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

MODELING TOTE STACKER OPERATION AS A WIP STORAGE DEVICE

PHILIP COBBIN
SIERRA SIMULATIONS & SOFTWARE
303 ESTHER AVENUE
CAMPBELL, CALIFORNIA 95008

ABSTRACT

This paper summarizes the modeling of a tote stacker device in support of manufacturing operations on a micro-computer using the SIMPLE_1 simulation environment. Animation of the model was obtained utilizing high level language concepts for animation of system operation. The results presented are an overview of the modeling techniques and statistical results that can be obtained for stacker crane type material handling and storage devices. A serial process feeding automated testing devices is used for illustrative purposes. The dynamic behavior of the system is illustrated visually in real time with the graphic system display augmented by collection of statistics on system performance.

INTRODUCTION

Crane type materials handling equipment and associated control systems have been in existence for some time. System designers have recognized the need to model the performance of such systems yet simulation of materials handling systems has proven to be a difficult undertaking in many instances. Much of the difficulty encountered in simulation modeling of materials handling equipment in general is due to logistics details that overrun the capabilities of simulation software or the user, or both. Recent developments in software have tended to attack the material handling modeling problem by building into simulation software canned constructs to represent elements of a generic materials handling hardware/software world view. An alternative approach to supporting material handling systems modeling is to provide general purpose modeling and programming features which facilitate modeling such complex systems. The SIMPLE_1 modeling environment contains features particularly useful in modeling material handling and manufacturing systems. This paper presents a model of a tote stacker type crane system in support of an idealized assembly and test process to illustrate the ability to model complex materials handling systems on a micro-computer with animation.

Simulation of material handling systems can be thought of as an embedded modeling process where a model of the material handling system is coupled to a model of the production process. Accordingly, material handling system models tend to exhibit extra run time due to the increased event calendar traffic required to handle production process and material handling system events.

In modeling the operation of a tote stacker type crane one can view the crane as a key resource which interacts with multiple work centers located throughout the system. What tends to complicate the modeling of such a system is the necessity to keep track of where every thing is, what state the crane/system is in, and merging/separation of entities in the system with the crane. The logic employed to manage usage of the crane has a significant bearing on the overall performance of the system. An element to consider in model construction is the evaluation of crane routing strategy to manage the overall production process.

The approach taken here is to tailor a model of the cranes operation to the nuances of a production process. The crane serves as an effective production control device which regulates the flow of materials throughout the system. The decision making process for routing the crane among competing task alternatives is an integral part of the modeling process.

SYSTEM DESCRIPTION

Initial stacker crane installations were to provide a automated storage and retrieval function in warehousing. Recently, tote stackers have been employed to support material transport and work in process storage to manufacturing operations directly. In the electronics industry functional testing of electronic devices tends to be highly automated and in one case the testing equipment was located in storage cells normally used for WIP storage. The hypothetical process modeled for this paper involves two sequential production steps which are performed at separate work stations and a functional test of the product. There are 20 functional testers located in WIP storage cells which are automatically plugged into by the tote stacker. Figure 1 is a schematic of the production process. The loading of the system simulated was periodic with work introduced once a day during the graveyard shift. The first two work stations were used only during first shift and the 20 testers were available 24 hours a day and are automatically plugged and un-plugged using the stacker crane. The Removal and introduction of new materials to the system occurs only once a day.

The shop loading scheme modeled involved introducing new work at the start of the third shift and removing completed items only during the first shift. The loading scheme employed illustrates the ability to level workload on the crane by shifting tasks to an off demand period. The shop loading scheme and parameters of the production process introduce periodic demands for work in process (WIP) storage.

The tote stacker delivers and picks up materials from works stations at locations designated for each work station. The input and output interfaces between work station and tote stacker have a fixed storage capacity of three units. When storage is not available at a work station the stacker stores work in process inventory items into general storage with storage cells allocated in FIFO manner over time to balance usage. When the stacker delivers an item to one of the functional testers the crane goes to the specified storage cell and "plugs in" the unit. In a similar fashion the crane dis-connects a completed device from the tester and routes it to a packing station or to storage.

The work stations are used to produce an entire days supply of parts in one shift. The work stations are staffed during first shift and work off the inventory loaded into storage during the previous third shift by the crane. During the first shift the work stations over run the capacity of the functional testers and work is banked into storage. During the second and third shifts the automated functional testers work off the inventory. Table 1 is a summary of process times and yield parameters used in the model. Note that the tote stacker type system has an inherent advantage in this type of situation being able to use WIP storage racks for different purposes throughout the day. During the animation of the model it becomes readily apparent that the storage cells are used differently during each shift. Statistics collected during the simulation verify these observations.

Part processing through the model involves routing PART entities to work station sub-models using the crane entity. The PART entities are combined with the CRANE entity to form a group which travel together through the network model representing the material transport phase. When parts are delivered to a work station, storage location, etc. the part is split from the crane. When the part/crane entity group is split both entities maintain their original attribute information and go their separate ways.

The introduction of parts to the system involves creation of a group of part entities once a day. The group, or lot, of 25 parts are created and introduced into the system via the code & network diagram fragment illustrated in Figure 2. The code fragment represents creation of the parts as a lot, setting the create time attribute and splitting the lot up into individual entities. In addition a CHART block is used to update the screen using ASCII character number 127 to represent the number of parts introduced into the system. The SHOW block employed in the fragment is used to display in numeric format a count of total number of parts introduced into the system.

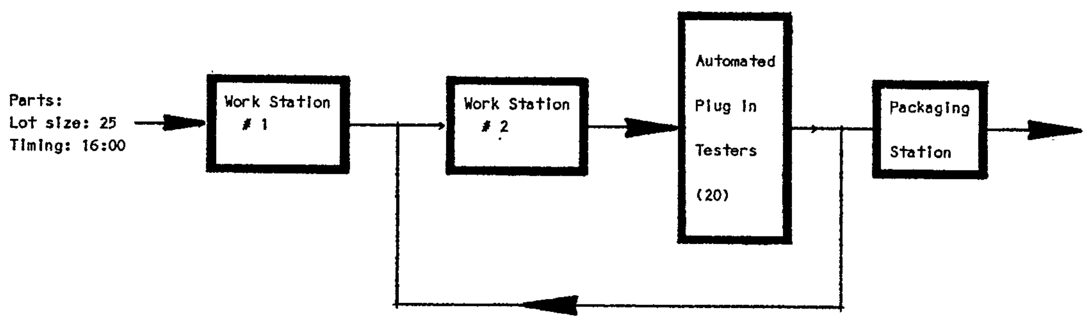


Figure 1 - Production Process

ITEM	Mean	Std. Dev.	Distribution
Work Station #1	15.0	2.0	Lognormal
Work Station #2	20.0	2.5	Lognormal
Functional Test (Yield: 80% pass, 20% fail)	960		Constant (pass) Uniform (fail)

Table 1 - Process Parameters

MODEL BUILDING

The production process is relatively simple in this case, and readily modeled. Model development revolved around the operation of the crane. The code was developed in stages with the basic operation of the crane modeled first. In the initial code the crane merely loaded items into storage and later retrieved them. From the initial crane model a series of embellishments were made to add in the production process. Production items were modeled as an entity of type PART which were introduced to the system in lots of 25 once a day. Part entities required two attributes to maintain the time the part entered the system and a part destination code.

The part entity is routed to the the WaitEntry labelled Queue where it resides until grouped with the crane entity. When the crane is assigned to pick up and incoming part a CONDITIONS block in the model controls grouping of the crane and part entities and routing them to the appropriate block in the model. When the part arrives to the first work station it is processed through the network and code fragment depicted in Figure 3. Two pairs of SET & CLONE blocks are used in the network fragment. The SET block establishes display values and the CLONE block creates a copy of the part. The cloned part is routed to a section of the model which controls updating the screen to display the current state of the work station. Two sets of these blocks are used to update the screen at the start and completion of the work station operation cycle. At a block labeled Show_Station, CHART blocks are used to display characters on the screen to represent the state of the work stations. Three CHART blocks are employed to update the number of parts in service, and waiting in the work stations input and output queues.

The input and output queues employed by the work stations provide an interface between work station operation and the material handling activities associated with the tote stacker. Once an item is deposited by the crane into a stations input queue the part is processed independent of the crane until

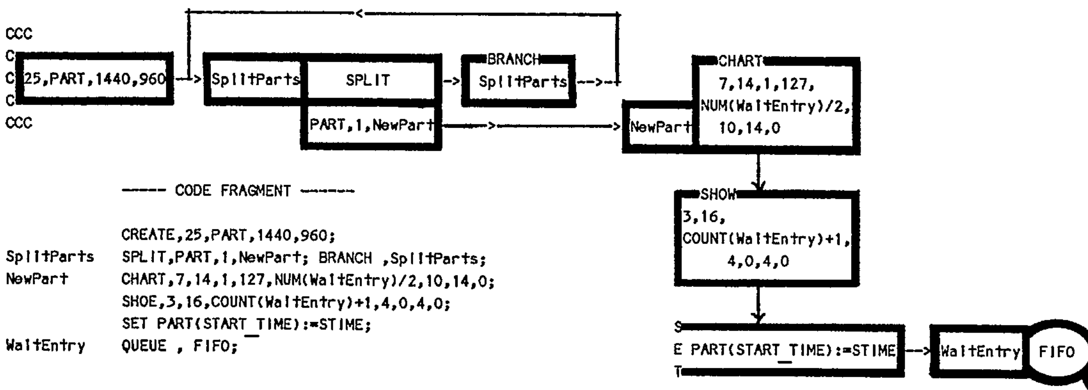
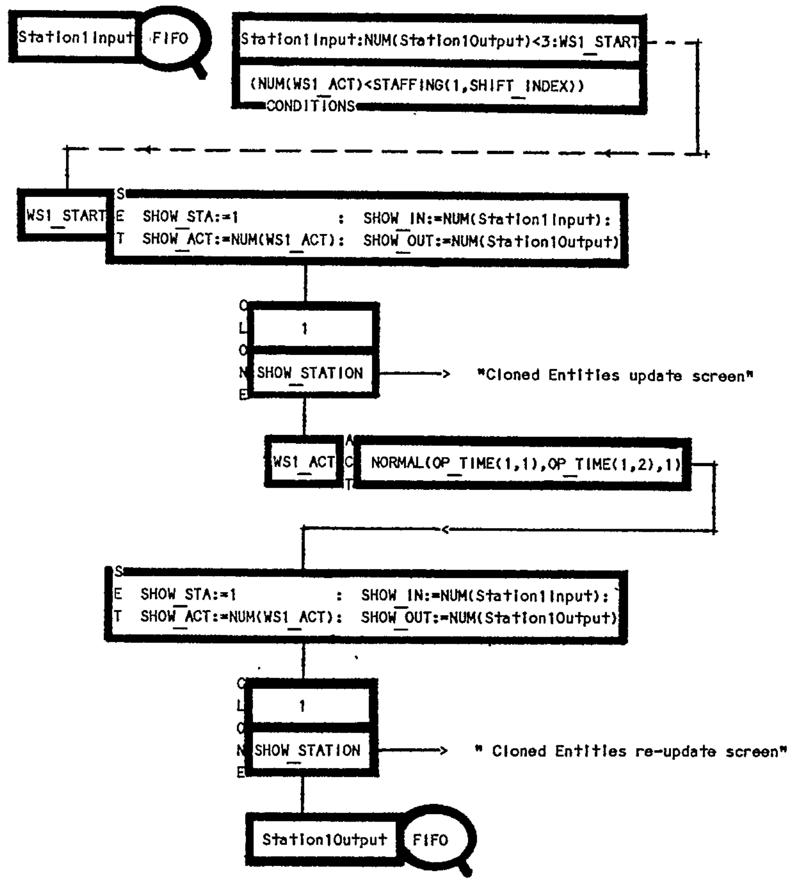


Figure 2 - Network fragment to load parts

TASK	Decision Rule	Priority
Storage to Exit	NUM(Exit_Q) < 10	1
Move from tester	NUM(EmptyCells) > 0 OR NUM(Exit_Q) < 10 AND NUM(FinishedTest) > 0	2
Move from Sta #2	NUM(EmptyCells) > 0 OR NUM(IdleTesters) > 0 AND NUM(Station2Output) > 0	3
Storage to Tester	NUM(TesterWip) > 0 AND NUM(IdleTesters) > 0	4
Storage to Sta #2	NUM(Station2Input) < 2	5
Storage to Sta #1	NUM(Station1Input) < 2	6
Move from Sta #1	NUM(EmptyCells) > 0 OR NUM(Station2Input) < 3 AND NUM(Station1Output) > 0	7
Move from Input Q	NUM(EmptyCells) > 0 AND NUM(WaitEntry) > 0	8

Table 2 - Crane allocation rules




```
{----- WS1 CODE  INPUT QUEUE...SERVICE ACTIVITY...OUTPUT QUEUE -----}

Station1Wip    QUEUE,FIFO;
Station1Input  QUEUE, FIFO; {AT WORKSTATION}
               CONDITIONS,(NUM(WS1_ACT)<STAFFING(1,SHIFT_INDEX)),
                      Station1Input,NUM(Station1Output)<3,WS1_START;
WS1_START      SET SHOW_STA:=1: SHOW_IN:=NUM(Station1Input): SHOW_ACT:=NUM(WS1_ACT):
                   SHOW_OUT:=NUM(Station1Output);
               CLONE,1,SHOW_STATION;
WS1_ACT        ACTIVITY  NORMAL(OP_TIME(1,1),OP_TIME(1,2),1);
               SET SHOW_STA:=1: SHOW_IN:=NUM(Station1Input): SHOW_ACT:=NUM(WS1_ACT):
                   SHOW_OUT:=NUM(Station1Output);
               SET PART(DEST):=2;
               CLONE,1,SHOW_STATION;
Station1Output QUEUE,FIFO;  {AT WORKSTATION}
```

FIGURE 3 - NETWORK FRAGMENT FOR WORK STATION # 1 OPERATION

routed to the work stations output queue for crane pick up.

The operation of the functional test equipment is dependent on the tote stacker to perform the hook up and un-hook task. When a tester is hooked up by the tote stacker part processing is modeled with the code and network fragment illustrated in Figure 4. The testing of non defective items involves a constant run time modeled with the OK_Test activity block. Defective items are processed by the Test_Defect labelled activity block which involves a uniform distributed test time.

The crane is viewed as performing ten key tasks. The tasked performed by the crane were the basic material moves and crane activities associated with:

1) Move to input queue for new material
2) Move to work station to deliver
3) Move item to storage
4) Move from storage to work station
5) Move to station to pick item up
6) Move to test to load
7) Move to tester to unload
8) Move tested item to exit or storage
9) Move tested item from storage to exit
10) Move from storage to test cell for loading

The crane was modeled as an entity of type CRANE with four attributes. Attributes of the crane include the current X & Y coordinates of the crane, it's destination, and travel time information. The status of storage cells was monitored using an array with individual elements set to zero or one to represent empty/idle states.

Crane usage is modeled as a series of SIMPLE_1 conditions block to prioratize crane activities. Eight CONDITIONS blocks were used to model crane allocation rules. The rules for assigning the crane to available tasks utilized a pull strategy. Preliminary model results indicated a pull allocation strategy provided a more effective use of system resources by attempting to vacate the system, thus tending to keep parts in the output queues rather than in work in process at the beginning of the system. The crane rules used and their relative priority of application are summarized in Table 2. The crane rules were implemented using eight CONDITIONS block to release the crane from an Idle_Crane queue to merge with part entities and perform selected tasks. Figure 5 is a SIMPLE_1 code fragment listing the CONDITIONS blocks used in the model. Re-prioratizing the rules in the model involves altering the order of the conditions blocks listed in figure 5.

When a crane is allocated to perform a task, the responsible CONDITIONS block routes the crane, or crane and part entities, to the specified block in the model. The blocks specified as the targets label in the CONDITIONS blocks are entry points of sub-networks which model crane task elements. For example, when the crane is assigned the task of moving an item from a storage location to a tester the conditions block:

```
CONDITIONS,NUM(TesterWip)>0 AND NUM(IdleTesters)>0,
           TesterWip,,STORE_TO_TEST:
           IdleCrane,,STORE_TO_TEST;
```

routes the crane entity and the item going to a tester to the STORE_TO_TEST labeled block in the model. The referenced block and associated code is illustrated in Figure 6. The move time for the crane to reach the cell is calculated first. Subsequent blocks model the activity time to move to the cell and remove the item. A CLONE block is used to route a copy of the crane to a section of code starting with the block labelled DRAW_CRANE. At Draw_Crane a sub-network manages ACTIVITY, SHOW and CHART blocks with update the cranes position on the screen using arrow characters to indicate crane position. The BRANCH block terminating the code section routes the crane to MoveToLoadTester which executes the moves from the storage cell to the tester and performs the plug in activity. When the crane is ready to unload an item, a SPLIT block is used in the model to control breaking the crane/part entity group into the original entities and routing them to seperate blocks in the model.

The CLONE block in affect is being used as a subroutine to "call" a section of code with a copy of an entity group. Throughout the model CLONE blocks are used to manage animation of the model to compartmentalize code. The compartmentalization served two purposes: to avoid adding redundant coding and secondly, to allow shutting off the graphics by conversion of a few specific blocks to KILL blocks. A sample printout of the screen while the model is running is illustrated in Figure 7. Owing to the methods used to produce this paper the illustration in the figure provides a limited image of what is actually displayed.

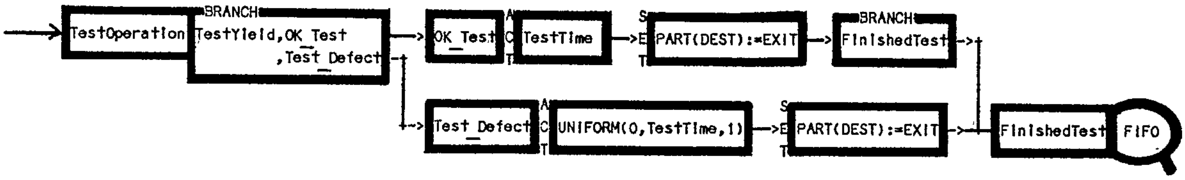
A series of background screens were used to construct a schematic of the system. Over the background screens the animation of the model was created using the CLONE block technique to manage updating the screen. The animation of the model was developed concurrently with the code to simulate the crane and process interactions. The animation of the system was implemented as a validation exercise to visually verify system operation. During the development of the model a number of problems were discovered and corrected based on feedback obtained visually.

RESULTS

The part loading mechanism had a significant impact on the behavior of work in process inventory levels over time. Banking of inventory upstream of the testers was observed during the first shift as expected. In addition to the casual feedback obtained by watching the animation, statistics were collected on inventory levels and cell utilization. Key inventory levels were sampled on half hour intervals using an the array TransientData dimensioned 48X8. In affect, the array was used to collect 48 snapshots per day for the eight key inventory levels:

1) Parts waiting in the input area
2) WIP Storage of items for station #1
3) WIP Storage of items for station #2
4) WIP Storage for testers
5) Number of testers in use
6) Number of testers with a completed part
7) WIP waiting to leave system
8) Part queued at exit station

The utilization of individual storage and tester cells was collected using the array CELLS dimensioned 14X5. The TransientData and CELLS arrays were used to collect observational and time persistent statistics by appending key words to their definition in the DECLARE portion of the model. The key word OBSERVE_STATS appended to the TransientData

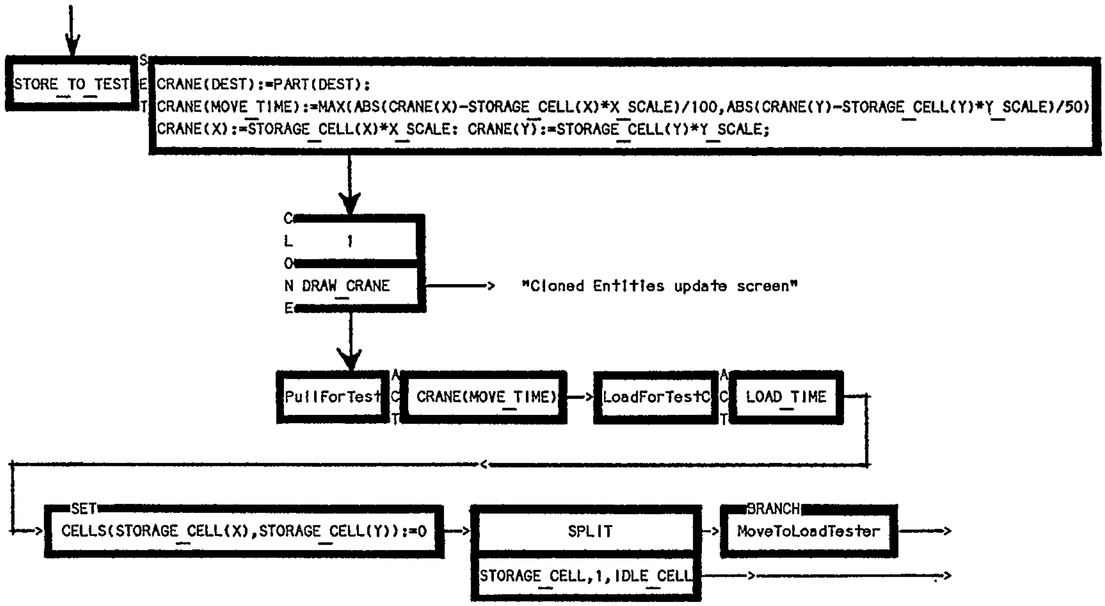



```
$-------------- Tester Code ----------------+
TesterWip       QUEUE,FIFO;
TestOperation   BRANCH  TestYield,OK_Test:
                          ,Test_Defect;
OK_Test         ACTIVITY TestTime;  SET PART(DEST):=Exit;
                BRANCH,FinishedTest;
Test_Defect     ACTIVITY UNIFORM(0,TestTime,1);  SET PART(DEST):=REWORK;
FinishedTest    QUEUE,FIFO;
TesterOutput    QUEUE,FIFO;
```

FIGURE 4 - NETWORK FRAGMENT FOR TESTER OPERATION

```
CONDITIONS,NUM(Exit_Q)<10,
          TesterOutput,,STORE_TO_EXIT:
          IdleCrane,,STORE_TO_EXIT;

CONDITIONS,NUM(EmptyCells)>0 OR NUM(Exit_Q)<10,
          IdleCrane,NUM(FinishedTest)>0,HandleMoveFromTest;

CONDITIONS,NUM(EmptyCells)>0 OR NUM(IdleTesters)>0,
          IdleCrane,NUM(Station2Output)>0,HandleMoveFrom2;

CONDITIONS,NUM(TesterWip) > 0 AND NUM(IdleTesters) > 0,
          TesterWip,,STORE_TO_TEST:
          IdleCrane,,STORE_TO_TEST;

CONDITIONS,NUM(Station2Input)<2,
          Station2Wip,,STORE_TO_WS:
          IdleCrane,,STORE_TO_WS;

CONDITIONS,NUM(Station1Input)<2,
         Station1Wip,,STORE_TO_WS:
         IdleCrane,,STORE_TO_WS;

CONDITIONS,NUM(EmptyCells)>0 OR NUM(Station2Input)<3,
          IdleCrane,NUM(Station1Output)>0,HandleMoveFrom1;

CONDITIONS,NUM(EmptyCells)>0 AND NUM(WaitEntry)>0,IdleCrane,,HandleEntry;
```

FIGURE 5 - CONDITIONS BLOCKS USED TO PRIORITIZE CRANE ALLOCATION




```
$---- MOVE FROM STORAGE LOCATION TO A TEST CELL -------t

STORE_TO_TEST SET CRANE(DEST):=PART(DEST): CRANE(MOVE_TIME) :=
                  MAX(ABS(CRANE(X)-STORAGE_CELL(X)*X_SCALE)/100,
                      ABS(CRANE(Y)-STORAGE_CELL(Y)*Y_SCALE)/50:
                  CRANE(X):=STORAGE_CELL(X)*X_SCALE:
                  CRANE(Y):=STORAGE_CELL(Y)*Y_SCALE;
              CLONE 1,DRAW_CRANE;
PullForTest   ACTIVITY CRANE(MOVE_TIME);
LoadForTestC  ACTIVITY LOAD_TIME;
              SET CELLS(STORAGE_CELL(X),STORAGE_CELL(Y)):=0:
              SPLIT,STORAGE_CELL,1,IDLE_CELL;
              BRANCH, MoveToLoadTester;
```

FIGURE 6 - NETWORK FRAGMENT TO MOVE ITEM FROM STORAGE TO A TEST CELL

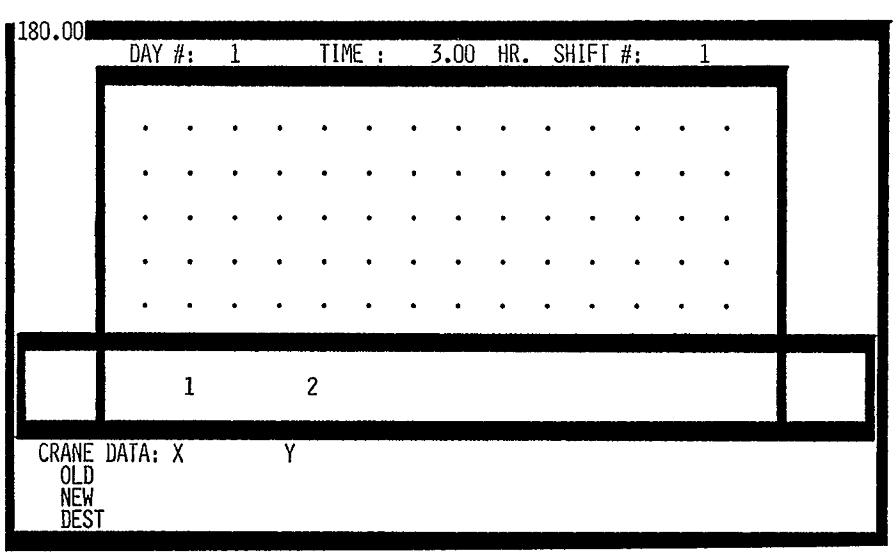


FIGURE 7 - SAMPLE OF SYSTEM ANIMATION PRODUCED BY MODEL

--- TRANSIENT INVENTORY STATISTICS ---

Hour	Entry	Station # 1 WIP	Station # 2 WIP	Tester WIP	# Testrs in use	# Testrs finished	Tester WIP Out	Exit WIP
0.50 :	0.0	21.0	0.4	0.0	12.3	0.0	5.6	10.0
1.00 :	0.0	19.5	0.3	0.0	13.1	0.0	3.6	10.0
1.50 :	0.0	17.9	0.1	0.0	14.0	0.0	1.7	10.0
2.00 :	0.0	16.4	0.1	0.0	15.1	0.0	0.3	9.2
2.50 :	0.0	15.1	0.0	0.0	16.0	0.0	0.0	7.3
3.00 :	0.0	13.4	0.0	0.0	17.1	0.0	0.0	4.8
3.50 :	0.0	11.7	0.1	0.3	18.0	0.0	0.0	2.0
4.00 :	0.0	10.1	0.0	1.0	19.1	0.0	0.0	0.3
4.50 :	0.0	8.4	0.1	2.0	19.5	0.0	0.0	0.0
5.00 :	0.0	7.0	0.0	3.5	19.9	0.0	0.0	0.0
5.50 :	0.0	5.6	0.0	4.8	19.9	0.0	0.0	0.0
6.00 :	0.0	4.0	0.0	6.4	20.0	0.0	0.0	0.0
6.50 :	0.0	2.5	0.0	8.0	20.0	0.0	0.0	0.0
7.00 :	0.0	0.7	0.0	9.6	20.0	0.0	0.0	0.0
7.50 :	0.0	0.0	0.0	10.9	19.9	0.0	0.0	0.0
8.00 :	0.0	0.0	0.0	12.6	20.0	0.0	0.0	0.0
8.50 :	0.0	0.0	0.0	13.0	19.5	0.0	0.0	0.3
9.00 :	0.0	0.0	0.0	11.8	19.6	0.0	0.0	1.5
9.50 :	0.0	0.0	0.1	10.4	19.7	0.0	0.0	2.8
10.00 :	0.0	0.0	0.1	8.9	19.9	0.0	0.0	4.2
10.50 :	0.0	0.0	0.2	7.8	19.6	0.0	0.0	5.1
11.00 :	0.0	0.0	0.2	7.3	20.0	0.0	0.0	5.6
11.50 :	0.0	0.0	0.2	7.2	20.0	0.0	0.0	5.6
12.00 :	0.0	0.0	0.3	7.1	20.0	0.0	0.0	5.6
12.50 :	0.0	0.0	0.3	7.0	20.0	0.0	0.0	5.6
13.00 :	0.0	0.0	0.4	6.8	19.9	0.0	0.0	5.6
13.50 :	0.0	0.0	0.6	6.6	20.0	0.0	0.0	5.6
14.00 :	0.0	0.0	0.7	6.5	20.0	0.0	0.0	5.6
14.50 :	0.0	0.0	0.8	6.4	19.9	0.0	0.0	5.6
15.00 :	0.0	0.0	0.9	6.2	20.0	0.0	0.0	5.6
15.50 :	0.0	0.0	1.0	6.1	20.0	0.0	0.0	5.6
16.00 :	0.0	0.0	1.1	6.0	20.0	0.0	0.0	5.6
16.50 :	16.4	5.7	1.1	5.3	20.0	0.0	0.0	6.3
17.00 :	8.3	13.8	1.1	3.7	19.6	0.0	0.0	7.9
17.50 :	0.5	21.5	1.1	2.3	19.6	0.0	0.5	8.8
18.00 :	0.0	22.1	1.2	1.3	19.2	0.0	1.2	9.4
18.50 :	0.0	22.1	1.3	0.3	18.7	0.0	2.0	9.9
19.00 :	0.0	22.1	1.3	0.0	17.5	0.0	3.6	10.0
19.50 :	0.0	22.1	1.3	0.0	16.4	0.0	4.8	10.0
20.00 :	0.0	22.1	1.3	0.0	15.1	0.0	6.0	10.0
20.50 :	0.0	22.1	1.3	0.0	14.5	0.0	6.7	10.0
21.00 :	0.0	22.1	1.3	0.0	14.0	0.0	7.2	10.0
21.50 :	0.0	22.1	1.4	0.0	13.5	0.0	7.6	10.0
22.00 :	0.0	22.1	1.5	0.0	13.1	0.0	7.9	10.0
22.50 :	0.0	22.1	1.5	0.0	13.1	0.0	7.9	10.0
23.00 :	0.0	22.1	1.5	0.0	12.9	0.0	8.1	10.0
23.50 :	0.0	22.1	1.6	0.0	12.7	0.0	8.2	10.0
24.00 :	0.0	22.1	1.6	0.0	12.5	0.0	8.3	10.0

--- STORAGE CELL % UTILIZATION STATISTICS ---

	<---STORAGE CELLS									--->	<---TESTERS			--->
	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0
1.00 :	26.2	33.4	28.4	32.9	40.3	35.9	34.3	35.9	33.9	33.0	91.6	88.5	90.4	90.7
2.00 :	35.8	35.6	27.1	38.3	34.7	37.2	25.4	36.2	35.6	34.9	91.3	91.8	87.6	89.5
3.00 :	33.9	32.9	30.4	25.2	28.6	36.2	36.2	35.8	35.7	28.3	89.9	89.4	85.4	90.5
4.00 :	36.1	30.6	27.6	36.8	35.3	36.9	35.1	27.3	37.4	29.9	86.0	89.2	88.5	88.8
5.00 :	34.8	33.3	30.9	27.8	36.4	35.9	36.2	35.8	34.7	29.1	87.0	89.8	88.9	87.3

FIGURE 8 - CUSTOMIZED SUMMARY REPORT OF MODEL RESULTS

declaration specified collection of observations statistics to be collected on each element of the TransientData array. The TIME_STATS key word was employed to trigger collection of time weighted statistics on individual elements of the CELLS array. A large number of statistics are accordingly produced by the standard report. A custom report was written to a disk file which summarizes the WIP and utilization statistics. A sample report is illustrated in Figure 8. The main result obtained was with regard to the time varying behavior of the inventory levels and the uniform utilization of storage cells. In addition to the custom summary report, the individual observations for inventory levels were written to a disk file in addition to time in system observations.

Due to the FIFO allocation strategy for re-using idle testers and storage cells the results obtained indicated a uniformly distributed utilization of storage cells. The overall utilization of approximately 30 percent for storage cells must be interpreted with caution. The aggregate utilization statistics are a conglomeration of cell usage across all three shifts. The utilization of storage cells is dependent on time of day as indicated by the transient inventory statistics sampled on half hour intervals. For example, with the start of the first shift occurring at hour 0.00, the maximum average utilization of storage cells occurred 1.5 hours into the third shift with almost 90 percent of the cells in use. Conversely, storage cell usage 1.5 hours into the first shift was approximately 40 percent.

The results obtained were from a simulation of 12 days operation of the system with statistics cleared at the end of the second day. The execution time for the model was calculated using the SIMPLE_1 function SYS_TIME which returns the time from the operating system of the computer. For the simulation of the pull strategy for crane allocation the model produced a file with the message:

```
CRANE.MDL RAN FOR :    2.3090  HOURS...
```

The run time was obtain running SIMPLE_1 on an AT&T PC 6300. The run time observed was for the animated model collecting all of the array statistics and writing observations to disk files.

The standard SIMPLE_1 report for all statistics collected was outputed to disk. One of the rationales for producing the custom formated report illustrated in Figure 8 is the volume of the standard report. For this example, the standard report prints full statistics on all array elements, blocks etc. and is nominally 800 lines in length.

CONCLUSIONS

The ability to model complex material handling systems has been demonstrated. The initial model summarizes modeling technique and is re-usable as a "tool box" for building models of similar systems. Follow on developments suggested by the results obtained is a generalization of the model for use in a library of "tool box" programs for analysis of materials handling systems. The approach taken to model the system tailored the crane allocation logic to the nuances of the system thus allowing the analysis and design of the control strategy for crane allocation. The ability to collect detailed statistics on the time varying behavior combined with an animation of the system was found particularly useful. Animation of the system in particular aided the understanding of the statistics obtained and their limitations in one case. Finally, animation of the model was found extremely helpful in validation of the models operation.

REFERENCES

Cobbin, Philip, "SIMPLE_1: A simulation environment for the IBM PC", Modeling and Simulation on Microcomputers, Claude, C. Barnett, Editor, Society for Computer Simulation, La Jolla, 1986, pp 243-248.

Cobbin, Philip, "Applying SIMPLE_1 to manufacturing systems", Summer Computer Simulation Conference, July 28-30 1986, Reno, Nevada, Roy Crosbie and Paul Luker, Editors, Society for Computer Simulation, La Jolla, pp 724-730.

Cobbin, Philip, " A tutorial on the SIMPLE_1 simulation environment" To be published in: Winter Simulation Conference proceedings, December 1986, Washington D.C.

Sierra Simulations & Software: SIMPLE_1 User's guide and reference manual, 1985.

Starr, Patrick, Skrien,Douglas, and Meyer, Robert,"Simulating schedule recovery strategies in manufacturing assembly operations" To be published in: Winter Simulation Conference proceedings, December 1986, Washington D.C.

Philip Cobbin is the owner of Sierra Simulations & Software and is the developer of SIMPLE_1. Phil has developed and taught simulation to students and industry. He holds a Master of Science in Industrial Engineering from Purdue University, a Bachelor of Science in Industrial Engineering and Operations Research from the University of Massachusetts at Amherst, and an Associate in Science degree in Manufacturing Engineering Technology from Waterbury Connecticut State College. Phil is a native of Los Angeles and has been previously employed by the General Products Division of the International Business Machines corporation performing simulation modeling and material handling engineering activities.