumable widgets. (See Henriksen (1981).)

4.3.2 Presentation and Discussion of the Model

Figure 3 shows statements 40 through 60 (the Conveyor Flow Segment and the Consumer Segment) for model 1(c). The other model 1(c) statements are identical one-for-one to those in model 1(a), and so are not repeated in Figure 3.

As was the case in models 1(a) and 1(b), a Storage (named ACCUM) is used to model the capacity of the conveyor. In the Conveyor Flow Segment, a second Storage (named AVAIL) is used to model the supply of widgets available for consumption at the conveyor exit. A widget which has spent the minimum travel time needed to reach the conveyor exit executes an ENTER Block (statement 45, Figure 3) to update the count of widgets available for consumption, then leaves the model.

In the Consumer Segment, a single Transaction loops 1,000 times in the process of consuming 1,000 widgets (statements 54 through 59, Figure 3), then terminates to stop the simulation (statement 60). Note how the consumer uses a refusal-mode GATE (statement 54) to test for the availability of another consumable widget, and how it updates the count of consumable widgets by executing an appropriate LEAVE Block (statement 57).

To summarize this modeling approach, a pair of Storages is used to model the conveyor, one as a constraint on object entry and the other as a constraint on the consumer. An object cannot enter the conveyor when the former Storage is full, and the consumer cannot consume until the latter Storage is non-empty. This approach is applicable only to systems in which objects are homogeneous, because the objects are represented entirely by the contents of Storages. If the consumer's behavior depends on the attributes of nonhomogeneous objects, a different modeling approach may be required. If the behavior of nonhomogeneous objects can be modeled by a random sampling process, the consumer can generate the required attributes on demand; i.e., attributes needn't be stored in data structures used to explicitly represent objects.

All numeric results from the model 1(c) simulation match those from the model 1(a) and (b) simulations, as is to be expected.

5. A MODEL FOR A MODERATELY COMPLEX ACCUMULATING CONVEYOR

5.1 Statement of the Problem

An accumulating conveyor is used to transport widgets from two (potential) producers to two (potential) consumers. The conveyor has one entry, one exit, and a capacity for 20 widgets. It takes 20 +/- 5 time units to produce a widget, and 30 +/- 20 time units to consume one. Minimum travel time on the conveyor is 40 time units. It takes 2 time units to place a widget on the conveyor, and 2 time units to remove a widget from the conveyor. Not more than one entry and one removal can take place at any one time, but an entry and a removal can take place (either partially or entirely) at the same time. The conveyor continues to move while an entry and/or a removal occurs. At the earliest, a producer does not start producing its next widget until after the widget it most recently produced has been placed on the conveyor.

As shown in the upper part of Figure 4, the length of the conveyor is divided into four logical zones of equal capacity (10 widgets per zone), with zone 1 at the conveyor entry and zone 4 at its exit. Normally, both producers make widgets, whereas only one consumer consumes them. If zone 3 fills up, however, the idle consumer begins to consume (so that there will be two active consumers temporarily). Later, when zone 4 is no longer full, the consumer who had most recently been officially idle continues to consume, whereas the other consumer becomes officially idle (after finishing its current consumption cycle).

The potential also exists for having one or both producers become temporarily idle. If zone 2 fills, the producer who has been producing for the longest time (since its immediately preceding period of official idleness) stops producing temporarily, and then starts producing again when zone 3 is no longer full. If zone 1 fills, then the single producer who is active at that time necessarily stops producing (because of the requirement that it must be able to place its most recently completed widget onto the conveyor before starting its next production cycle).

Build a model for the producers and consumers and the conveyor which links them. Design the model so that a simulation performed with it will stop when a specified number of widgets (e.g., 1,000 widgets) have been made available for consumption.

5.2 Presentation and Discussion of the Model

The listing of a GPSS model for this system is shown in Figure 4. In the Producer Segment (statements 46 through 62, Figure 4), two Transactions represent the two producers. These two producers are tagged as PROD1 and PROD2 in their respective PRODNO Parameters (see statements 36, 37, and 51 in Figure 4). When the PROD1 or PROD2 Logic Switch is Reset (zero), then the associated producer can start a production cycle (refusal mode GATE, statement 53). After capturing the corresponding Facility (PROD1 or PROD2) a producer proceeds with production of a widget, (eventual) capture of the conveyor entry, waiting (if necessary) for space in zone 1, placing the widget on the conveyor, splitting off another Transaction to represent the widget on the conveyor, and then returning to wait (if necessary) to start production of another widget. Note that the SEIZE Block (statement 54) never denies entry to a producer Transaction; its sole purpose is to gather producer statistics.

In the Figure 4 Conveyor Flow Segment (statements 64 through 80), widgets go through a sequence of appropriately layered ENTER, ADVANCE, and LEAVE Blocks to model their progress through conveyor zones 1 through 4. The "minimum travel time" approach discussed in Section 3.1 is used for each zone. A widget which has eventually spent the minimum travel time needed in zone 4 to reach the conveyor exit executes an ENTER Block (statement 79, Figure 4), thereby updating the current contents of a Storage used to model the number of widgets available for consumption. (The approach here is analogous to that in model 1(c).) The widget Transaction then terminates.

The Consumer Segment in model 2 (statements 82 through 99, Figure 4) is analogous to the Consumer Segment for model 1(c) in that consumers are modeled by Transactions which use a refusal-mode GATE (statement 89) to test for the availability of another consumable widget, and update the record of consumable widgets by executing an appropriate LEAVE Block (statement 93). The model 2 Consumer Segment also bears similarities to the model 2 Producer Segment in that the consumer Transactions are tagged as CONS1 and CONS2 in their respective CNSNO Parameters (see statements 38, 39, and 87, Figure 4). When the CONS1 or CONS2 Logic Switch is Reset (zero), then the associated consumer can start a consumption cycle (refusal-mode GATE, statement 89). After capturing the corresponding Facility (CONS1 or CONS2), the consumer Transaction proceeds through the remainder of its consumption cycle. (Note that the SEIZE Block (statement 90) never denies entry to a producer Transaction; its sole purpose is to gather producer statistics.) After completing the remainder of the consumption cycle, the consumer Transaction then returns to wait (if necessary) to start its next consumption cycle.

Zones 2 and 3 are monitored by a monitor Transaction which loops in the Producer Monitor Segment (statements 101 through 113, Figure 4). The monitor Transaction controls producer Transactions by setting and resetting the PROD1 and PROD2 Logic Switches, which are "sensed" by producer Transactions, as dicussed above. The monitor Transaction alternates between PROD1 and PROD2 to accomplish the idling of the appropriate producer (when zone 2 fills) and then the eventual restarting of that producer (when

```
GPSS/H VAX/VMS RELEASE 0.96 (UG176)     15 AUG 1986   07:50:37     FILE: CONVAC2.GPS

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

   1     1                           SIMULATE
   2     2                           RMULT            111111111,333333333  RN1, RN2 INDEPENDENT

   4     4                  *************************************************************************
   5     5                  *                                                                       *
   6     6                  *      GPSS/H MODEL OF A MODERATELY COMPLEX ACCUMULATING CONVEYOR       *
   7     7                  *                                                                       *
   8     8                  * *------------*                                       *------------*  *
   9     9                  * * PRODUCER 1 *---*                               *---* CONSUMER 1 *  *
  10    10                  * *------------*   ¦ ZONE1   ZONE2   ZONE3   ZONE4 ¦   *------------*  *
  11    11                  *                  *------->¦------->¦------->¦------*                   *
  12    12                  * *------------*   ¦                               ¦   *------------*  *
  13    13                  * * PRODUCER 2 *---*                               *---* CONSUMER 2 *  *
  14    14                  * *------------*                                       *------------*  *
  15    15                  *                                                                       *
  16    16                  *        RULES:  1.  NORMALLY, BOTH PRODUCERS RUN CONTINUOUSLY.         *
  17    17                  *                2.  NORMALLY, ONLY ONE CONSUMER OPERATES AT A TIME.    *
  18    18                  *                3.  IF ZONE3 FILLS UP, THE IDLE CONSUMER STARTS UP.    *
  19    19                  *                    AS SOON AS ZONE4 BECOMES NOT FULL, THE CONSUMER    *
  20    20                  *                    WHICH WAS RUNNING WHEN THE IDLE CONSUMER WAS       *
  21    21                  *                    RESTARTED IS STOPPED.                              *
  22    22                  *                4.  IF ZONE2 FILLS UP, THE PRODUCER WHICH HAS BEEN     *
  23    23                  *                    RUNNING FOR THE LONGEST TIME IS FORCED TO STOP     *
  24    24                  *                    UNTIL ZONE 3 IS NOT FULL.                          *
  25    25                  *                5.  IF ZONE1 FILLS UP, THE SINGLE ACTIVE PRODUCER      *
  26    26                  *                    STOPS.                                             *
  27    27                  *                                                                       *
  28    28                  *************************************************************************

  30    30                  ZONE1    STORAGE          5                ZONE1 HOLDS 5 WIDGETS
  31    31                  ZONE2    STORAGE          5                ZONE2 HOLDS 5 WIDGETS
  32    32                  ZONE3    STORAGE          5                ZONE3 HOLDS 5 WIDGETS
  33    33                  ZONE4    STORAGE          5                ZONE4 HOLDS 5 WIDGETS
  34    34                  END4     STORAGE          5                WIDGETS AVAIL FOR CONSUMPTION

  36    36                  PROD1    EQU              1,F,L            PRODUCER 1
  37    37                  PROD2    EQU              2,F,L            PRODUCER 2
  38    38                  CONS1    EQU              3,F,L            CONSUMER 1
  39    39                  CONS2    EQU              4,F,L            CONSUMER 2

  41    41                  PTIME    FUNCTION         RN1,C2           PRODUCER CYCLE TIME
  42    42                  0,15/1,25                                  20 + - 5
  43    43                  CTIME    FUNCTION         RN2,C2           CONSUMER CYCLE TIME
  44    44                  0,10/1,50                                  30 + - 20

  46    46                  *************************************************************************
  47    47                  *        PRODUCER SEGMENT                                               *
  48    48                  *************************************************************************

  50    50             1             GENERATE         ,,,2             TWO PRODUCERS ACTIVE
  51    51             2             ASSIGN           PRDNO,PROD1+N(*),PH PRODUCER ID (PROD1 OR PROD2)

  53    53             3    PLOOP    GATE LR          PH$PRDNO         WAIT UNTIL MACHINE ACTIVE
  54    54             4             SEIZE            PH$PRDNO         INDICATE PRODUCER ACTIVE
  55    55             5             ADVANCE          FN$PTIME         TIME TO PRODUCE A WIDGET
  56    56             6             SEIZE            ENTRY            ONE-AT-A-TIME
  57    57             7             GATE SNF         ZONE1            MAY BE COMPLETELY FULL
  58    58             8             ADVANCE          2                PLACE ON CONVEYOR
  59    59             9             RELEASE          ENTRY            ALLOW ANOTHER WIDGET TO ENTER
  60    60            10             SPLIT            1,INTO1          ROUTE WIDGET TO THE CONVEYOR
  61    61            11             RELEASE          PH$PRDNO         INDICATE PRODUCER IDLE
  62    62            12             TRANSFER         ,PLOOP           GO PRODUCE ANOTHER WIDGET
```

Figure 4: Model 2: A Moderately Complex Accumulating Conveyor System

```
        ***********************************************************************
        *       CONVEYOR FLOW SEGMENT                                         *
        ***********************************************************************

   13   INTO1   ENTER       ZONE1           ENTER FIRST ZONE
   14           ADVANCE     10              ZONE 1 MIN TRANSIT TIME
   15           ENTER       ZONE2           POSSIBLE BLOCKAGE
   16           LEAVE       ZONE1           LEAVE PREVIOUS ZONE
   17           ADVANCE     10              ZONE 2 MIN TRANSIT TIME
   18           ENTER       ZONE3           POSSIBLE BLOCKAGE
   19           LEAVE       ZONE2           LEAVE PREVIOUS ZONE
   20           ADVANCE     10              ZONE 3 MIN TRANSIT TIME
   21           ENTER       ZONE4           POSSIBLE BLOCKAGE
   22           LEAVE       ZONE3           LEAVE PREVIOUS ZONE
   23           ADVANCE     10              ZONE 4 MIN TRANSIT TIME
   24           ENTER       END4            WIDGET AVAIL FOR CONSUMPTION
   25           TERMINATE   1               END OF THE LINE

        ***********************************************************************
        *       CONSUMER SEGMENT                                              *
        ***********************************************************************

   26           GENERATE    ,,,2,1          TWO CONSUMERS ACTIVE
   27           ASSIGN      CNSNO,CONS1+N(*),PH CONSUMER ID (CONS1 OR CONS2)

   28   CLOOP   GATE LR     PH$CNSNO        WAIT UNTIL OK TO CONSUME
   29           SEIZE       PH$CNSNO        INDICATE CONSUMER ACTIVE
   30           SEIZE       EXIT            EXIT ONE-AT-A-TIME
   31           GATE SNE    END4            WAIT UNTIL A WIDGET AVAIL
   32           LEAVE       END4            REMOVE FROM CONVEYOR
   33           ADVANCE     2               WIDGET REMOVAL TIME
   34           LEAVE       ZONE4           ALLOW ENTRY OF ANOTHER WIDGET
   35           RELEASE     EXIT            ALLOW ANOTHER EXIT
   36           ADVANCE     FN$CTIME        CONSUMER CYCLE TIME
   37           RELEASE     PH$CNSNO        INDICATE CONSUMER IDLE
   38           TRANSFER    ,CLOOP          LOOP CONTINUOUSLY

        ***********************************************************************
        *       PRODUCER MONITOR SEGMENT                                      *
        ***********************************************************************

   39           GENERATE    ,,,1,2          HIGH PRIORITY MONITOR
   40           ASSIGN      STOP,PROD1,PH   ID OF CONSUMER TO STOP
   41   MFULL3  GATE SF     ZONE2           WAIT FOR ZONE2 TO FILL
   42           LOGIC S     PH$STOP         STOP APPROPRIATE PRODUCER
   43           GATE SNF    ZONE3           WAIT UNTIL ZONE3 NOT FULL
   44           LOGIC R     PH$STOP         RESTART APPROPRIATE PRODUCER
   45           ASSIGN      STOP,PROD1+PROD2-PH$STOP  "FLIP" PRODUCER ID
   46           GATE SNF    ZONE2           WAIT FOR ZONE2 FLOW INTO ZONE3
   47           TRANSFER    ,MFULL3         GO MONITOR ZONE2

        ***********************************************************************
        *       CONSUMER MONITOR SEGMENT                                      *
        ***********************************************************************

   48           GENERATE    ,,,1,2          HIGH PRIORITY MONITOR
   49           ASSIGN      STOP,CONS2,PH   ID OF CONSUMER TO STOP
   50   STOPC   LOGIC S     PH$STOP         STOP APPROPRIATE CONSUMER
   51           GATE SF     ZONE3           WAIT FOR ZONE3 TO FILL
   52           LOGIC R     PH$STOP         START APPROPRIATE CONSUMER
   53           GATE SNF    ZONE4           WAIT UNTIL ZONE4 NOT FULL
   54           ASSIGN      STOP,CONS1+CONS2-PH$STOP  "FLIP" CONSUMER ID
   55           GATE SNF    ZONE3           WAIT FOR ZONE3 FLOW INTO ZONE4
   56           TRANSFER    ,STOPC          GO STOP THE OTHER GUY

        ***********************************************************************
        *       RUN CONTROLS                                                  *
        ***********************************************************************

                START       1000            MAKE 1000 WIDGETS AVAILABLE
        *                                   FOR CONSUMPTION
                END
```

Figure 4: Model 2: A Moderately Complex Accumulating Conveyor System (Continued)

zone 3 is no longer full).

In an analagous manner, zones 3 and 4 are monitored by a monitor Transaction which loops in the Consumer Monitor Segment (statements 115 through 127, Figure 4). The monitor Transaction controls producer Transactions by setting and resetting the CONS1 and CONS2 Logic Switches, which are "sensed" by consumer Transactions, as discussed above. The monitor Transaction alternates between CONS1 and CONS2 to accomplish the activation of the appropriate consumer (when zone 3 fills) and then the eventual idling of the other consumer (when zone 4 is not full).

Model 2 is a natural extension of models 1(a), (b), and (c) with respect to use of the "minimum travel time" concept to model widget movement along a conveyor, and is a natural extension of model 1(c) in terms of the mechanism used to reflect the availability of widgets for consumption, and the representation of producers and consumers with Transactions. Most of the complications in the model 2 logic involve the monitoring of the status of conveyor zones and the resulting need to control movement of the consumer and producer Transactions. Although this monitoring and control is not intrinsic to conveyor modeling as such, it is representative of the types of logical issues which frequently occur in the modeling of moderately complex systems in general.

In output produced when model 2 was run under GPSS/H, Release 2, the simulation stopped at clock time 15,993.6. The average number of widgets in zones 1 through 4 was 4.33, 4.92, 4.96, and 4.98, respectively. Average times in these respective zones were 68.2, 77.9, 78.8, and 79.6 time units per widget. Producers 1 and 2 were officially at work 77.5 and 78.2 of the time, whereas consumers 1 and 2 were officially at work 100 and 98.8 of the time, respectively. The conveyor entry and exit were utilized 27.4 and 12.8 of the time, respectively.

The disparity between the entry and exit utilizations is explained as followes. The entry is utilized not just while being actively used, but also while a producer is waiting for space in zone 1 into which a widget can be placed (see statements 56 through 59 in Figure 4). Zone 1 is frequently full, resulting in a relatively long space-wait on average and a relatively high aggregate entry-utilization statistic. The exit is likewise utilized not just while being actively used, but also while a consumer is waiting for availability of a consumable widget at the end of zone 4 (see statements 90 through 98 in Figure 4). Consumable widgets are usually available at the end of zone 4, however, resulting in a relatively short consumable-widget wait on average, and a relatively low aggregate exit-utilization statistic.

.
.
.

6. TWO MODELS OF A SIMPLE NONACCUMULATING CONVEYOR SYSTEM

Two models for a simple nonaccumulating conveyor system are presented in this section. The second of these models is a variation of the first. Each model is described in its own subsection.

6.1 Model 3(a): A Nonaccumulating Conveyor which Moves During Entries and Removals

6.1.1 Statement of the Problem

A nonaccumulating conveyor is used to transport widgets from a single producer to a single consumer. The conveyor has one entry and one exit. It takes 20 +/- 5 time units to produce a widget, and 20 +/- 15 time units to consume one. Minimum travel time on the conveyor is 30 time units. It takes 2 time units to place a widget on the conveyor, and 2 time units to remove a widget from the conveyor. Not more than one entry and one removal can take place at any one time, but an entry and a removal can take place (either partially or entirely) at the same time. The conveyor must be moving while an entry and/or a removal occur. (Think of this as meaning that to place a widget on the conveyor, the producer first places the leading edge of the widget on the trailing edge of the moving belt, with the belt's motion then helping pull the rest of the widget along until the entire widget is in contact with the belt.) A producer does not start producing its next widget until after the most recently produced widget has been placed on the conveyor.

As explained in Section 2, a nonaccumulating conveyor is one which must be stopped when an object which has reached the conveyor exit has to wait for a consumer to remove it. When the removal occurs, however, the conveyor must have resumed its motion. (Think of this as meaning that to remove a widget from the conveyor, the consumer first lifts the leading edge of the widget from the leading edge of the belt, with the belt's motion then helping move the rest of the widget along until none of the widget is any longer in contact with the belt.)

The simple condition needed to place a widget on the nonaccumulating conveyor in this problem is that the conveyor be moving. This means the capacity of the nonaccumulating conveyor need not be incorporated explicitly into a model of the system, and need not even be formally known. The conveyor's capacity can be deduced, however, from knowledge of the time required to place a widget on the conveyor, the minimum time then required by a widget to travel to the exit, and the time required to remove it from the conveyor. The entry and removal times of 2 time units imply that the conveyor travels one widget-length every 2 time units. The minimum travel time of 30 time units then means the conveyor moves a distance equal to the length of 15 widgets while minimum travel time elapses. The conveyor consequently has a capacity of 17 widgets (1 for entry, 15 for traveling, and 1 for removal).

Build a model for the producer and consumer and the nonaccumulating conveyor which links them. Design the model so that a simulation performed with it will stop when a specified number of widgets (e.g., 1,000 widgets) have been consumed.

6.1.2 Presentation and Discussion of the Model

The listing of a GPSS model for this system is shown in Figure 5. In the Producer Segment (statements 23 through 35), a single producer Transaction loops repeatedly through the steps of producing a widget, waiting (if necessary) for the moving- conveyor condition needed to place the widget on the conveyor, splitting off another Transaction to represent the loaded widget, and then returning to begin producing another widget.

Whether or not the conveyor is moving (so that an entry can take place) is reflected by the status of the CONVEYOR Facility. When the conveyor is moving, this Facility is idle, and so can be captured by the producer who wants to place a widget on the belt. When the conveyor is halted, however, the CONVEYOR Facility is placed into a state of unavailability. This has one of two alternative effects on the producer:

(1) If the producer is producing, it cannot later capture the CONVEYOR Facility, and so cannot place a widget on the halted conveyor. (A Transaction cannot capture a Facility unless it is both idle AND available.)

(2) If the producer is in the process of placing a widget on the conveyor when the halt occurs, it controls the CONVEYOR Facility at the time. When this Facility becomes unavailable at the time the conveyor halts, this results in a "time suspension" (the suspending of ongoing simulated time) for the producer, which means the ongoing entry step is suspended and will not resume until the conveyor later starts to move again (at which time the CONVEYOR Facility will again be placed into a state of availability).

In the Conveyor Flow Segment (statements 37 through 54), an arriving widget (which has just been placed on the conveyor) records its time of arrival (statement 41) and the conveyor's total halt time measured from the start of the simulation (statements 42 and 43). It then either becomes the "leader" on the conveyor (see the Section 3.2 discussion and statements 44 and 46) or goes to the back of the BELT User Chain (statement 44) if there is already a leader.

After the current leader has reached the conveyor exit, (comes out of the statement 46 ADVANCE Block), it examines the consumer's status (statement 49) and immediately captures the consumer if it is idle (statement 60) or records the time (statement 50), halts the conveyor (statement 51), and waits (statement 52) for the busy consumer to become idle. In this latter case, when the consumer does become idle the leader restarts the conveyor (statement 53), updates the conveyor's total halt time (statement 54), and captures the consumer (statement 60).

When the leader has captured the consumer, it UNLINKs the next leader from the BELT User Chain (statement 61). If the BELT Chain is empty, the attempted unlink Resets the Chain's Link Indicator. This means that when the producer later places the next widget on the conveyor, the widget won't go onto the User Chain (at statement 44), but will immediately take on the role of leader.

In the Consumer Segment (statements 56 through 65, Figure 5), the CONSUMER Facility is used to model the consumer. The first two Blocks in this segment (statements 60 and 61) have already been commented on in the above Conveyor Flow Segment discussion. The remaining Blocks in the Consumer Segment (statements 62 through 65) are straightforward.

As discussed earlier, the producer Transaction must capture the CONVEYOR Facility (statement 30) as a prerequisite to placing a widget on the conveyor, thereby satisfying the requirement that the conveyor be moving when a widget is placed onto the conveyor. The reader may wonder why the consumer Transaction doesn't also have to capture some Facility before initiating a removal, corresponding to the requirement that the conveyor be moving when a widget is removed. The consumer need not check for conveyor movement prior to a removal, because the old leader widget who has just captured the CONSUMER Facility (statement 60) has either just restarted the conveyor (statement 53) or, upon arriving at the exit, found the consumer to be idle and so didn't have to halt the conveyor in the first place.

In output from a model 3 simulation performed under GPSS/H, Release 2, the simulation stopped at clock time 24,352.2. The conveyor's total halt time as of then was 6,385.9 time units. which means the conveyor was in a halted state somewhat more than 25% of the time. The consumer was captured (either removing or consuming) 90% of the time. The BELT User Chain had an average content of 0.8 widgets, with an average widget residence time on the User Chain of 18.7 time units. The User Chain's maximum content was 2, meaning there were never more than 3 widgets on the conveyor (1 leader widget, and 2 widgets waiting their turn to become the leader). The interested reader can easily determine from the model's initial conditions (empty conveyor), cycle times for production and consumption, and minimum conveyor travel time that the above numbers are reasonable.

6.2 Model 3(b): A Nonaccumulating Conveyor which is Halted During Entries and Removals

6.2.1 Statement of the Problem

In nonaccumulating conveyor model 3(a), widgets could not be placed onto or removed

```
GPSS/H VAX/VMS RELEASE 0.96 (UG176)     15 AUG 1986    07:50:55     FILE: CONVNAC1.GPS

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

   1     1                           SIMULATE
   2     2                           RMULT            111111111,333333333     RN1, RN2 INDEPENDENT

   4     4                  *********************************************************************
   5     5                  *                                                                   *
   6     6                  *        GPSS/H MODEL OF A SIMPLE NON-ACCUMULATING CONVEYOR SYSTEM  *
   7     7                  *                                                                   *
   8     8                  *        *__________*                              *__________*     *
   9     9                  *        * PRODUCER *--->|---------------------->|---* CONSUMER *   *
  10    10                  *        *__________*                              *__________*     *
  11    11                  *                                                                   *
  12    12                  *        RULES:  1.  IF A WIDGET REACHES THE CONSUMER END OF THE    *
  13    13                  *                    CONVEYOR, AND THE CONSUMER IS BUSY, THE CONVEYOR *
  14    14                  *                    IS HALTED.                                     *
  15    15                  *                                                                   *
  16    16                  *********************************************************************

  18    18                  PTIME    FUNCTION         RN1,C2            PRODUCER CYCLE TIME
  19    19                  0,15/1,25                                   20 + - 5
  20    20                  CTIME    FUNCTION         RN2,C2            CONSUMER CYCLE TIME
  21    21                  0,5/1,35                                    20 + - 15

  23    23                  *********************************************************************
  24    24                  *        PRODUCER SEGMENT                                           *
  25    25                  *********************************************************************

  27    27               1           GENERATE         ,,,1,,1PL         SINGLE PRODUCER
  28    28               2  PLOOP    ADVANCE          FN$PTIME          PRODUCTION CYCLE TIME
  29    29               3           QUEUE            ENTRY             MEASURE TIME TO GET ON CONVEYOR
  30    30               4           SEIZE            CONVEYOR          CONVEYOR MUST BE MOVING
  31    31               5           ADVANCE          2                 PLACE WIDGET ON CONVEYOR
  32    32               6           RELEASE          CONVEYOR          IT'S ON NOW
  33    33               7           DEPART           ENTRY             MEASURE DELTA T
  34    34               8           SPLIT            1,FLOW            OFFSPRING XACT MODELS FLOW
  35    35               9           TRANSFER         ,PLOOP            PRODUCER CYCLES ENDLESSLY

  37    37                  *********************************************************************
  38    38                  *        CONVEYOR FLOW SEGMENT                                      *
  39    39                  *********************************************************************

  41    41              10  FLOW     MARK             ,                        SAVE TIME ONTO CONVEYOR
  42    42              11           ASSIGN           SAVESUMH,_               SAVE TOTAL CONVEYOR
  43    43              11                            XL$SUMHALTS,PL           HALT TIME TO DATE
  44    44              12           LINK             BELT,FIFO,NEW1ST         NEW LEADER -> NEW1ST

  46    46              13  NEW1ST   ADVANCE          30-M1_                   TRAVEL TIME = 30 - TIME ON
  47    47              13                            +(XL$SUMHALTS-PL$SAVESUMH)    SO FAR + HALT TIME
  48    48                  *                                                       SO FAR WHILE ON
  49    49              14           GATE U           CONSUMER,CONSUME         SKIP IF CONSUMER IDLE
  50    50              15           MARK             ,                        START OF HALT INTERVAL
  51    51              16           FUNAVAIL         CONVEYOR                 STOP THE CONVEYOR
  52    52              17           GATE NU          CONSUMER                 WAIT FOR IDLE CONSUMER
  53    53              18           FAVAIL           CONVEYOR                 RESTART THE CONVEYOR
  54    54              19           SAVEVALUE        SUMHALTS+,M1,XL          ACCUMULATE HALT TIME

  56    56                  *********************************************************************
  57    57                  *        CONSUMER SEGMENT                                           *
  58    58                  *********************************************************************

  60    60              20  CONSUME  SEIZE            CONSUMER          WIDGET AVAILABLE; GET CONSUMER
  61    61              21           UNLINK           BELT,NEW1ST,1     UNLEASH A NEW LEADER
  62    62              22           ADVANCE          2                 TAKE WIDGET FROM CONVEYOR
  63    63              23           ADVANCE          FN$CTIME          CONSUMPTION CYCLE
  64    64              24           RELEASE          CONSUMER          DONE
  65    65              25           TERMINATE        1                 WIDGET CONSUMED
```

Figure 5: Model 3(a): A Nonaccumulating Conveyor which Moves During Entries and Removals

```
*************************************************************************
*    RUN CONTROLS                                                       *
*************************************************************************

         START        1000                      CONSUME 1000 WIDGETS
         END
```

Figure 5: Model 3(a) (Continued)

from the conveyor unless the conveyor was moving. Show how to modify model 3(a) under the assumption that widgets cannot be placed onto or removed from the conveyor unless it is not moving. Assume the producer cannot begin a production cycle until the previously produced widget has been placed onto the halted conveyor and the conveyor has been restarted.

6.2.2 Presentation and Discussion of the Model

The reader who has carefully followed the reasoning presented thus far in the paper may want to try his or her hand at building a model for this problem before reading on (!).

This problem presents several complications above and beyond those of the system of model 3(a). A first complication is that there are now two independent reasons for stopping the conveyor:

(1) As in model 3(a), the conveyor might have to be stopped because the leader has reached the exit when the consumer is busy consuming some other widget. (This was the only reason for stopping the conveyor in model 3(a).)

(2) The conveyor might have to be stopped for purposes of placing a widget on it.

Note that the conveyor never has to be halted by anyone for purposes of removing a widget from it; if there is a widget ready to be removed, then the conveyor has already been halted because of (1).

A second complication is that if the conveyor happens to be halted for reason (1) above (or, having been halted for reason (1), continues in a halted state because a removal is taking place), then the producer can at least start to place a widget on the conveyor without having to halt it first.

When a removal concludes, the conveyor normally can be restarted. This is not the case, however, if an entry is in progress at the time a removal concludes. This another complication.

Yet another complication is that when an entry concludes, the conveyor can be restarted if there is not a widget at the conveyor exit and no removal is in progress, but cannot be restarted otherwise.

As indicated above, there may be overlapping reasons for the conveyor to be halted. This affects the accumulation of total conveyor halt time, simply because two different Transactions (the producer, and the leader widget) may be taking responsibility at one and the same time for updating the conveyor's total halt time, and yet double-counting during periods of overlap must be avoided.

In the model, the current content of the STOPPER Storage is used to count the number of Transactions which are currently causing the conveyor to be halted. A current content of zero indicates that the conveyor is moving; a current content of one indicates that the conveyor is halted on behalf of exactly one Transaction (the producer, or the leader widget at the exit, or the leader while it is being removed); and a current content of two indicates that the conveyor is halted for each of two independent reasons (because of an entry; and because there is a leader at the exit or because the leader is being removed). A halted conveyor cannot be restarted until the STOPPER Storage is empty.

The LEADER Facility is used in the model to represent the conveyor. To stop the conveyor for an entry, the producer Transaction simply renders the LEADER Facility unavailable. (It is not an error to redundantly (try to) place a Facility into a state of unavailability when the Facility already is unavailable. This explains why the producer and leader Transactions simply execute FUNAVAIL Blocks in the model when it is logically appropriate for them to do so, without regard to the current availability status of the LEADER Facility.)

When the producer has finished an entry, or the leader has been removed, the corresponding Transaction should restart the conveyor (by rendering the LEADER Facility available again) only if the conveyor is not to remain stopped for some other reason (that is, because an entry is ongoing when a removal finishes; or because there is a leader at the exit; or because a removal is ongoing when an entry finishes). The producer or leader Transaction, after leaving the STOPPER Storage discussed above, can condition the return of the LEADER Facility to an available state on whether or not the STOPPER Storage is empty. An empty Storage means there is now no reason why the conveyor is halted, and so it should be restarted; a nonempty Storage means there continues to be another reason why the conveyor is halted, and so it should not be restarted.

As for accumulating total conveyor halt time, this is easily done in GPSS/H, Release 2, which offers a Standard Numerical

Attribute whose class name is FUT (for Facility Unavailable Time). The FUT attribute of the LEADER Facility (FUT$LEADER) is a direct measure of the total conveyor halt time. Hence, the simulationist does not have to provide logic to avoid double counting when the conveyor is halted for two independent reasons at one and the same time.

A listing of model 3(b) is given in Figure 6. The ways in which the special logical complications of the problem have been accounted for in the Figure 6 model have been outlined above in some detail. The reader should study the particulars of the model and interpret them in light of the above discussion.

7. A MODEL OF AN AIRPLANE BAGGAGE LOADING CONVEYOR SYSTEM

7.1 Statement of the Problem

A nonaccumulating conveyor is used to transport baggage from ground-level baggage carts up into the underbelly of an airplane. The conveyor has one entry and one exit. Minimum transit time on the conveyor is 12 time units. It takes a (human) loader 10 +/- 4 time units to obtain a bag from a baggage cart and another 1 +/- 0.5 time units to place the bag on the conveyor. Not more than one entry can take place at a time.

Two loaders work at obtaining bags from the baggage carts and placing them on the conveyor.

At the conveyor exit, bags fall off the head of the conveyor into a small buffer area which has a capacity for 3 bags. If this buffer is full when a bag reaches the exit, then the conveyor must be halted until the buffer is no longer full. (Note, then, that this system takes the form of a nonaccumulating conveyor which feeds into a capacitated accumulation buffer.) When the buffer becomes no longer full, the conveyor is then restarted, and things proceed from there.

Two (human) unloaders work at transferring bags from the buffer at the head of the conveyor to a storage area in the plane. The time required to accomplish this bag transfer is 1 +/- 0.5 time units per bag. Unloader access to the buffer is unrestricted. That is, the two unloaders can remove bags from the buffer simultaneously.

Build a model for the loaders and unloaders and the nonaccumulating conveyor and capacitated accumulation buffer which link them. Design the model so that a simulation performed with it will stop when a specified number of bags (e.g., 200 bags) have been transferred into the plane.

7.2 Presentation and Discussion of the Model

Figure 7 shows the listing of a GPSS model for the baggage transfer system. The Bag Loading Segment is shown in statements 15 through 27. Each of the two loaders is simulated by a Transaction. Time elapses while a loader Transaction grabs a bag, waits (if necessary) for exclusive access to the conveyor, places the bag on the conveyor, splits off another Transaction to represent the loaded bag, and then returns to grab the next bag if there are more bags left. Each loader updates a bags-left counter as it cycles. When there are no more bags left, the loader Transaction exits the model.

The Conveyor Flow Segment appears as statements 29 through 45 in Figure 7. The concept of a leader bag and the follow-the-leader approach, first discussed in Section 3.2 and already illustrated in models 3(a) and (b), is applied again in straightforward fashion in model 4. And as also previously discussed and illustrated in models 3(a) and (b), access to the conveyor entry is temporarily denied, or the ongoing placing of a bag onto the conveyor is temporarily suspended, by having the leader bag render the HEAD Facility (which simulates the conveyor entry) unavailable (statement 43, Figure 7) when it reaches the conveyor head and finds that it cannot drop into the buffer because it is already full (statement 44).

The Bag Unloading Segment (statements 47 through 57, Figure 7) is straightforward in terms of earlier examples and discussion, and will not be discussed further.

GPSS/H, Release 2, uses a double-precision floating point clock. This explains why floating point holding times at several ADVANCE Blocks in model 4 (statements 23 and 53, Figure 7) are valid.

When model 4 was executed under GPSS/H, Release 2, the simulation stopped at clock time 1144.8. The HEAD Facility was in an unavailable state (that is, the conveyor was halted) 2.3% of the time. The accumulation buffer at the head of the conveyor had an average content of 1.5 bags, and an average residence time of 8.6 time units per bag. The BELT User Chain had an average content of 1.2 bags, with an average residence time on the User Chain of 6.9 time units per bag. The User Chain's maximum content was 3, meaning there were never more than 4 bags on the conveyor (1 leader bag, and 3 bags waiting their turn to become the leader). Note that, per reasoning provided in the discussion of model 3(a), the conveyor in this problem has a capacity of 14 bags (1 being placed upon, 12 in transit, and 1 dropping off).

8. EXERCISES

In this section, exercises are presented to suggest ways in which the ideas applied in the preceding conveyor models can be reinforced and extended.

(1) Build a model for the Section 4.1.1 accumulating conveyor problem in your choice of a simulation language other than GPSS, e.g., MAP/1; SIMAN; SIMFACTORY; SIMSCRIPT; SLAM.

```
GPSS/H VAX/VMS RELEASE 0.96 (UG176)     15 AUG 1986    07:51:04     FILE: CONVNAC2.GPS

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

   1     1                           SIMULATE
   2     2                           REALLOCATE       COM,25000               REQUEST MORE MEMORY
   3     3                           RMULT            111111111,333333333     RN1, RN2 INDEPENDENT

   5     5                  ************************************************************************
   6     6                  *                                                                      *
   7     7                  *        GPSS/H MODEL OF A SIMPLE NON-ACCUMULATING CONVEYOR SYSTEM     *
   8     8                  *                                                                      *
   9     9                  *        *__________*                                   *__________*  *
  10    10                  *        * PRODUCER *--->|---------------------->|---* CONSUMER *  *
  11    11                  *        *__________*                                   *__________*  *
  12    12                  *                                                                      *
  13    13                  *        RULES:  1.  IF A WIDGET REACHES THE CONSUMER END OF THE       *
  14    14                  *                    CONVEYOR, AND THE CONSUMER IS BUSY, THE CONVEYOR  *
  15    15                  *                    IS HALTED.                                        *
  16    16                  *                2.  WIDGETS ARE HEAVY.  ACCORDINGLY, EACH TIME THE    *
  17    17                  *                    PRODUCER PLACES A WIDGET ON THE CONVEYOR, OR      *
  18    18                  *                    THE CONSUMER REMOVES ONE, THE CONVEYOR MUST BE    *
  19    19                  *                    STOPPED.                                          *
  20    20                  *                                                                      *
  21    21                  ************************************************************************

  23    23                  STOPPER STORAGE           2                 REASONS FOR STOPPING

  25    25                  PTIME   FUNCTION          RN1,C2            PRODUCER CYCLE TIME
  26    26                  0,15/1,25                                   20 + - 5
  27    27                  CTIME   FUNCTION          RN2,C2            CONSUMER CYCLE TIME
  28    28                  0,5/1,35                                    20 + - 15

  30    30                  ************************************************************************
  31    31                  *       PRODUCER SEGMENT                                               *
  32    32                  ************************************************************************

  34    34              1           GENERATE          ,,,1,,1PL         SINGLE PRODUCER
  35    35              2   PLOOP   ADVANCE           FN$PTIME          PRODUCTION CYCLE TIME
  36    36              3           QUEUE             ENTRY             MEASURE TIME TO GET ON
  37    37              4           ENTER             STOPPER           ANOTHER REASON TO STOP
  38    38              5           FUNAVAIL          LEADER            BE SURE THE CONVEYOR IS STOPPED
  39    39              6           ADVANCE           2                 PLACE WIDGET ON CONVEYOR
  40    40              7           LEAVE             STOPPER           ONE LESS REASON TO STOP
  41    41              8           GATE SE           STOPPER,*+2       SEE IF WE'RE RESTARTING
  42    42              9           FAVAIL            LEADER            RESTART THE CONVEYOR
  43    43             10           GATE SE           STOPPER           WAIT FOR RESTART
  44    44             11           DEPART            ENTRY             MEASURE DELTA T
  45    45             12           SPLIT             1,FLOW            OFFSPRING XACT MODELS FLOW
  46    46             13           TRANSFER          ,PLOOP            PRODUCER CYCLES ENDLESSLY

  48    48                  ************************************************************************
  49    49                  *       CONVEYOR FLOW SEGMENT                                          *
  50    50                  ************************************************************************

  52    52             14   FLOW    MARK              ,                        SAVE TIME ONTO CONVEYOR
  53    53             15           ASSIGN            SAVESUMH,_               SAVE TOTAL CONVEYOR
  54    54             15                             FUT$LEADER,PL            HALT TIME TO DATE
  55    55                  *                                                  ("FUT" => Facility
  56    56                  *                                                  Unavailable Time)
  57    57             16           LINK              BELT,FIFO,NEW1ST         NEW LEADER -> NEW1ST

  59    59             17   NEW1ST  SEIZE             LEADER                   NEW LEADER WIDGET
  60    60             18           ADVANCE           30-M1_                   TRAVEL TIME = 30 - TIME ON
  61    61             18                             +(FUT$LEADER-PL$SAVESUMH)   SO FAR + HALT TIME
  62    62                  *                                                     SO FAR WHILE ON
  63    63             19           ENTER             STOPPER                  ANOTHER REASON TO STOP
  64    64             20           RELEASE           LEADER                   WILL GET A NEW LEADER
  65    65             21           FUNAVAIL          LEADER                   STOP THE CONVEYOR
```

Figure 6: Model 3(b): A Nonaccumulating Conveyor which is Halted During Entries and Removals

```
GPSS/H VAX/VMS RELEASE 0.96 (UG176)     15 AUG 1986   07:51:04    FILE: CONVNAC2.GPS

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

  67    67                  *************************************************************************
  68    68                  *       CONSUMER SEGMENT                                                *
  69    69                  *************************************************************************

  71    71              22          SEIZE           CONSUMER
  72    72              23          ADVANCE         2                          UNLOADING TIME
  73    73              24          LEAVE           STOPPER                    ONE LESS REASON TO STOP
  74    74              25          GATE SE         STOPPER,*+2                SEE IF WE'RE RESTARTING
  75    75              26          FAVAIL          LEADER                     RESTART THE CONVEYOR
  76    76              27          UNLINK          BELT,NEW1ST,1              UNLEASH SUCCESSOR XACT
  77    77              28          ADVANCE         FN$CTIME                   UNLOADING CYCLE
  78    78              29          RELEASE         CONSUMER                   DONE
  79    79              30          TERMINATE       1                          WIDGET CONSUMED

  81    81                  *************************************************************************
  82    82                  *       RUN CONTROLS                                                    *
  83    83                  *************************************************************************

  85    85                          START           1000                       CONSUME 1000 WIDGETS
  86    86                          END
```

Figure 6: Model 3(b): (Continued)

(2) Repeat exercise (1), but for the Section 5.1 accumulating conveyor problem.

(3) Repeat exercise (1), but for the Section 6.1 nonaccumulating conveyor problem.

(4) Repeat exercise (1), but for the Section 7.1 nonaccumulating conveyor problem.

(5) Assume that the accumulating conveyor in the Section 4.1.1 problem is in operating condition for 1000 +/- 100 time units, then breaks down. After taking 50 +/- 20 time units to repair the conveyor, it then returns to operating condition for another 1000 +/- 100 time units, breaks down again, etc. Assume further that when the conveyor breaks down, ongoing production and consumption cycles continue to completion, but then are suspended until operation of the conveyor is later resumed. Show how to model this modified problem in the language of your choice.

(6) Repeat exercise (5), but for the Section 5.1 accumulating conveyor problem.

(7) Compose a problem similar to exercise (5) for the Section 6.1 nonaccumulating conveyor system, then show how to model the problem in the language of your choice.

(8) If you use GPSS but the FUT Standard Numerical Attribute is not offered by your GPSS implementation, show how models 3(b) and 4 can be modified so that they do not use FUT.

(9) In model 3(b), the producer does not begin the next production cycle until the most recently produced widget has been placed on the conveyor AND the conveyor has been restarted. Using the language of your choice, build a model for the system if it is assumed the producer begins the next production cycle as soon as the most recently produced widget has been placed on the conveyor, whether or not the conveyor has yet been restarted.

(Caution: when the producer has produced the next widget, it can't be placed on the conveyor unless the conveyor has operated long enough in the interim to move the preceding widget away from the entry point.)

(10) Using the Section 4.1.1 accumulating conveyor problem as a basis, compose a problem of your own along the following lines: Add a second producer to the system. Have the first and second producers produce different types of widgets, types which differ in their production cycle times and in the capacity of the conveyor for them. For example, the conveyor might have a capacity of 6 for one type widget, and a capacity of 11 for another type. Using a language of your choice, build a model for the problem you've composed.

(11) Build a model for the Section 4.1.1 accumulating conveyor problem in a language of your choice. Instead of using the "minimum travel time" approach described in Section 3.1, however, try an alternative approach in which the model keeps information about the conveyor state on an object-by-object, inch-by-inch basis. How easy or difficult do you find it to do this?

(12) Build a model for the Section 6.1.1 nonaccumulating conveyor problem in a language of your choice. Instead of

```
GPSS/H VAX/VMS RELEASE 0.96 (UG176)     15 AUG 1986   07:51:15     FILE: CONVNAC3.GPS

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

   1     1                           SIMULATE

   3     3                   ***********************************************************************
   4     4                   *       GPSS/H MODEL OF AN AIRPLANE BAGGAGE LOADING CONVEYOR SYSTEM     *
   5     5                   *                                                                       *
   6     6                   *       (A NONACCUMULATING CONVEYOR WITH A CAPACITATED ACCUMULATION     *
   7     7                   *       BUFFER AT THE RECEIVING END)                                    *
   8     8                   ***********************************************************************

  10    10                           INITIAL          X$BAGSLEFT,200           LOAD 200 BAGS
  11    11                           INITIAL          X$LOADERS,2              TWO LOADERS
  12    12                           STORAGE          S$UNLOADER,2             TWO UNLOADERS
  13    13                           STORAGE          S$BUFFER,3               3 BAG BUFFER AT TOP

  15    15                   ***********************************************************************
  16    16                   *       BAG LOADING SEGMENT                                             *
  17    17                   ***********************************************************************

  19    19                1          GENERATE         ,,,X$LOADERS,,1PL        N WORKERS LOADING BAGS
  20    20                2 LOAD     SAVEVALUE        BAGSLEFT-,1              ANOTHER BAG
  21    21                3          ADVANCE          10,4                     GRAB IT; TAKE TO BELT
  22    22                4          SEIZE            HEAD                     ACCESS TO LOADING END
  23    23                5          ADVANCE          1.0,0.5                  PLACE BAG ON BELT
  24    24                6          RELEASE          HEAD                     RELINQUISH CONTROL
  25    25                7          SPLIT            1,FLOW                   OFFSPRING XACT = BAG
  26    26                8          TEST E           X$BAGSLEFT,0,LOAD        LOAD N BAGS
  27    27                9          TERMINATE        0                        ALL BAGS ON THE WAY

  29    29                   ***********************************************************************
  30    30                   *       CONVEYOR FLOW SEGMENT                                           *
  31    31                   ***********************************************************************

  33    33               10 FLOW     MARK             ,                     SAVE TIME ONTO CONVEYOR
  34    34               11          ASSIGN           SAVESUMH,_            SAVE TOTAL CONVEYOR
  35    35               11                           FUT$HEAD,PL           HALT TIME TO DATE
  36    36                  *                                               ("FUT" => Facility
  37    37                  *                                                Unavailable Time)
  38    38               12          LINK             BELT,FIFO,NEW1ST      NEW LEADER -> NEW1ST

  40    40               13 NEW1ST   ADVANCE          12-M1_                TRAVEL TIME = 12 - TIME ON
  41    41               13                           +(FUT$HEAD-PL$SAVESUMH)     SO FAR + HALT TIME
  42    42                  *                                                     SO FAR WHILE ON
  43    43               14          FUNAVAIL         HEAD                  TENTATIVELY STOP CONVEYOR
  44    44               15          ENTER            BUFFER                TEST FOR ROOM IN BUFFER
  45    45               16          FAVAIL           HEAD                  RESTART CONVEYOR

  47    47                   ***********************************************************************
  48    48                   *       BAG UNLOADING SEGMENT                                           *
  49    49                   ***********************************************************************

  51    51               17          UNLINK           BELT,NEW1ST,1            UNLEASH SUCCESSOR XACT
  52    52               18          ENTER            UNLOADER                 ONE WORKER REQUIRED
  53    53               19          ADVANCE          1.0,0.5                  GRAB BAG
  54    54               20          LEAVE            BUFFER                   FREE UP BUFFER SPACE
  55    55               21          ADVANCE          10,8                     STOW THE BAG
  56    56               22          LEAVE            UNLOADER                 AVAIL FOR NEXT BAG
  57    57               23          TERMINATE        1                        THIS BAG IS LOADED

  59    59                   ***********************************************************************
  60    60                   *       RUN CONTROLS                                                    *
  61    61                   ***********************************************************************

  63    63                           START            X$BAGSLEFT               LOAD N BAGS
  64    64                           END
```

Figure 7: Model 4: An Airplane Baggage Loading Conveyor System

using the "follow-the-leader" approach described in Section 3.2, however, try an alternative approach in which the model keeps information about the conveyor state on an object-by-object, inch-by-inch basis. How easy or difficult do you find it to do this?

(13) Read the description of the conveyor system in Henriksen (1986). Build a model for that system in a language of your choice. If you build the model in GPSS, compare your model with the GPSS model which Henriksen gives.

9. CONCLUSIONS

Some of the characteristics of conveyors have been described, and two relatively simple techniques for modeling the movement of objects on conveyors have been introduced and discussed. The "minimum travel time" technique is appropriate for modeling accumulating conveyors, and the "follow-the-leader" technique is applicable to the modeling of nonaccumulating conveyors. Applications of these two techniques have been illustrated in a collection of seven models corresponding to four conveyor systems and variations on them. Although the illustrative models have been implemented in GPSS/H, the techniques themselves are language-independent, and so have broad-based applicability. Demonstration of this fact is called for in a series of exercises challenging the reader to build models in alternative languages both for the illustrative problems used here, and for other problems which are given as well.

REFERENCES

Bobillier, P. A., Kahan, B. C., and Probst, A. R. (1976). Simulation with GPSS and GPSS/V. Prentice-Hall, Englewood Cliffs, New Jersey.

Gordon, G. (1975). The Application of GPSS V to Discrete Systems Simulation. Prentice-Hall, Englewood Cliffs, New Jersey.

Henriksen, J. O. (1981). GPSS - Finding the Appropriate World-View. In: Proceedings of the 1981 Winter Simulation Conference (T. I. Oren, C. M. Delfosse, and C. M. Shub, eds.). Society for Computer Simulation, San Diego, California.

Henriksen, J. O. (1986). You Can't Beat the Clock: Studies in Problem Solving. In: Proceedings of the 1986 Winter Simulation Conference (S. D. Roberts and J. O. Henriksen, eds.). Society for Computer Simulation, San Diego, California.

Henriksen, J. O., and Crain, R. C. (1986). GPSS/H User's Manual, 3rd ed. Wolverine Software Corporation, Annandale, Virginia.

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

Schriber, T. J. (1986). Introduction to GPSS. In: Proceedings of the 1986 Winter Simulation Conference (S. D. Roberts and J. O. Henriksen, eds.). Society for Computer Simulation, San Diego, California.

AUTHORS' BIOGRAPHIES

JAMES O. HENRIKSEN is the president of Wolverine Software Corporation, located in Annandale, Virginia (a suburb of Washington, D.C.) Wolverine Software was founded in 1976 to develop and market GPSS/H, a state-of-the-art version of the GPSS language. Since its introduction in 1977, GPSS/H has gained wide acceptance in both industry and academia. Mr. Henriksen is an Adjunct Professor in the Computer Science Department of the Virginia Polytechnic Institute and State University. He teaches courses in simulation and compiler construction at the university's Northern Virginia Graduate Center. Prior to forming Wolverine Software, he worked for CACI, Inc., where he served as project manager for development of the Univac 1100 Series version of Simscript II.5. Mr. Henriksen is a frequent contributor to the literature on simulation. He has given invited presentations at the Winter Simulation Conference, the Summer Simulation Conference, and at the Annual Simulation Symposium. Mr. Henriksen is a member of ACM, SIGSIM, SCS, the IEEE Computer Society, ORSA, and SME.

James O. Henriksen
Wolverine Software Corporation
7630 Little River Turnpike - Suite 208
Annandale, VA 22003-2653
(703) 750-3910

THOMAS J. SCHRIBER is a professor in the Graduate School of Business at The University of Michigan. He holds B.S. (University of Notre Dame) and M.S.E., M.A., and Ph.D. degrees (University of Michigan). Professor Schriber teaches, does research, and consults in the area of discrete-event simulation. He has published over 25 articles, and has authored or edited nine books.

Thomas J. Schriber
Graduate School of Business Administration
The University of Michigan
Ann Arbor, MI 48109, U.S.A.

(313) 764-1398