SIMULATION SYSTEM FOR REAL-TIME PLANNING, SCHEDULING, AND CONTROL

Glenn R. Drake

Systems Modeling Corporation
504 Beaver Street
Sewickley, Pennsylvania 15143, U.S.A.

Jeffrey S. Smith

Department of Industrial Engineering
Texas A&M University
College Station, Texas 77843, U.S.A.

ABSTRACT

This paper introduces a framework for on-line simulation systems in the operational planning, scheduling, and control of manufacturing systems. Five basic concepts for software design of an on-line simulation system are identified and an example simulator is illustrated.

1 MOTIVATION

There has been a growing interest in real-time, "intelligent" shop floor control systems that use simulation technology to predict the future impact of short-term manufacturing decisions (Erickson *et al.* 1987, Wu and Wysk 1989, Harmonosky and Robohn 1991, Rogers and Flanagan 1991, Smith *et al.* 1994, Drake *et al.* 1995, Jones *et al.* 1995). Although recent researchers have examined the responsibilities and underlying issues of on-line simulation systems, there has been a tendency to view the simulation software as a "black box" and examine solely its interactions with other system components such as data collection devices, neural networks, expert systems, genetic algorithms, control mechanisms, and process planning functions. Thus, the design of the simulation system itself has been largely ignored and a framework for on-line simulation systems is still needed.

There is currently a large commercial market for simulation-based analysis in manufacturing. This market is illustrated by the demand for products such as Arena/SIMAN® (Pegden *et al.,* 1995), SlamSystem™ and FACTOR™ (O'Reilly, 1995), Witness™ and PROVISA™ (Thompson, 1995), Extend™ (Krahl, 1995), and AutoMod/AutoSched™ (Roher, 1995). Some of these packages (e.g., Arena/SIMAN, Extend, Witness, SlamSystem) are primarily designed for and used in design applications (i.e., long-term, predictive analysis). Others, such as AutoSched, PROVISA, and FACTOR, are primarily designed for and used in short-term finite capacity planning and scheduling applications. Irrespective of their primary focus, it is often difficult to implement these systems in on-line planning, scheduling, and control applications due to one or more of the following reasons:

(1) The conceptual framework is geared exclusively towards use by humans and not the decision support software commonly found in shop floor control systems for CIM.

(2) The system does not employ modeling constructs or a framework that facilitates the development of simulation models intended for real-time use. This includes inadequate support for modeling complex control logic, whereby either the necessary constructs are not available or the modeling of control logic is integrated with physical characteristics such that changes are difficult to implement. Additionally, many systems do not incorporate constructs for real-time task dispatching.

(3) The conceptual framework implicitly assumes that simulationists (i.e., modelers) and end-users are identical by providing a single primary interface or a set of "rigid" interfaces (e.g., a "scheduling" interface and a "modeling" interface) for constructing models and executing simulation experiments. However, in the context of on-line planning, scheduling, and control applications, end-users often include a *variety* of personnel (e.g., schedulers, capacity planners, managers) who were not directly involved in model development, have little or no simulation expertise, and who desire to use only a portion of the simulation tool.

The common inefficiencies of current commercial packages and the increasing interest in real-time planning, scheduling, and control indicate that an effective framework for on-line simulation systems is needed. When developing important concepts for such a framework, it is useful to examine the applications of on-line simulation technology.

2 THE USERS OF ON-LINE SIMULATION SYSTEMS

On-line simulation systems incorporate two powerful features: 1) the ability to reliably *predict* the future behavior of the shop floor given its current status, and 2) the ability to *emulate* and/or *dictate* the control logic of a manufacturing system. These two capabilities offer potential benefits to a variety of end-users in a manufacturing organization. Figure 1 shows an on-line simulation system integrated with several functional areas of a facility.

First, simulation can help *capacity planners* determine order lot sizes, release dates, and work calendars for resources by incorporating simulated scheduling constraints in their decision making. Second, once orders are released, simulation offers real-time *schedulers* the ability to evaluate strategies for sequencing jobs through the workcenters. In dynamic manufacturing environments, it may be advantageous to change the way a shop is controlled at certain points in time.

Third, once the desired operational strategy and part mix have been determined, simulation can interact in real-time with *shop floor management.* Work lists can be generated and distributed to shop floor execution or dispatched on-line in a task-by-task manner (Smith *et al.* 1994). Running parallel to the actual system in real-time allows the simulation system to keep track of current status and send feedback to the scheduling function on the current schedule's performance.

Fourth, by emulating current shop conditions, simulation can help the marketing and sales functions reliably predict order leadtimes and quote accurate duedates to customers. Using simulation in this manner can decrease the amount of work in process and add to the reputation of the organization by improving its ability to meet promised delivery dates (Rogers and Flanagan 1991). Finally, the simulation system must interact with engineering for the necessary data (e.g., process plans, layout information) to perform valid analysis. Simulation output can give feedback to the engineering function on the performance of a current design (e.g., the layout of a workcenter).

3 CONCEPTS FOR ON-LINE SIMULATION SYSTEMS

It is apparent that several functional areas of an organization might interact with on-line simulation technology. In response, five basic concepts for simulation systems in on-line planning, scheduling, and control are identified:

(1) The system should be multifaceted as suggested by Zeigler (1980). A multifaceted environment acknowledges that the system is approachable from different points of view by different users with different objectives at different times.

(2) The system should separate the logically distinct activities of on-line simulation-based planning, scheduling, and control. Thus, the functional stages of modeling physical structure, modeling control logic, simulation input, experimentation, output analysis, and task dispatching should be distinguished through a modular approach that recognizes the details and mechanisms required for each element.

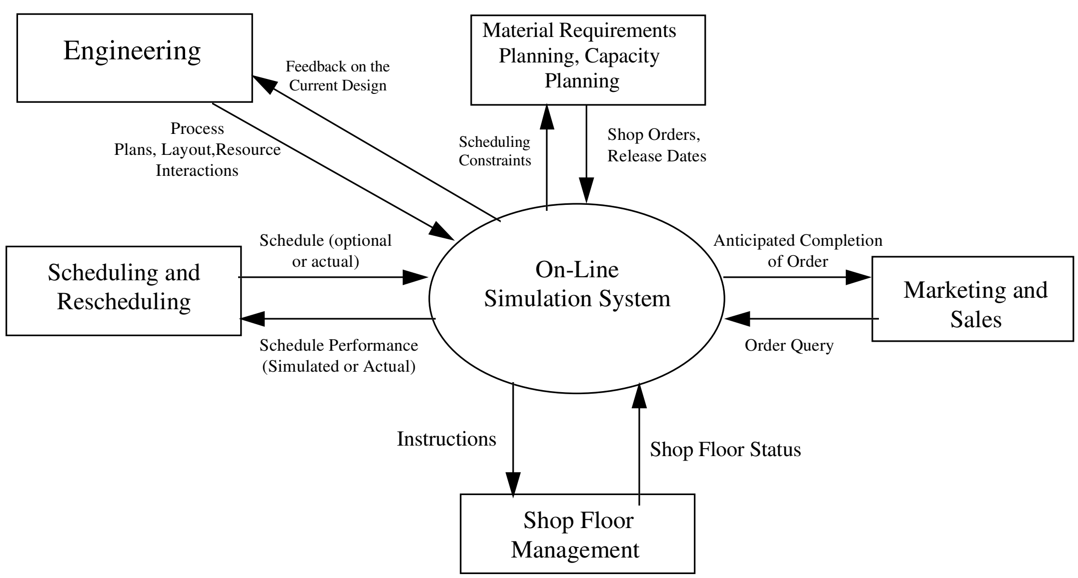

Figure 1. An on-line simulation system for planning, scheduling, and control.

(3) The system should incorporate not only good human interfaces, but also explicit software interfaces to the activities of model development, simulation input, experimentation, output analysis, and task dispatching.

(4) The system should allow a model of a manufacturing system to be *re-used* for design, planning, scheduling and control with minimal modification within and between applications.

(5) The system should be flexible yet easy to use for modelers *and* end-users (Pinedo 1995). Flexibility within the simulation system is necessary for modeling complex control logic and reusing resident simulation models. The ease of use of the software is dependent on how well the skills and perspectives of end-users are recognized (i.e., a multifaceted environment).

4 GENERAL FRAMEWORK

When simulationists develop simulation models and turn them over to end-users for experimentation, they are delivering *simulators*. These end-users often include plant engineers, managers, and sales representatives with little or no simulation expertise. Though they may contribute input or functional specifications to the modeler(s) for project purposes, these personnel are usually not directly involved in model development. Instead, they require the simulator for daily decision making and task dispatching. If the same interface is used for these end-users as was used for the simulationists, then these end-users have access to simulation primitives that may be confusing no matter how "easy" the package is to use (Brunner and Crain 1991). Moreover, access to these primitives is generally unnecessary.

Figure 2 shows a general framework for on-line simulation systems in scheduling and control. In departure from traditional simulation software, the user interface is separated into two distinct environments: the *modeling environment* for simulationists and the *simulator environment* for end-users. The modeling environment focuses on the simulation constructs and logic necessary for developing simulation models intended for on-line use. Once a model is completed, it is "handed over" to the simulator environment which focuses upon the activities that comprise the real-time planning, scheduling, and control process (e.g., data entry, experimentation, output analysis).

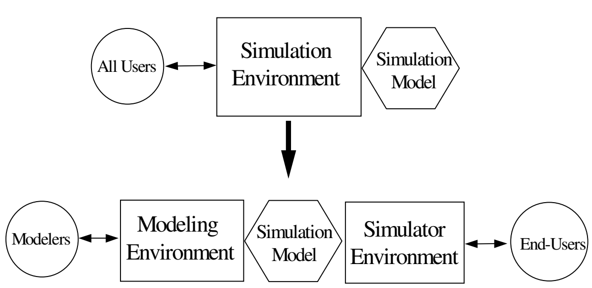

Figure 2. General framework.

The simulation model developed for on-line planning, scheduling, and control applications can be decomposed into five primary functional elements. These functional elements are:

(1) *Physical Structure:* The definition of machines, people, tooling, transporters, and storage areas in the facility, as well as the layout and decision points of the manufacturing process.

(2) *Task Dispatching:* The simulator's I/O links with external components for task dispatching and the protocol of these messages.

(3) *Control Logic:* The operating rules (e.g., scheduling rules for equipment and people, batching rules), simulator run-time parameters (e.g., run length, output generation period), and exception handling (e.g., a dispatched task is not performed or is performed improperly).

(4) *Simulation Input:* The product definitions (e.g., routings, processing times, resource requirements, bills of materials), product demand (e.g., order quantities, due dates, start dates, priorities), resource relationships (e.g., resource groups), and work calendars (e.g., alternative shifts).

(5) *Simulation Output:* The start- and end-times for tasks in the system. These times may include additional "tag" information such as task codes, affected products, or utilized resources.

Figure 3 illustrates the functional elements of an on-line simulation model and their relationships with the modeling and simulator environments in Figure 2. Note that the modeling environment is separated into two distinct frames or interfaces: the *model frame* for defining the physical structure and task dispatching of the simulator, and the *rules frame* for developing the control logic or rule base of the simulator (e.g., alternative scheduling rules, batching rules, procedures for exception handling). Functions for loading simulation input from input databases and writing simulation output to output databases are automatically compiled during model development and do not require manual editing. End-users define simulation input and manipulate simulation output (e.g., generate Gantt charts) through the simulator environment.

The modeling environment's organization incorporates flexibility into the simulation system in two primary ways. First, it facilitates the development of simulators which separate the functional elements of physical structure, control logic, simulation input, and simulation output so that different scenarios (e.g., part mixes, scheduling rules, work calendars) can be readily "plugged in" and evaluated without model recompilation. Constructs for modeling task dispatching are incorporated with constructs for modeling physical structure such that the same model used for design, planning, and scheduling can be used for direct shop floor control. Thus, within the simulator environment, end-users exploit generic, reusable models designed for easy "what if" experimentation and automated execution. This concept is referred to as *flexible simulation* (Drake *et al.* 1995). The modularity of the rules frame facilitates the addition of new rules to the simulator's rule base.

Figure 3 shows the simulator environment separated into three distinct frames: the *entity frame* for simulation input (e.g., entering production data), the *experiment frame* for experimentation and execution (e.g., selecting operating rules from the rule base), and the *output analysis frame* for data manipulation and presentation (e.g., creating Gantt charts, order tracking). The *interactive montitor* provides a real-time interface to the on-line simulator tool. This monitor might include real-time animation, and could feasibly be detached from the simulator environment and implemented "stand-alone" at workcenters for real-time interaction (e.g., entering status and receiving tasks from the simulator).

Additionally, the proposed simulator environment incorporates software interfaces for the activities of simulation input, experimentation, output analysis, and real-time task dispatching. These software interfaces are illustrated in Figure 4. A *simulator language* to the manager of the simulation engine facilitates the automation of real-time planning, scheduling and control applications. The language incorporates commands for verifying simulator programs (i.e., debugging), loading simulation input from the entity frame, setting up and executing simulation experiments in the experiment frame, and calling analysis and presentation routines in the output analysis frame. Macros of commands might be created for end-users with different "views" or objectives (e.g., a scheduling macro, a marketing macro). To dispatch tasks on-line, the simulation system and shop floor management communicate through a *task initiation queue* (TIQ) and a *task completion queue* (TCQ) (Smith *et al.* 1994). Monitoring and tracking of tasks can also be implemented using the TIQ/TCQ interface. *Structured query language* (SQL) might be used to link with a variety of database management systems and extract simulation input (e.g., shop status, process plans) or forward simulation output (e.g., schedules, performance, expected completion times).

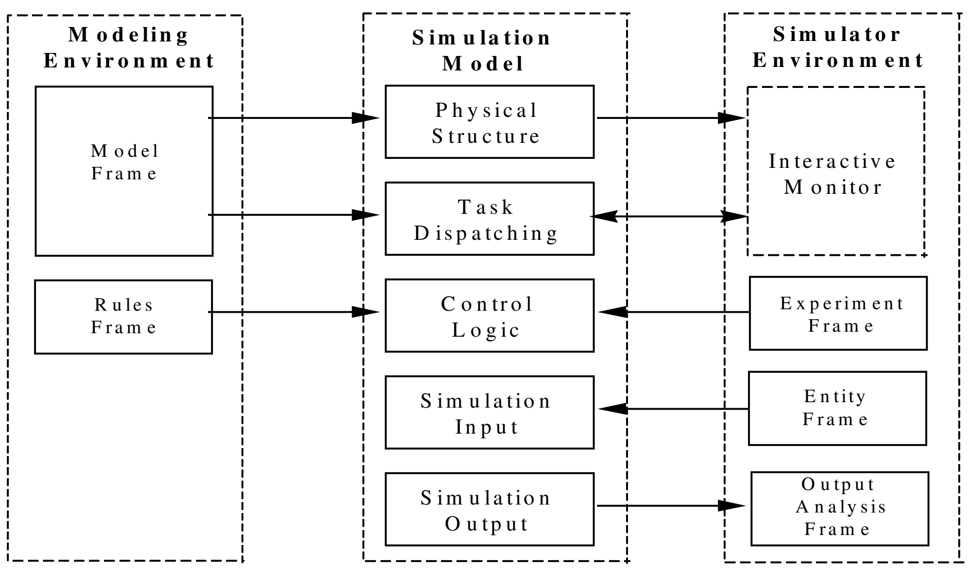

Figure 3. General framework and functional elements of an on-line simulation model.

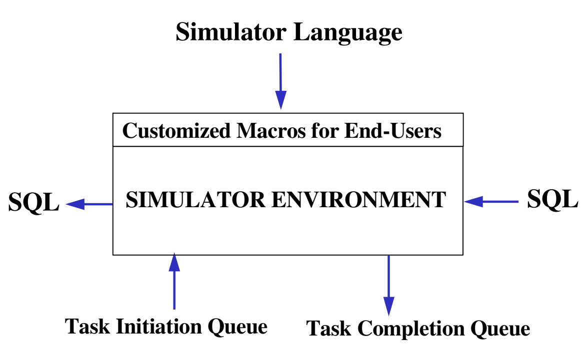

Figure 4. Software interfaces to the simulator environment.

5 EXAMPLE OF A CONVENTIONAL ON-LINE SIMULATOR

To further motivate the general framework discussed in Section 4, a conventional SIMAN V (Pegden, *et al.*, 1995) simulator for a simple manufacturing system is now illustrated. The example system is shown in Figure 5. Six types of jobs arrive from Workcenter A to Workcenter B. Workcenter B is a small station that consists of two machines with an input buffer in front of each. Depending on the part type, jobs can be processed on Machine 1 only, Machine 2 only, or either of the resources. Jobs are forwarded to Workcenter C after processing. The operators at Workcenter B have three primary objectives. First, they want short lead times to aid on-time completions of orders at the factory level. Second, they want low levels of work-in-process (WIP) to minimize inventory costs. Third, they would like to forward anticipated completion times of jobs to the personnel of Workcenter C so that those operators can plan their operations. To meet their objectives, the operators at Workcenter B confront four major operating decisions. Three of these decision points are illustrated as question marks in Figure 5. The first decision entails incoming jobs with alternative routings, whereby one of the two machines must be selected. Machine 2 is the more expensive model and processing times are always shorter on this resource. The next two decisions involve the processing sequences on Machine 1 and Machine 2 (i.e., job dispatching from the input buffers). The last decision involves the work calendars of Machine 1 and Machine 2, whereby a fixed daily percentage of downtime is required for each resource for routine maintenance.

Due to the dynamic nature of the product mix released from Workcenter A, and the impact of the four operating decisions, it is considered advantageous to change the decision strategies of Workcenter B at certain points in time. Thus, to maintain short lead times and low WIP, strategies are alternated based on real-time shop conditions. Table 1 lists the processing times for the six part types manufactured in Workcenter B and the alternative strategies (i.e., decision rules) for machine selection and queue dispatching. These rules are sensitive to the workstation's dynamics and considered good candidates for improved performance. A model of Workcenter B was developed for on-line planning, scheduling, and control applications at Workstation B using the SIMAN V simulation language and environment. The simulator is generic in that different combinations of the decision rules in Table 1 can be assigned without model editing or recompilation. This simplifies "what if" experimentation by end-users unfamiliar with SIMAN or the model's structure. Once a satisfactory strategy is determined, task lists can be generated and distributed to the operators in Workcenter B and predicted completion times of jobs forwarded to the personnel in Workcenter C. SIMAN code for the model and experiment files is shown in Figures 6 and 7 respectively. X's mark the functional elements (i.e., physical structure (P), control logic (C), task dispatching (T), simulation input (I), and simulation output (O)) associated with each SIMAN construct. For example, Line 6 in Figure 6 defines some of the physical structure of the model as well as task dispatching.

Figures 6 and 7 illustrate the modeling effort required to develop a "user-friendly" simulator using the conventional Arena/SIMAN environment. Though an attempt was made by the modeler to textually partition the model and experiment files into their functional elements, the simulation system does not explicitly support a modular separation of these activities. Some general observations will now be made on the simulator's functional elements:

(1) *Physical Structure:* Lines 2 through 12 in the model file and Lines 3 through 6 in the experiment file define the layout and decision points of the process in Workcenter B. Within the proposed framework, these constructs would be defined in the model frame of the modeling environment.

(2) *Task Dispatching:* Lines 6, 11, and 52 of the model file define the I/O points for external communication and Lines 27 through 30 of the experiment file define the communication protocol. Within the proposed framework, these constructs would be defined in the model frame.

(3) *Control Logic:* Lines 13 through 48 in the model file and Line 25 in the experiment file define the rule base. Within the proposed framework, these constructs would be defined in the rules frame of the modeling environment. End-users select rules from the

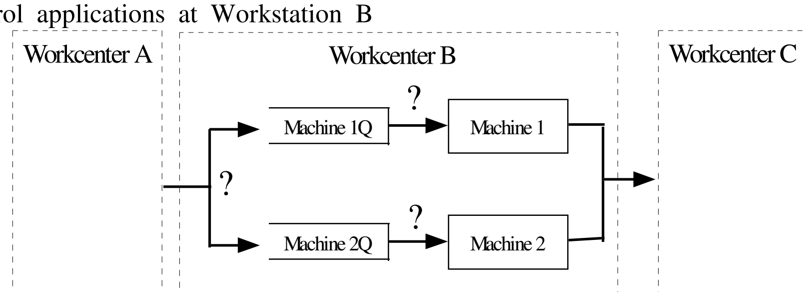

Figure 5. Example system

rule base by assigning numerical values to the variables *Decision1*, *Decision2*, and *Decision3*.

(3) The assignment of work calendars to the machines requires the editing of Lines 3 and 4 in the experiment file. Within the proposed framework, all of these assignments would take place in the experiment frame of the simulator environment and not require model editing.

(4) *Simulation Input:* Lines 51 through 57 in the model file define code for reading production orders. Within the proposed framework, this logic would be automatically compiled during model development. In the experiment file, new orders are entered from the external file Orders.DB (Line 31 in Figure 7). The processing times for each part type are defined in Lines 13 and 14, and the work calendars for machines defined in Line 26. Entering new product types, processing times, or calendars requires modification of the model code. Within the proposed framework, this data is external to the model and is defined through the entity frame of the simulator environment.

(5) *Simulation Output*: Lines 59 through 66 in the model file define code for recording the start- and end-times of tasks in the system. This aggregate data is forwarded to the external file Output.DB (Line 32 in Figure 7). Within the proposed framework, these constructs would be automatically compiled during model development. Manipulation of the data (e.g., computing performance measures) would take place in the output analysis frame of the simulator environment. Different statistics/presentation macros might be applied by end-users with different objectives.

SUMMARY

The simulator presented in Figures 6 and 7 is "flexible" in that different scheduling rules can be defined in the model without recompilation. Also, the simulation output is aggregate such that multiple statistics can be calculated from the raw data by external functions. Thus, the output from the simulator is useful for a variety of end-users with different objectives. The generic characteristics of the model simplifies operational use of the tool by personnel unfamiliar with the SIMAN simulation language.

However, to change work calendars or enter additional production data *would* require editing of the model code. Developing flexibility for these instances necessitates additional effort by the modeler. For large models, it is likely the efforts to make user-friendly models could be considerable. The framework presented in this paper simplifies the development of generic simulation models for on-line use.

REFERENCES

Brunner, D. T., and Crain, R. C., 1991, GPSS/H in the 1990s. *Proceedings of the 1991 Winter Simulation Conference*: 81-85.

Drake, G., Smith, J.S., and Peters, B. A., "Simulation as a Planning and Scheduling Tool for FMS," *Proceedings of the 1995 Winter Simulation Conference*, December 1995, Washington DC.

Table 1. Operation times and decision rules.

Processing Times (Min)			Rule Base
PartType	M/C 1	M/C 2	*Alternative Machine Selection:*
1	10	-	Always M1, Always M2, SNQ, STPT
2	-	6	*Machine 1Q Dispatching:*
3	13	-	FIFO,SPT
4	-	9	*Machine 2Q Dispatching:*
5	15	8	FIFO,SPT
6	12	6	

Erickson, C., Vandenberge, A., and Miles, T., 1987, Simulation, animation, and shop-floor control. *Proceedings of the 1987 Winter Simulation Conference*: 649-652.

Harmonosky, C. M., and Robohn, S. F., 1991, Real-time scheduling in computer integrated manufacturing: A review of recent literature. *International Journal of Computer Integrated Manufacturing*, **4**, 331-340.

Jones, A., Rabelo, L., and Yuehwern, Y., 1995, A hybrid approach for real-time sequencing and scheduling. *International Journal of Computer Integrated Manufacturing*, **8**, 145-154.

Krahl, D., An Introduction to Extend, *Proceedings of the 1995 Winter Simulation Conference*, December 1995, Washington DC.

O'Reilly, J. J., Introduction to SLAM II and SLAMSYSTEM, *Proceedings of the 1995 Winter Simulation Conference*, December 1995, Washington DC.

Pegden, C. D., Shannon, R. E., and Sadowski, R. P., 1995, *Introduction to Simulation Using SIMAN*, McGraw-Hill, Inc., New York.

Pindeo, M., 1995, *Scheduling: Theory, algorithms, and Systems*, Prentice Hall, Englewood Cliffs, NJ.

Rogers, P., and Flanagan, M., 1991, On-Line simulation for real-time scheduling of manufacturing systems. *Industrial Engineering*, **23**, 37-40.

Rohrer, M., AutoMod, *Proceedings of the 1995 Winter Simulation Conference*, December 1995, Washington DC.

Smith, J. S., Wysk, R. A., Sturrock, D. T., Ramaswamy, S. E., Smith, G. D., and Joshi, S. B., 1994, Discrete event simulation for shop floor control. *Proceedings of the 1994 Winter Simulation Conference*: 962-969.

Thompson, W. B., A Tutorial for Modeling with the WITNESS Visual Interactive Simulator, *Proceedings of the 1995 Winter Simulation Conference*, December 1995, Washington DC.

Wu, D. S., and Wysk, R. A., 1989, An application of discrete-event simulation to on-Line control and scheduling in flexible manufacturing. *International Journal of Production Research*, **27**, 1603-1624.

Zeigler, B. P., 1980, Concepts and software for advanced simulation methodologies. *IEEE: Simulation with Discrete Models: A State-of-the-Art View*, 25-43.

AUTHOR BIOGRAPHIES

GLENN R. DRAKE is an associate applications engineer for Systems Modeling Corporation. He received his B.S.I.E. in 1994 and his M.S.I.E. in 1996 from Texas A&M University.

JEFFREY S. SMITH is an assistant professor in the Industrial Engineering Department at Texas A&M University. His research interests are in shop floor control, manufacturing systems design, analysis, control, and simulation.

SIMAN Constructs	Comments	P	C	T	I	O
BEGIN;	Begin Model	X				
;**********Physical Structure******************						
;Processing at Machine 1						
Mach1Q QUEUE,Machine1Q;	The Machine 1 Queue	X				
SEIZE,Machine1:NEXT(Out1);	Seize Machine 1	X				
In1 DELAY,ProcessTime,ProcessPart:NEXT(Out2);	Process the Part	X		X		
In2 RELEASE,Machine1:DISPOSE;	Release Machine 1&Leave	X				
;Processing at Machine 2						
Mach2Q QUEUE,Machine2Q;	The Machine 2 Queue	X				
SEIZE,Machine2:NEXT(Out3);	Seize Machine 2	X				
In3 DELAY,ProcessTime,ProcessPart:NEXT(Out4);	Process the Part	X		X		
In4 RELEASE,Machine2:DISPOSE;	Release Machine 2&Leave	X				
;**********Control Logic of System**********************						
;Alternative Machine Decision (Decision 1)						
Ent BRANCH,1:	Alternative Machine Rules		X			
IF,PartNum.EQ.1.OR.PartNum.EQ.3.OR.Decision1.EQ.1,Mach1:	Select Machine 1 In-Buffer		X			
IF,PartNum.EQ.2.OR.PartNum.EQ.4.OR.Decision1.EQ.2,Mach2:	Select Machine 2 In-Buffer		X			
IF,Decision1.EQ.3,SNQ:	SNQ Rule to Select		X			
IF,Decision1.EQ.4,STPT;	STPT Rule to Select		X			
;Smallest Number in Queue Rule (SNQ)						
SNQ BRANCH,1:	SNQ Rule		X			
IF,NQ(Machine1Q).EQ.NQ(Machine2Q),STPT:	STPT Tiebreaker		X			
IF,NQ(Machine1Q).GT.NQ(Machine2Q),Mach2:	Select Machine 2 In-Buffer		X			
ELSE,Mach1;	Select Machine 1 In-Buffer		X			
;Shortest Total Processing Time in Queue Rule (STPT)	STPT Rule					
STPT ASSIGN:Counter=1:			X			
ProcessTime=OpTime(PartNum,1):	Calculate the Total				X	
SUM1=ProcessTime;	Processing Time in the In-					
WHILE:Counter.LE.NQ(Machine1Q);	Buffer for Machine 1		X			
ASSIGN: SUM1=SUM1+AQUE(Machine1Q,Counter,1):			X			
Counter=Counter+1;						
ENDWHILE;			X			
ASSIGN:Counter=1:			X			
ProcessTime=OpTime(PartNum,2):	Calculate the Total				X	
SUM2=ProcessTime;	Processing Time in the In-					
WHILE:Counter.LE.NQ(Machine2Q);	Buffer for Machine 2		X			
ASSIGN: SUM2=SUM2+AQUE(Machine2Q,Counter,1):			X			
Counter=Counter+1;						
ENDWHILE;			X			
BRANCH,1:			X			
IF,SUM1.GT.SUM2,Mach2:	Select Machine 2 In-Buffer		X			
ELSE,Mach1;	Select Machine 1 In-Buffer		X			
;Machine 1 Queue Dispatching (Decision 2)						
Mach1 ASSIGN: Priority=QueueRules(Decision2):	Assign Queue 1 Rank		X			
ProcessTime=OpTime(PartNum,1):NEXT(Mach1Q);					X	
;Machine 2 Queue Dispatching (Decision 3)						
Mach2 ASSIGN: Priority=QueueRules(Decision3):	Assign Queue 2 Rank		X			
ProcessTime=OpTime(PartNum,2): NEXT(Mach2Q);					X	
;******Simulation Input and Output**********************						
;Reading Orders from Orders Database						

Line	SIMAN Constructs	Comments	P	C	T	I	O
51	CREATE,1:,1;	Create One Entity				X	
52	loop1 DELAY,0,ReadOrder;	Send ReadOrder Task			X	X	
53	loop2 READ,OrdersFile:OrderNum,PartNum,Quantity,Again;	Read an Order				X	
54	DUPLICATE: Quantity,Id;	Create the Order				X	
55	BRANCH,1:	Read another Order?				X	
56	IF,Again.EQ.0,loop1:ELSE,loop2;	Assign Part Identifiers				X	
57	Id ASSIGN: NUM=NUM+1: PartID=NUM: NEXT(Ent);	Enter Part into System				X	
58	**;Generating Output**						
59	Out1 WRITE,OutFile:”S,1,%8.3f,%6.0f\n”:	Write StartTime of job					X
60	TNOW,OrderNum,PartID:NEXT(In1);	at Machine 1					
61	Out2 WRITE,OutFile:”F,1,%8.3f,%6.0f\n”:	Write EndTime of job					X
62	TNOW,OrderNum,PartID:NEXT(In2);	at Machine 1					
63	Out3 WRITE,OutFile:”S,2,%8.3f,%6.0f\n”:	Write StartTime of job					X
64	TNOW,OrderNum,PartID:NEXT(In3);	at Machine 2					
65	Out4 WRITE,OutFile:”F,2,%8.3f,%6.0f\n”:	Write EndTime of job					X
66	TNOW,OrderNum,PartID:NEXT(In4);	at Machine 2					

Figure 6. Example SIMAN V model file.

Line	SIMAN Constructs	Comments	P	C	T	I	O
1	BEGIN,Yes,Yes;	Begin Experiment	X				
2	PROJECT,Example,GRD,,No;	Model Information	X				
3	RESOURCES: 1,Machine1,SCHEDULE(Daily);	Machine 1 Resource	X	X			
4	2,Machine2,SCHEDULE(Daily);	Machine 2 Resource	X	X			
5	QUEUES: 1,Machine1Q,LVF(Priority);	Machine 1 Queue	X	X			
6	2,Machine2Q,LVF(Priority);	Machine 2 Queue	X	X			
7	VARIABLES: PartExec:	Emulation Variable			X		
8	ReadExec:	Emulation Variable			X		
9	Counter:	Loop Variable		X			
10	SUM1:	Sums Variable		X			
11	SUM2:	Sums Variable		X			
12	NUM:	Part Identifier Variable					
13	OpTime(6,2),10,0,13,0,15,12,	Processing Times				X	
14	0,6,0,9,8,6:						
15	Decision1:	Alternative Machine Var.		X			
16	Decision2:	Queue 1 Dispatching Var.		X			
17	Decision3;	Queue 2 Dispatching Var.		X			
18	ATTRIBUTES: ProcessTime:	Processing Time				X	
19	OrderNum:	Order Identifier				X	
20	PartNum:	Part Type				X	
21	PartID:	Part Identifier				X	
22	Quantity:	Order Quantity				X	
23	Priority:	Queue Rank		X			
24	Again;	Order Variable				X	
25	EXPRESSIONS: QueueRules(2),TNOW,ProcessTime;	Queue Ranking Rules		X			
26	SCHEDULES: Daily,1*480,0*120,1*480,0*120,1*480	“Daily” Work Calendar				X	
27	TASKS: 1,ProcessPart,PartExec,	TIQ task for processing			X		
28	“process part %6.0f”,PartID,IDENT:						
29	2,ReadOrder,ReadExec,	TIQ task for readingOrders			X	X	
30	“Enter Order TGID=%6.0f”,IDENT;						
31	FILES: 1,OrdersFile,”Orders.DB”,SEQ,,Rewind,”,”:	Orders Database				X	
32	2,OutFile,”Output.DB”,SEQ,,Rewind,;	Output Database					X
33	REPLICATE,1,0,1800,Yes,Yes;	Run Control Information		X			

Figure 7. Example SIMAN V experiment file.