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

SIMULATING SCHEDULE RECOVERY STRATEGIES IN MANUFACTURING ASSEMBLY OPERATIONS

Patrick J. Starr and Douglas S. Skrien
Department of Mechanical Engineering
University of Minnesota
111 Church Street S.E.
Minneapolis, Minnesota 55455

Robert M. Meyer
Materials and Processing Department
University of Wisconsin, Stout
215 Fryklund Hall
Menomonie, Wisconsin 54751

ABSTRACT

An animated simulation is used to evaluate two workforce management options in response to work stoppages in an electronic assembly workcell. An industrial based workcell consisting of 15 workstations is modeled in the SIMPLE_1 environment. A work stoppage is introduced at a station and the throughput time for each assembly is recorded. Two strategies are compared: one has five worker groups, each serving a set of stations, and the other allows workers to move between groups. Steady state performance under each strategy is identical, but they differ dramatically in the ability to restore throughput time to its pre-stoppage values. The results quantitatively show a benefit of cross-trained workers.

1. INTRODUCTION

This work reports on preliminary results of an investigation into identifying and evaluating various workforce management options in response to flow interruptions in an electronics assembly workcell. Manufacturing is rarely steady state. There is a myriad of causes of work stoppages and load variations, such as down workstations, part shortages, urgent prototype work, illness of skilled personnel, etc. All of these cause either a short term reduction of capacity at selected workstations or sudden (unscheduled) shifts in loading. The management goal is to "recover the schedule", that is, to make short term capacity adjustments to minimize delinquency to due dates.

The paper presents a workcell model having 15 separate workstations. All stations have a maximum output capacity, but can be operated below capacity by reducing the workforce. The model is rooted in an industrial application, but the numerical data have been altered for proprietary reasons. The actual workcell is in an evolutionary process, since the products are being modified along with the production means. In this mode of operation, "exception events" which cause either a work stoppage or a sudden load at a workstation are commonplace. Thus, it is important to (1) investigate the parameters of such stoppages, (2) identify possible responses and (3) evaluate such responses.

For certain stoppages, one response is to maintain the system input and then make dynamic adjustments in station capacity by transferring workers among stations. To do so, workers must be cross-trained to perform the tasks at various stations, and strategies for transfers must be developed, along with appropriate performance measures. Cross training is a popular issue, since with the interest in JIT (Just in Time) philosophy of operation, flexible capacity, rather than inventory, is considered a suitable means of accomodating temporary flow interruptions. Since a work stoppage will increase throughput time for a number of assemblies, a suitable performance measure is the time it takes for the throughput time per assembly to be restored to its pre-stoppage value, for then scheduled arrivals will meet their due dates. This work reports preliminary results from exploring two strategies for schedule recovery using cross-trained workers. The strategies are compared using the time to restore throughput time. The operation of the workcell is simulated using the SIMPLE_1 environment (Cobbin, 1986 A, B). This environment was selected to explore how a moderately priced PC based environment could accomodate some intricacies of workcell operation and produce an animated output.

In the following, the operation of the workcell is described and some SIMPLE_1 code fragments are shown. Results for a base level operation are discussed, followed by a description of the work stoppage and recovery approaches. Finally we describe our preliminary findings.

2. WORKCELL DESCRIPTION

Figure 1 shows a schematic of the workstations and the routes followed by an assembly. Numbers in circles identify the stations, while those in parenthesis are nominal processing times per batch. Symbols S or M indicate a single or multiple server workstation. The circular arc attached to certain stations signifies that when an assembly leaves those stations, it requires a brief cleaning at station 5 before going to the subsequent station. The operator at a station having the arc, carries the assembly to station 5, performs the cleaning and takes the assembly to the next station on the route. The operator may then return. Station 5 has a short cleaning time, but as it is frequently used, there is often a queue. All stations utilize a batch size of one except station 6, which must use batches of five. Assemblies are brought to station 6 individually, but when a batch of five accumulates, they are placed on a fixture by an operator who starts the operation if the machine is available. Once started, the operator is no longer needed. Assemblies are cycled through stations 6, 7 and 8 twice, where the dotted lines signify the first pass. Stations 13 and 14 are inspection and rework, respectively, and as this product and process are evolving, most assemblies require some rework.

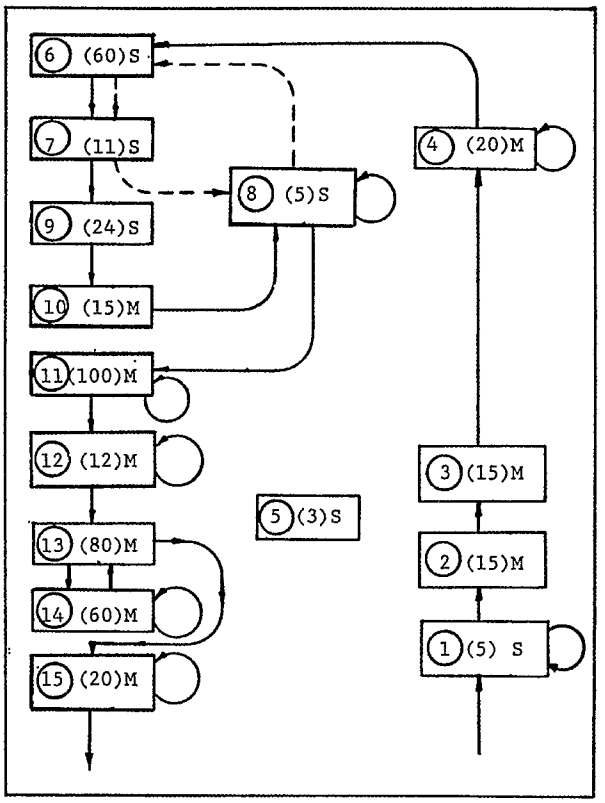

FIGURE 1. WORKCELL STATIONS AND ROUTING OF ASSEMBLIES

There is interest in a JIT (Just in Time) type of operation, which is relected in a desire to provide uniform loading and cross-trained worker groups. For this model, the input is uniform with one assembly arriving every 30 minutes, and a workforce of 20 which is broken up into five groups as follows:

GROUP #1 SERVES STATIONS 1, 2, 3, 4, 6, 7, 8, AND 10 (WORKER_Q1)
GROUP #2 SERVES STATION 9 (WORKER_Q2)
GROUP #3 SERVES STATIONS 11, 12, AND 14 (WORKER_Q3)
GROUP #4 SERVES STATION 13 (WORKER_Q4)
GROUP #5 SERVES STATION 15 (WORKER_Q6)

With this workforce grouping, the only cross-training is within groups 1 and 3.

3. USE OF SIMPLE_1

The operation of the workcell was studied by simulating its behavior in the Simple_1 environment on an IBM PC-AT. We chose SIMPLE_1 to become familiar with a moderately priced ($895.00) package that promised broad capabilities, including output animation. The following describes a few code fragments from the model code. The intention is not to show the capabilities of SIMPLE_1 for that can be found elsewhere. Rather, we simply show the use of the language to describe some situations that are not illustrated in the examples in the references.

Figure 2 shows the code fragment that describes the operation of the very popular station 5. Entity groups are queued at WAIT_STA5. Each group consists of one SMT_Board entity and one WORKER entity. VAPCL refers to an activity having a normal distribution with μ, σ and seed indicated. The conditions statement checks whether the VAPCL activity is free, and if so, sends the entity group from WAIT_STA5 to VAPCL for processing. When completed, the SMT_BOARD entity and the worker split, with the worker going back to its worker group and the SMT_BOARD going to its next queue on the route. This is managed by the BRANCH statement which utilizes attribute (2) of the SMT_BOARD entity. This attribute value was set as the entity left the previous station. For example, if the prior station was 11, the attribute value would be 5 and control would branch to the ROUTE_12 statement. The worker would be split off and sent to WORKER_Q3 and the SMT_BOARD would join the queue at WAIT_STA12.

Figure 3 shows the fragment for operations at station 6. The SMT_BOARD entities are in queue at WAIT_STA6. Statement BAKE105 is the activity where a batch of five is processed. When the conditions of an idle activity at BAKE105, the number of SMT_BOARDs in WAIT_STA6 queue is five or more, and a worker from group 1 is available, the worker and five boards are joined into an entity group and sent to SETUP_STA6 where a fixturing activity occurs. Following that, the worker is split off and returns to WORKER_Q1 while the batch of five SMT_BOARDS continues to the BAKE105 activity. Then, the batch is broken up by successive execution of the SPLIT and BRANCH statements at UNCLUMP.

SIMPLE_1 allows the user to design interactive screen menus, and three were developed to (1) input the means and std. dev. for processing time at each process, (2) initialize queue lengths at each station, which also included the choice of first or second pass initial queues at selected stations, and (3) select workforce size for each group. In addition, an animation of the workcell was developed. Though it loses much impact when stationary and in black and white, Figure 4 shows the screen layout. Assembly flow starts at the lower left and concludes at the upper left. Station numbers are shown in the upper part of each station block which is formed in asterisks. Routes to station 5 are indicated by arrows leaving the appropriate stations. The number of idle workers in each group is shown as "avail wkrs", and are color coded. In the animation, queues of waiting entities can form on the input side of each station while the number of items in process is shown by a number within each station which is color coded to the worker group. The SIMPLE_1 code for the model had 675 lines, which were distributed among program tasks as follows:

a. Interactively setting processing times and initial queue lengths: 200 lines
b. Interactively setting workforce size and distribution: 100 lines
c. Creating the output animation: 200 lines
d. Describing workcell operations: 175 lines.

Thus, it is seen that the code needed to describe the simulated operations is relatively brief,

```
WAIT_STA5       QUEUE,FIFO;
                CONDITIONS,NUM(VAPCL)<1,
                WAIT_STA5,,VAPCL;
VAPCL           ACTIVITY NORMAL(VAPCL_TIME(1),VAPCL_TIME(2),1);
                BRANCH SMT_BOARD(2)=1,ROUTE_1:
                       SMT_BOARD(2)=2,ROUTE_6A:
                       SMT_BOARD(2)=3,ROUTE_6B:
                       SMT_BOARD(2)=4,ROUTE_11:
                       SMT_BOARD(2)=5,ROUTE_12:
                       SMT_BOARD(2)=6,ROUTE_13A:
                       SMT_BOARD(2)=7,ROUTE_13B:
                       1.0,ROUTE_16;
          ROUTE_1    SPLIT:WORKER,1,WORKER_01;BRANCH,WAIT_STA1;
          ROUTE_6A   SPLIT:WORKER,1,WORKER_01;BRANCH,WAIT_STA6;
          ROUTE_6B   SPLIT:WORKER,1,WORKER_01;BRANCH,WAIT_STA6;
          ROUTE_11   SPLIT:WORKER,1,WORKER_01;BRANCH,WAIT_STA11;
          ROUTE_12   SPLIT:WORKER,1,WORKER_03;BRANCH,WAIT_STA12;
          ROUTE_13A SPLIT:WORKER,1,WORKER_03;BRANCH,WAIT_STA13;
          ROUTE_13B SPLIT:WORKER,1,WORKER_03;BRANCH,WAIT_STA13;
          ROUTE_16   SPLIT:WORKER,1,WORKER_06;BRANCH,LAST_OP;
```

FIGURE 2. CODE FRAGMENT FOR STATION 5

```
WAIT_STA6       QUEUE,FIFO;
    CONDITIONS,NUM(BAKE105)<1 AND NUM(WAIT_STA6)>4 AND NUM(SETUP_STA6)<1,WORKER_
01,,SETUP_STA6:
                WAIT_STA6,,SETUP_STA6:WAIT_STA6,,SETUP_STA6:
                WAIT_STA6,,SETUP_STA6:WAIT_STA6,,SETUP_STA6:
                WAIT_STA6,,SETUP_STA6;
SETUP_STA6      ACTIVITY NORMAL(SETUP6(1),SETUP6(2),1);
                SPLIT:WORKER,1,WORKER_01;
BAKE105         ACTIVITY NORMAL(BAKE105_TIME(1),BAKE105_TIME(2),1);
 UNCLUMP         SPLIT:SMT_BOARD,1,WAIT_STA7;BRANCH,UNCLUMP;
```

FIGURE 3. CODE FRAGMENT FOR STATION 6

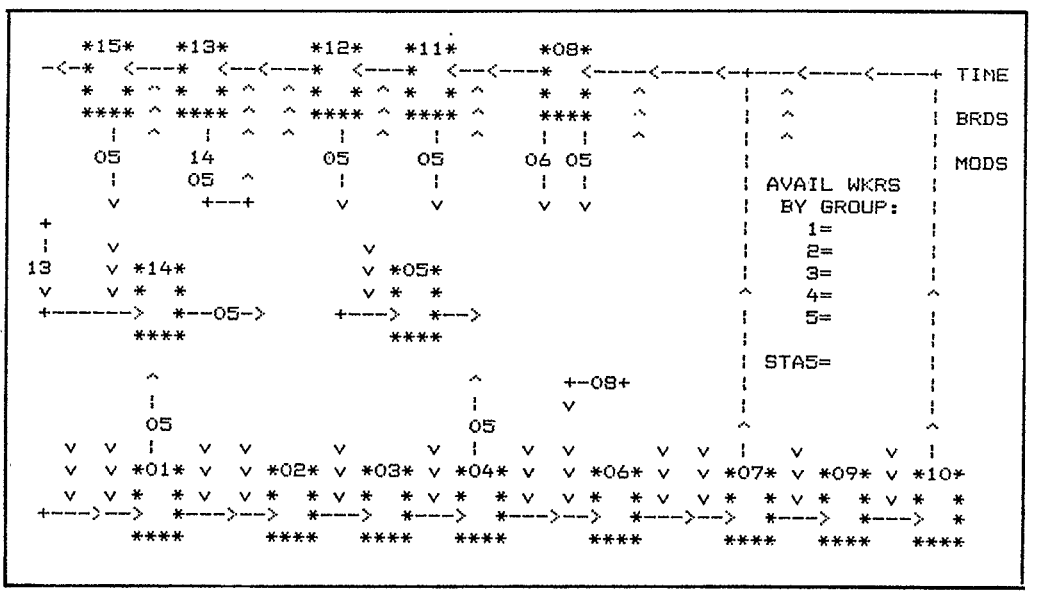

FIGURE 4. ANIMATION SCREEN LAYOUT

while most of the effort is in the busy work of creating the interactive portion and the animation. The user is relieved from statistics collection and reporting, as it is automatically done in SIMPLE_1. It also includes a convenient editor which allows quick changes during program development. Of course, the above numbers of lines is wholly dependent upon our "learning while doing", but we suspect that as proficiency grows, the relative proportion of code for simulated operations vs. input and animation would reamin the same.

4. SIMULATED RESULTS: FIVE WORKER GROUPS

Many experiments were performed to select the worforce size and distribution among the five groups of workers. Each group was represented internally as a queue, for which statistics were accumulated. Simulations were run for 4800 minutes of simulated time (one week, two shifts). Steady state was judged qualitatively when the throughput time per assembly fluctuated around a steady average value. Initial queues of assemblies were selected iteratively from one run to the next as the sum of the average number in process plus the average number waiting at each station (rounded down). A base run was developed where the worker groups and subsequent average number idle in each group were found as follows:

Group	Size	Average Idle
1	5	.94
2	1	.20
3	7	.77
4	6	.77
5	1	.20
	20	2.88

The average throughput time per assembly was 801 minutes, with a std. dev. of 29 minutes. The average number of assemblies waiting at most stations was less than one, indicating a "near JIT" operation. This was judged to be a good system and was used as the baseline. It required 4 minutes to compile on the IBM PC-AT and 19 minutes to run, with the screen being updated every 5 minutes of simulated time.

6. SCHEDULE RECOVERY

There are many possible work stoppages and many management reactions. We investigate the situation where a stoppage occurs at station 9, which is roughly midway through the system and is a single server station that has a technologically limited capacity. This station has a newly developed robot, and is expected to require adjustments on a regular, but random, basis. Adjustments will take between two and three hours. We explore the option of keeping the line running and examine the effect upon throughput time. The utilization for station 9 in the base line was 0.8, which indicates there is some excess capacity that could be exploited in the short term. However, it is most important that no new bottlenecks are created after station 9 when the stoppage is over. That is, since a queue will have developed at station 9 during the stoppage, it is crucial that downstream stations have sufficient capacity to accomodate the sudden load without introducing additional delays. One way to accomplish this is to temporarily transfer some workers from stations ahead of nine (Group 1) to stations following it (Groups 3 and 4). This implies that further cross-training would be needed, and so it is necessary to identify a quantitative means to support such a decision. The hoped-for effect would be a faster recovery of the schedule, so it is appropriate to measure the time it takes for the throughput time per assembly to return to its pre-stoppage levels. This can also be measured in the number of late assemblies, since the input is uniform.

To investigate a strategy of shifting workers among stations outside of their own group, the most simple control strategy was employed: replace five worker groups with one cross-trained group, and send workers to whatever station needs a server on a FCFS basis. This is not an intelligent scheme, but it does utilize some information on the needs at stations and it is easily implemented! Our intention was to first explore simplistic schemes and then increase their complexity.

The baseline model was revised to have one group of 20 cross-trained workers replacing the five groups which totalled 20. Maximum capacities at each station were unchanged. The revised model was run to identify steady state characteristics. The average number of idle workers was 2.75 which compares favorably with the sum across the five groups of the original model which gave 2.88. The average throughput time was 788 minutes which was within 2% of that of the original model. Its std. dev. was 28 minutes. We concluded that the base runs of each were sufficiently close so as not to confound our finding in schedule recovery. This could have posed a real problem, since it may have been possible for the single pool to far exceed the performance of the five worker group model. This did not occur, probably due to the many single server stations which will bottle things up no matter how workers are pooled.

The work stoppage at station 9 was then implemented in both models for a 180 minute delay. Figure 5 shows the changes in throughput time for the assemblies following the stoppage. The horizontal axis shows assemblies in order of arrival to the system, which also corresponds to time, since the arrivals are uniform at 30 minute intervals. Each plot starts from its respective steady state value and the one std. dev. bands are shown. The plot is for one run, but it is typical. It shows a dramatic difference in the ability to recover the throughput time. If an assembly is defined as "late" when its throughput time falls outside of the one std. dev. limit, the figure shows that the five worker group model gives roughly twice as many late boards as the single pool model. Furthermore the pattern of recovery is very different. Throughput time for the five group model continues to increase following the stoppage, and peaks after many assemblies, while the single group model peaks immediately and starts its recovery as soon as the stoppage is removed.

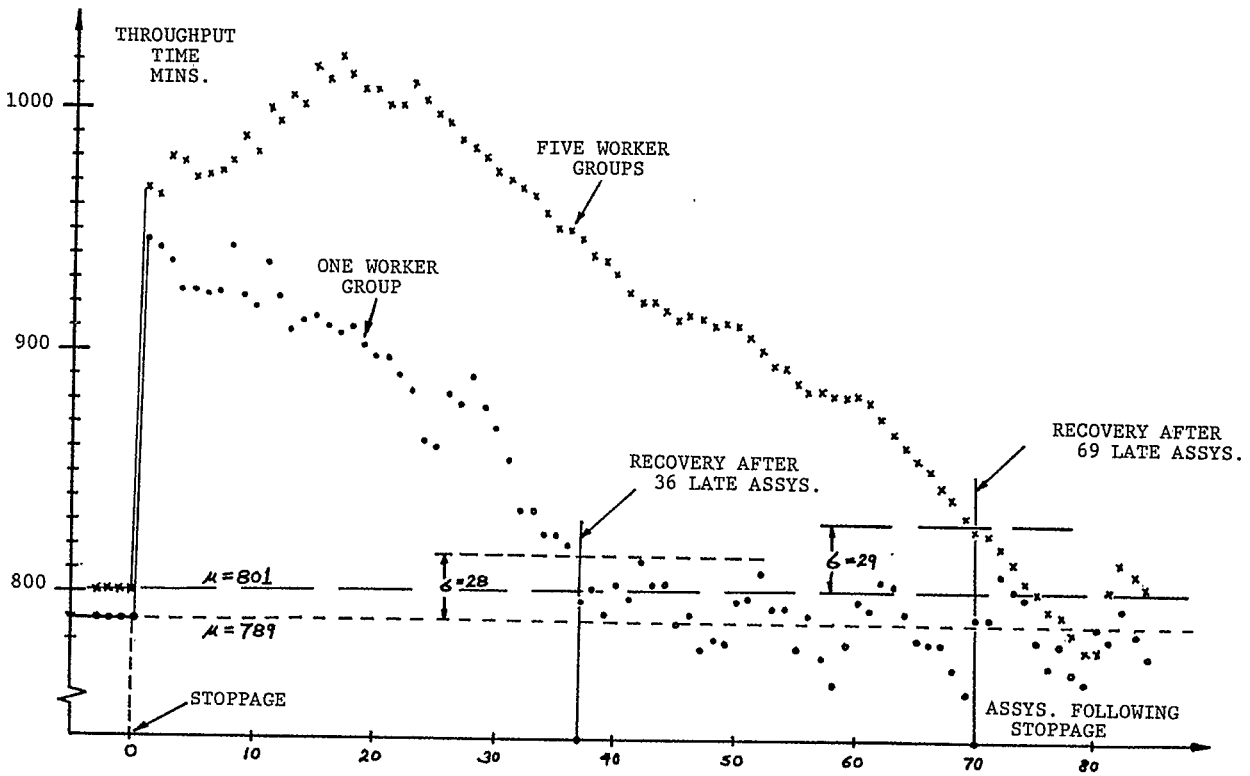

FIGURE 5. SIMULATED RECOVERY PATTERNS

7. CONCLUSIONS

These results show the benefits of cross-training in quantitative terms for a workstoppage having a duration of roughly 20% of the throughput time. This is a very specific situation, but it is an encouraging initial result and raises many interesting questions, such as the effect of "pooling" which is well known in queueing theory (Taha, 1982). Pooling usually improves steady state performance. However, as these results show, the creation of a single pool had a negligible effect upon the steady state behavior, but dramatically affected the transient response. We are currently investigating more complex control schemes along with determining the effects of stoppage location and duration.

REFERENCES

Cobbin, P. (1986A), SIMPLE_1 User's Guide and Reference Manual, Version 1.1, Sierra Simulations and Software, Campbell, California.

Cobbin, P. (1986B) SIMPLE_1: A Simulation Environment for the IBM PC, Modeling and Simulation on Microcomputers, Claude C. Barnett, Ed., Society for Computer Simulation, La Jolla, California, pp. 243 - 248.

Taha, H.A. (1982), Operations Research: An Introduction, 3rd. Edition, Chapter 16, Macmillan Publishing Company, Inc., New York.

AUTHOR'S BIOGRAPHIES

PATRICK J. STARR is an Associate Professor in the IEOR Division of the Mechanical Engineering Department at the University of Minnesota. He has developed and taught courses in many topics including Operations Research, System Dynamics, Statistics, Global Modeling, Technology Assessment, Design, and Cellular Manufacturing. He received the Institute of Technology Distinguished Teaching Award in 1973 and was elected Outstanding I.E. Educator by the Minnesota Student Chapter of I.I.E. in 1982. His consulting and research has included model building in a variety of settings, from the definition and analysis of solid waste management options to the flow of circuit boards through various processes. He has published in a variety of journals, and is currently doing work in sensitivity analysis of large scale systems, manufacturing simulation and expert systems.

Patrick J. Starr
Mechanical Engineering Department
University of Minnesota
111 Church Street S.E.
Minneapolis, Minnesota 55455
(612) 625-2315

DOUGLAS S. SKRIEN is a graduate student in Industrial Engineering at the University of Minnesota. He received a B.A. in Physics and Mathematics from St. Olaf College in 1984. His

current research interests include Modeling and analysis of Manufacturing Systems, and the study of Transient Behavior in these systems using Computer Simulation. He is a student member of I.I.E.

Douglas S. Skrien
Mechanical Engineering Department
University of Minnesota
111 Church Street S.E.
Minneapolis, Minnesota 55455
(612) 625-4344

ROBERT M. MEYER is a lecturer at the University of Wisconsin-Stout where he teaches coursework in Computer Integrated Manufacturing and Robotics. He is pursuing his Ph.D. in Industrial Engineering at the University of Minnesota and has received a B.S. in Industrial Education from the University of Wisconsin-Stout (1980) and an M.S. in Management Technology from the University of Wisconsin-Stout (1983). His current research interests include Management of Transient Behavior in the Industrial Environment using Computer Simulation. He is an associate member of CASA/SME, Machine Visions Group (SME), the Robotic Industries Association, and the Society of Manufacturing Engineers.

Robert M. Meyer
Materials and Processing Department
University of Wisconsin-Stout
215 Fryklund Hall
Menomonie, Wisconsin 54751
(715) 232-1259