*Proceedings of the 1990 Winter Simulation Conference*
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

# NEAR-TERM DISTRIBUTED SIMULATION OF APPAREL MANUFACTURING

Roy P. Pargas
John C. Peck
Prashant K. Khambekar
Satish K. Dharmaraj

Department of Computer Science
Clemson University
Clemson, South Carolina 29634-1906

## ABSTRACT

This paper describes the design and implementation of a near-term simulator for apparel manufacturing. The primary purpose of the simulator is to provide management with a tool for production planning. In order to run as accurate a simulation as possible, data describing the status of the apparel plant is continually collected by a real-time shop-floor control system. This data is then used as input to the simulator. In order to increase the speed of the simulator and reduce response time, the simulator is implemented on a multiprocessor system consisting of seventeen T-800 INMOS Transputers with a COMPAQ Deskpro 386/25 serving as front-end processor. A user enters a plan for the next production period and receives back more than seventy different measures of performance of shop-floor operations. The user may iterate this process, making modifications to the plan, until the simulated shop-floor performance is satisfactory. The final result is a detailed plan for the next production period.

## 1. INTRODUCTION

Apparel plants with scheduling problems operate on a larger scale than manufacturing plants in many other industries. A typical apparel plant of this type may have more than 400 direct labor (incentive) employees with perhaps 500 machines (some large plants have 1500+ employees and a correspondingly larger number of machines). Both employees and machines are capable of performing multiple operations, but only a small percentage of the total operations are required to manufacture a particular garment. Active on the shop-floor at any one time might be 100 or more production lots (orders) each consisting of perhaps 200 bundles of garment parts, each bundle consisting of five or more subassemblies, each subassembly requiring one to twenty operations. Production lots are possibly of different styles, meaning the operations and sequencing of operations are different. Bundle subassemblies flow through the manufacturing process in parallel and join (merge), as operations are completed, to produce finished garments. The correct matching of subassemblies from parent bundles is important since color shading variations will be noticeable otherwise. Since employees and machines have multiple, but limited, skills and capabilities, load balancing of these resources against required work is a major problem.

The ultimate goal in production scheduling is to improve the operation of a plant. The primary question is "How does one know if a change in an operational plan produces a better or worse schedule?" The performance goals of management, at different plants at different times, may produce very different answers to the question. At one point in time, the goal may to be maximize the output of size 16/34 red button-down collar shirts, whereas at another time, it may be to minimize the work-in-process inventory level subject to keeping the average efficiency of employees above 95%. With different goals, different objective functions, and different management styles, the question then is: how does one build a single management tool to facilitate production scheduling?

This paper describes one attempt at building such a tool: *a near-term simulator of shop-floor operations of an apparel manufacturing plant.* We set four objectives for the simulator:

(1) It should accurately predict near-term performance ("near-term" is loosely defined as one hour to five days).

(2) It should be possible to use the simulator interactively and iteratively. The user should be able to change values of the simulation and observe the effects quickly. Very short response time is therefore necessary.

(3) The simulator should have wide applicability, i.e., it should be usable in different plants with different performance goals.

(4) The simulator should be easy to use, and the results easy to read and comprehend.

The simulator, which is conservative and event-driven [Bryant 1977; Chandy and Misra 1979; Peacock et al. 1979] is described in detail in the rest of this paper. Section 2 shows how one may use a simulator as a tool for production planning. Section 3 describes the user interface, both input and output. Section 4 describes a set of performance metrics available to the user. Section 5 gives an overview of the design and implementation of the simulation. Section 6 discusses conclusions and future plans.

## 2. SIMULATION AS A PLANNING TOOL

Most simulations are beset with one major problem: the difficulty of obtaining actual data on which the simulation can operate. As a result, simulation designers attempt to estimate crucial information such as the rate of arrival of goods to be processed, processing rates of different machines, and skill levels (efficiencies) of employees. The results predicted by the simulation will, of course, be close to reality only if the estimates are accurate. However, coming up with good estimates can often be both difficult and frustrating.

An alternative approach is to measure, in advance, the processes normally estimated by many simulations. For example, in an apparel manufacturing plant, one may measure the skill level of each operator on each different machine type. Or one need not estimate the rate of arrival of goods into the plant if one knows exactly what orders are arriving and at what time. In apparel manufacturing, as in many industries, skill levels vary widely from employee to employee. Assuming that all employees perform at some average rate or according to some statistical distribution may seriously handicap the ability of a simulator to make accurate predictions. One can also take note that, for example, on a given day, a one-hour company-wide meeting will be called to discuss employee benefits; the effect, of course, is that production stops completely for that one hour. If it is known that a high-priority order of goods is to be started in the morning, one may mark the lot associated with the order accordingly, causing the lot to be sewn ahead of others already on the shop-floor. In short, every piece of information that can be measured *is* measured. Estimates of missing data are made only if there is no way to measure the information directly.

This on-going measurement requires that a real-time shop-floor control system be in place. Such a real-time system can collect data through a network of intelligent devices, one at each workstation on the shop-floor. These devices measure and record

a variety of facts, a few of which are:

(1) which bundles have arrived at which operation (this provides accurate tracking of every bundle currently in process),

(2) the number of minutes it takes for Employee A on Workstation 1 performing operation P on Bundle X (this provides one data point which will contribute to a measure of the skill level of this particular employee at this operation, as well as keeps track of the amount of work this employee has done),

(3) the number of bundles on the floor (providing a global view of the number and distribution, across the shop-floor, of bundles of parts),

(4) the number of minutes a particular workstation has been idle (giving up-to-the-minute information on utilization of equipment).

The data are transmitted from the workstations, through the network, and are collected on a PC for viewing or for processing (for example, to produce a payroll at the end of each day or week).

The real-time system employed in this study is one developed and marketed by Foxfire Technologies Corporation [Foxfire Technologies Corporation 1989]. This system, currently installed in a number of apparel manufacturing plants across the country, essentially provides detailed information on every employee, every lot, and every workstation on the shop-floor.

## 3. THE USER INTERFACE

The simulator, a block diagram of which is shown in Figure 1, operates as follows. The user defines a plan for operation of the plant in the near-term, i.e., for the next one or several days. The starting point for the plan is the current state of the plant as represented by data provided by the real-time system. The user then specifies changes to the plan which he or she wishes to implement at given times. New work may arrive at midmorning, personnel may be reassigned to different operations at specified times, machines may be taken out of service for preventive maintenance, new machines which operate at greater speeds may be placed into (simulated) service, etc. With this new plan for operation (usually a small variation of the real plan currently in effect) the simulation begins execution. The operator can request that the plant be simulated until a specified point in time, for a specified duration or until an interrupt key is pressed.

Figure 2 shows the instructions available to the user when developing an input plan. The parameters associated with each are not shown, in order to reduce clutter. The selected instructions are entered by the user and saved in a file which serves as input to the simulator. The instructions have been grouped by function. In the first group, Initialization, there are instructions to start and end the simulation at specified (simulated) times. Instruction 3 names the files produced by the real-time system. These files provide information on the current status of the plant. Instructions 4 and 5 inform the simulator of changes in personnel. For instruction 5, it is necessary to specify which workstation the now-present employee will be assigned to; this is because the simulator must look up this employee's efficiency on the specified workstation in the database provided by the real-time system.

The next group of instructions inform the simulator of events that will occur during the day. For example, (6) a new lot will arrive at a certain time, (7) a meeting is scheduled later in the day, or (8) a regular plant-wide break will take place. Instructions 9 and 10 schedule events relating to specific employees; an employee is paid at a different rate as a result of employee training, or an employee is moved from one workstation to another because of anticipated bottlenecks in the workflow. Instructions 11 and 12 allow changes made to a workstation. Instruction 13 modifies the lot priority, allowing it to move faster or slower through the plant.

Instructions 14 through 17 provide the user with control over the simulator itself: pausing, restarting, and stopping. Instructions 18 through 28 give a variety of information on employees, workstations, and pay code tables. Buffers refer to the queues of bundles of parts waiting to be sewn. Finally, instructions 29 through 31 are warning signals for which the user may request. The user may want to be alerted when the simulator signals that an employee has no more work, that an entire buffer (which typically provides bundles of parts to several workstations) is empty, or that an empty buffer has just received a new set of bundles.

During the simulation, a set of approximately seventy-five performance metrics (Section 4) are maintained. These metrics measure different aspects of plant operations displayed as column graphs. The user may select up to four of these for graphical display on the computer monitor (Figure 3). The user has available the typical graphics window capabilities: opening and closing of windows, bringing windows to the front, line scrolling up and down, page scrolling, moving windows around the screen, and resizing. One window is used for input and enables the user to enter a plan, whereas the rest are output windows showing a variety of information (see items 18 through 28 of Figure 2). User input menus prompt the user for specific information whenever needed. Finally, a help window gives information on all other windows. All window graphics routines are implemented with Borland's Turbo-C graphics library calls.

As the simulation progresses, the display graphics are dynamically updated. If the simulation operator is dissatisfied with the values of certain metrics, he or she may choose to restart the simulation after making changes to the current plan. In this interactive and iterative manner, a plan which best suits management can be developed. This plan can then be printed and used as a recommendation to shop-floor supervisory personnel. We believe that in the beginning a user will experiment with different sets

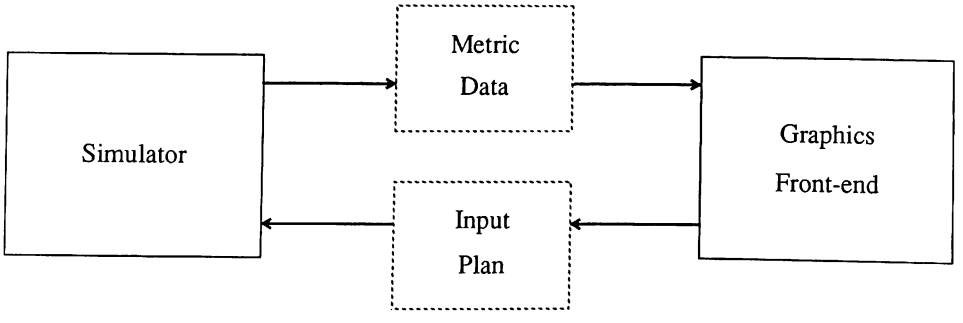


**Figure 1.** Block Diagram of Simulator

INITIALIZATION
1. Start time
2. End time
3. Input data files (RTS files)
4. Employee present during last period (yesterday) but absent now
5. Employee absent during last period (yesterday) but present now

SHOP-FLOOR MANAGEMENT
6. New lot arrival
7. Schedule meeting
8. Set time of break
9. Change pay code
10. Move employee to another workstation
11. Add an operation to a workstation
12. Delete an operation of a workstation
13. Change priority of a particular lot

SIMULATOR CONTROL
14. Stop immediately
15. Resume simulation
16. Stop at a given time
17. Restart

LISTS
18. Show list of off-standard emplyees
19. Show list of on-standard emplyees
20. Show list of idle employees
21. Show list and status of all employees
22. Show list of skills and efficiencies of an employee
23. Show list of workstations attached to a buffer
24. Show list of empty buffers
25. Show list of idle workstations
26. Show list of assigned workstations
27. Show list of machines and operations of a workstation
28. Show Pay Code Table

INTERRUPTS
29. Inform user that an employee is idle
30. Inform user that a buffer is empty
31. Inform user that work has arrived at a previously empty buffer

**Figure 2.** Commands Available to the User

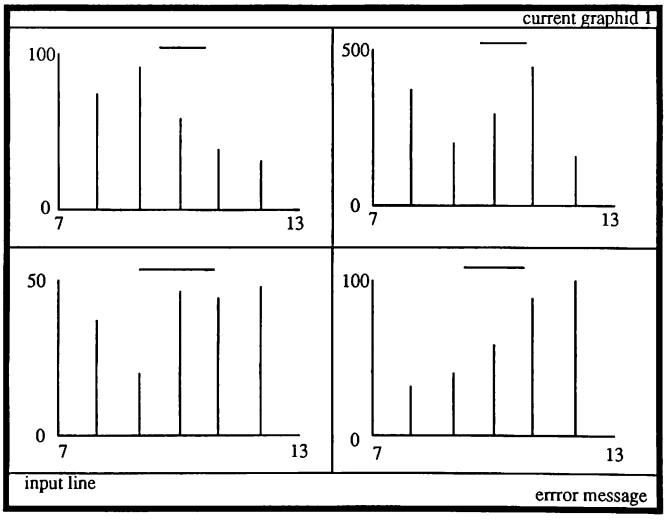


**Figure 3.** The Normal Graphics User Interface

of metrics, but that over time, he or she will settle on one set which will be used thereafter.

A useful time to use the simulator is at the end of a shift, or just before a shift starts. During this period, the user has time to work with the simulator and plan the operations for the next shift. However, because a snapshot of the current status of the shop-floor is available at any time (a benefit of using a real-time shop-floor control system), it is possible for a user to run the simulation at virtually any time of the day using as the current snapshot of the plant that information most recently provided by the real-time system. The simulation gives the user a tool for quickly predicting the performance of the shop-floor during the remainder of the current production period.

In summary, this simulator provides the user with a tool for developing a production scheduling plan interactively. The manner one uses this tool is very similar to the manner one uses a spreadsheet program. With a spreadsheet, the user can change a single cell, and every other cell whose value depends upon it changes automatically. The user can immediately see the "bottom line" result of a single change. Similarly, the user of this simulation can change one part of the current production plan (say, replace one employee with another on Workstation 1 at 10:00 a.m.) and be able to see, very quickly, the effect of the change on overall production for some future period of time. The feedback the user receives is the performance metric information selected for viewing. If he or she is satisfied with the new results, the user

may opt to keep the employee change in the plan. Otherwise, the user may delete the change and may continue trying other modifications.

## 4. PERFORMANCE METRICS

One design decision was to provide a wide variety of performance metrics for the user to select and view. The reason for this is the belief that different managers will want to optimize different metrics. Some may want to maximize workstation utilization. Others may try to minimize amount of work-in-process inventory. Still others prefer minimizing employee idle time. As a result, the best approach is to provide all of the metrics and let the user select those which he or she wishes to observe and improve. We expect that after some experience with the simulation has been gained, the user will settle on a subset of the metrics thereafter. As an aside, note that some metrics conflict and thus cannot be optimized simultaneously. For example, in order to keep employees working on production, it may be necessary to increase the amount of work-in-process.

The metrics used in this study were derived largely from concepts developed for job-shop scheduling, as described by French [1982] and Rinnooy Kan [1976]. Metrics are provided at four different levels: lot (a single production order), style, department, and plant. Seven classes of metrics are available to the user:

(1) Waiting time. Bundles of parts generally wait in queues before they are sewn. A manager may be interested in knowing whether a particular lot (composed of several bundles) is being inordinately delayed or whether employees should be moved to operations where excessive work has accumulated.

(2) Cost/Value. As a lot moves through the plant, it accumulates value. Each sewing operation adds labor value. Machine breakdown, lack of work, employee training, and the like, add excess cost to plant operation but add no value to the product.

(3) Flow time. This measures the amount of time a lot is on the shop-floor. One may compute the percentage of the time jobs are waiting in queues, or the average amount of dollar-value per minute each lot accrues while on the shop-floor.

(4) Lateness. Being able to estimate by how many days a lot will be late (or early) allows a user to plan ahead. For example, he may decide to give a higher priority to a lot to allow it to move along through the plant. On the other hand, a lot that will be completed too early may require unwanted inventory handling. The user may decide to hold processing on an early lot for a day or two.

(5) Labor utilization. Significant labor underutilization indicates several possible problems: workload imbalance, suggesting that the idle employees be trained to perform other operations; too little work-in- process, suggesting that more new orders be started; or simply too many employees for the current work available, suggesting that fewer employees be scheduled for the next several days.

(6) Production. This provides the user with a variety of measures for the amount of work produced and the rate at which it is produced.

(7) Efficiency. This provides a comparison of the actual output produced by the plant and the maximum possible output of the plant.

For a detailed description of the complete set of metrics used, the interested reader is referred to a paper by Peck et al. [1990].

## 5. DISTRIBUTED SIMULATION

A major concern in this simulation is simulation response time. A simulation of a large apparel manufacturing plant with 500 or more workstations is anticipated to require much more computing power than is available on a PC, even on the latest models such as the IBM PS/2 family, some of which use the very fast Intel 80386/387 processors. For this reason, we decided to implement the simulation on a distributed memory multiprocessor system.

The multiprocessor system used was built by Computer System Architects of Provo, Utah. It consists of seventeen INMOS T-800 Transputers, each with two megabytes of memory. The processor integrates a 32-bit processor, a 64-bit floating point unit, and 4 Kbytes of static RAM. At peak speeds, each processor provides 10 MIPS and 1.5 Mflops. Each has four communication ports which can be used to interconnect with other processors. Sixteen processors, therefore, can be linked to form a linear array, a ring, a mesh, a 2-D torus, or a 4-D hypercube. One of the Transputers is called the root processor and serves as a liaison between the front-end PC and the sixteen other Transputers, called nodes. The front-end processor, which accepts input from and provides graphics output to the user, is a standard PC, a COMPAQ Deskpro 386/25, with a 110 Mbyte hard disk.

A major design decision was to break up the major functions of the simulation among the processors available. The primary functions are input and output, execution of the simulation itself, sending of user input to the simulation, and collection and processing of performance metric data (Figure 1). A natural assignment of function to processor is to assign all input/output function to the front-end PC, the simulation to the Transputer nodes, and to let the root Transputer serve as a liaison, collecting metric data and broadcasting user input to the nodes.

The responsibilities of the front-end computer are to provide the interface allowing the user to enter and modify plans, and to display all metric information in graphical format. All shop-floor status information is stored in a database which resides on the PC hard disk. The root Transputer collects and processes performance metric data before sending the data to the front-end for display. As each node steps through the simulation, it accumulates performance metric information and, at predefined time intervals, sends the data to the root Transputer. The root collects the data, performs a few simple computations (such as computation of means and variances), and when the data for the time interval is complete, sends a packet of metric information to the front-end PC for graphic display.

This division by function is quite clean. The user sees only the front-end computer and does not know about the existence of the root or node Transputers. The program executing on the front-end is unaware of the number of processors running the simulation. It is only aware of simulator commands and data it sends down the communication link to the root Transputer, and metric and other information received from the root. In the same way, the root program is independent of the number of Transputer nodes and the nodes depend on the root only for startup information. The simulation may run on one node Transputer, or as many as the user can afford. For smaller problems, one Transputer may suffice. For larger problems with 500 or more workstations, the user may decide whether the increased speed of execution justifies the cost of additional processors. The design is flexible enough, however, to accommodate any number of node Transputers.

The Transputer nodes execute an event-driven simulation. Because physical memory is limited on each node Transputer (two megabytes per processor) and virtual memory is not provided by the node operating system, we have opted for a conservative simulation, rather than the optimistic Time Warp approach proposed by Jefferson [1985]. The most common event is completion of a sewing operation on a subassembly. Other events include lot events (arrival of a lot, change in a lot's priority), employee events (employee is assigned to another workstation, employee stops work to attend a meeting), and workstation events (a workstation is reconfigured for a different sewing operation).

Each sewing operation has a unique buffer which holds a queue of subassemblies waiting to be sewn. One or more workstations, configured to perform the operation, pick subassemblies from the buffer in a first-come, first-served order. A processor, one of the node Transputers, simulates the activity of one or more buffers. For correct first-come, first-served simulation, the workstations must coordinate. This is most easily achieved by re-

quiring that all workstations that pick from a common buffer be simulated in a single processor. Each processor has a single process, a single event queue, and a single logical clock. The event queue is stored in ascending order and, as a result all events within (and resultant messages from) a processor are in logical time sequence. Each node processor goes through the event queue, generating new events for itself and other processors, until the end of the simulation period.

Subassemblies may flow from a buffer in one processor to a buffer in a different processor. Buffers in a processor which can receive subassemblies from another processor are called Front Buffers. For correct conservative simulation, a processor cannot proceed if there is a possibility of receiving a subassembly message from a predecessor processor at an earlier clock time. This means that if correct time synchronized simulation is to be guaranteed, all Front Buffers of a processor must be non-empty (other buffers may be empty) before a processor advances its simulation clock. One way to handle this is for predecessor processors to send NULL messages to successor processors. However, source-driven NULL messages are likely to flood the system, as reported by Fujimoto [1988]. An alternative approach is: if the Front Buffer of a processor becomes empty, the processor sends out appointment-request messages to its predecessors. The predecessors then make an estimate based on their current statuses and send appointments to the requesting processor, thereby enabling it to proceed. This demand-driven method, described by Khambekar and Dharmaraj [1990], is an adaptation of the appointment approach presented by Nicol and Reynolds [1984] and Nicol [1988]. In terms of the design space outlined by Reynolds [1982], this method is accurate, non-aggressive, has no risk, and employs knowledge acquisition and knowledge embedding.

Subassemblies have user-assigned priorities and are arranged in the buffer queues according to their priorities and arrival times. Occasionally, two or more subassemblies must merge into one. For example, fronts and backs must merge to form a complete shirt. Hence, the mere arrival of a subassembly in a buffer does not guarantee that it is ready for processing; it may have to wait for its companion subassembly to arrive at the same buffer. Information on the flow of companion subassemblies is extracted and stored in compressed form prior to the start of the simulation. A simulation node ready to select a subassembly must scan the buffer queue, skipping over all unmatched companion subassemblies.

The service time for an operation in a subassembly is calculated from the pre-engineered Standard Allowed Minutes (SAMS) and the efficiencies of the employees. For example, if the SAMS value of an operation is 5 minutes and an employee has an efficiency of 110% (better than average), then the service time for this employee working on this operation is $5/1.1 = 4.55$ minutes. The completion time of the simulated operation is computed and an event is added to the event queue.

Subassemblies usually flow from one operation to another; however, sometimes there are alternate paths available for the subassembly. For example, instead of performing two consecutive operations on two slow older machines, it may be possible to combine both operations on one fast, newer machine. When alternate paths are available, a subassembly is sent along the path with the smallest wait-queue. This implies that the path of the subassembly must be determined at runtime, i.e., determining the successor buffer (which buffer to send the subassembly to next) is determined on-the-fly by examining the wait-queues of all the successor buffers. If the selected successor buffer happens to be in the same processor, then the subassembly is sent directly to the buffer; otherwise, additional table look-up is necessary to determine the successor processor, and a subassembly message is sent to that processor.

When an operation completion event occurs, the processor calculates all contributions to the collection of metrics. Results are stored in local memory. After a predetermined interval of simulated time, the processor sends the accumulated metrics to the root Transputer. The root accumulates these in its own global set of metrics. When all processors have sent metric values for a single time interval, the root computes a set of secondary metrics (means, variances, plant-wide totals) and sends the entire list of metric information to the host PC for immediate display.

## 6. SUMMARY

This paper describes a near-term simulator for use in scheduling production in an apparel manufacturing plant.

In the Introduction, we set four objectives for the simulator: accuracy of simulation, interactive and iterative use, wide applicability and ease of use. We feel these goals are being met. This simulation is accurate because little information is estimated. Current and historical data obtained from a real-time control system are used, rather than statistical distributions to estimate such things as rate of arrival of goods and employee skill level. Interactive and iterative use is due, in large part, to the division of tasks among multiple processors. The very short response time which results encourages the user to sit before the monitor and work with the simulator. Wide applicability is achieved because of the large number of performance metrics made available to the user. The user selects as many of the metrics as he or she feels are important in planning shop-floor operations. Finally, ease of use is a result of the graphics windowing system available; the user views many performance metrics changing over time.

We believe that this simulation will be a valuable tool for a manager of shop-floor operations.

## ACKNOWLEDGEMENT

This research was sponsored in part by the Defense Logistics Agency under Contract No. DLA900-87-D-0017 Task No. 0003 through the Clemson Apparel Research Center.

## REFERENCES

Bryant, R.E. (1977), "Simulation of Packet Communication Architecture Systems," Technical Report MIT/LCS/TR-188, MIT, Cambridge, MA.

Chandy, K.M. and J. Misra (1979), "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs." *IEEE Transactions on Software Engineering SE-5,* 5, 440-452.

Foxfire Technologies Corporation (1989), *Real-time Shop-Floor Control System User Manual,* Marietta, GA.

French, S. (1982), *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop,* Ellis Norwood, London.

Fujimoto, R.M. (1988), "Performance Measurements of Distributed Simulation Strategies," In *Proceedings of the 1988 SCS Multiconference on Distributed Simulation,* San Diego, CA, 14-20.

Jefferson, D.R. (1985). "Virtual Time," *ACM Transactions on Programming Languages and Systems 7,* 3, 404-425.

Khambekar, P.K. and S.K. Dharmaraj (1990), "Approaches to Solving Synchronization Problems in Parallel Simulation of an Apparel Plant," In *Proceedings of the 1990 ACM Southeast Regional Conference,* C.M. Pancake and R.M. Geist, Eds. ACM, Greenville, SC, 274-281.

Nicol, D.M. and P.F. Reynolds (1984), "Problem Oriented Protocol Design," In *Proceedings of the 1984 Winter Simulation Conference,* S. Sheppard, U. Pooch, and D. Pegden, Eds. IEEE, Dallas, TX, 471-476.

Nicol, D.M. (1988), "Parallel Discrete-Event Simulation of FCFS Stochastic Queuing Networks," In *Proceedings of the 1988 ACM SIGPLAN PPEALS,* Yale University, New Haven, CT, 124-137.

Peacock, J.K., J. W. Wong, E.G. Manning (1979), "Distributed Simulation Using a Network of Processors." *Computer Networks 3,* 1, 44-56.

Peck, J.C., R.P. Pargas, P.K. Khambekar, and S.K. Dharmaraj (1990), "Shop-Floor Performance Metrics for the Apparel Industry." Submitted to the *International Journal of Clothing Science and Technology.*

Reynolds, P.F. (1982), "A Shared Resource Algorithm for Distributed Simulation." In *Proceedings of the Ninth Annual International Computer Architecture Conference,* Austin, Texas, 259-266.

Rinnooy Kan, A.H. (1976), *Machine Scheduling Problems: Classification, Complexity and Computations,* Martinus Nijhoff, The Hague.