Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

IMPLEMENTING ON-LINE SIMULATION UPON THE WORLD-WIDE WEB

Wayne J. Davis
Xu Chen
Andrew Brook

General Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

ABSTRACT

This paper addresses the implementation of advanced, on-line simulation experiments upon the World-Wide Web. The paper first discusses the system considered for this study. The paper then discusses the distributed object computing environment that has been employed in the study, including the specialized displays that would be accessed by a remote participant when viewing the on-line simulation analysis on the Web. Finally, directions for future research are outlined.

1 INTRODUCTION

This paper demonstrates that it is now possible to project the near-term response of a system while operating under one or more alternative control strategies, given the current system state. In addition, it is demonstrated that it is now possible to implement such on-line simulation analyses in a manner that will permit the projected system response to be concurrently viewed by several participants at remote sites on the World-Wide Web.

We begin by discussing the system considered within the experiment. The model for this system was programmed in Java. Next, the distributed object-oriented computing environment that has been constructed for implementing the experiment is discussed. In particular, the displays that a remote viewer will interact with upon the Web and the manner in which the simulating objects provide information to these viewing objects are detailed. The paper closes by citing future research directions.

2 THE SYSTEM

In the past, simulation analyses have been employed primarily to project steady-state system performance in order to support off-line planning. In these analyses, explicit procedures, such as the employment of a warm-up period, are often adopted in order to keep from considering the transient system dynamics resulting from initializing the simulation trial to a given state. In this paper, we will discuss on-line simulation analyses, which explicitly consider the near-term transient performance of a system as it evolves from its current state. In Davis [1998], considerable discussion was devoted to contrasting off-line simulation and on-line simulation analyses. This paper will not replicate that discussion. Rather, it focuses upon implementing the on-line simulation upon the Web.

The experiment employs a model of the Automated Manufacturing Research Facility (AMRF) which was constructed by the National Institute of Standards in Gaithersburg, Maryland (see Figure 1).

The AMRF was constructed as an experimental flexible manufacturing cell (FMC) consisting of four primary workstations: vertical milling, horizontal milling, turning and cleaning/deburring. Jobs are delivered to the stations

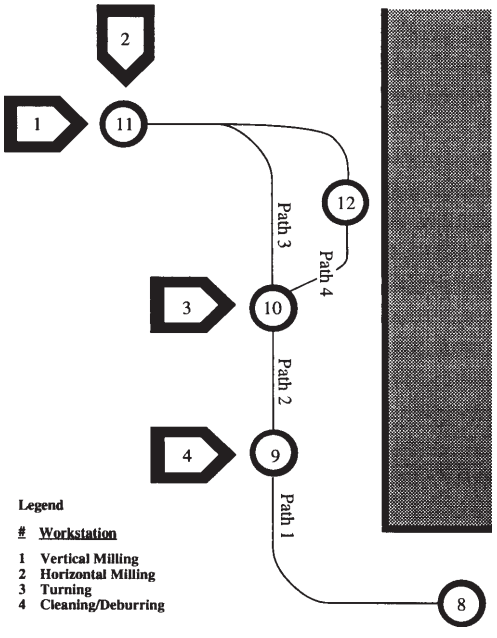

Figure 1: Schematic Layout for the AMRF

via an Automated Guided Vehicle (AGV) system. The AGV's path is illustrated in Figure 1 as the set of arcs connecting nodes 8 through 12. As shown, Node 11 represents the entry/exit point to stations 1 and 2. Node 10 and 9 represent the entry/exit point for stations 3 and 4, respectively. Node 8 represents the entry/exit point to the entire cell. Incoming jobs can be stored here until they are dispatched into the cell. Finally, node 12 represents a position where a single AGV can be parked while its battery is being recharged.

The AGV system employs two AGVs. Looking at the cart path that connects nodes 8 through 12, it is clear that there are no major loops for allowing the AGVS to pass each other. Hence, the carts must operate in both directions upon the indicated cart path, including path segments 1 through 4. Obviously, the potential for deadlock exists. In order to prevent deadlock, a Petri-network control logic diagram was developed for allocating the path segments to the carts as they move between nodes comprising the AGV path network (see Figure 2). In order for a cart to move from node 8 to node 10, the cart must be at node 8 (requiring a token to be at place 8) and take ownership of paths 1 and 2 (requiring tokens to be at both places 1 and 2). When the cart arrives at node 10, it releases ownership of path one, which returns a token to place 1. Note that no cart can reside at nodes 9 through 12 without owning at least one path. The logic depicted here is rather complex, but it is also essential to prevent deadlock. The employed Petri-net control logic diagram has been tested for liveness in order to guarantee that there are no terminal states (i.e. deadlock cannot occur).

A detailed discussion of the developed simulation model is beyond the scope of this paper. The reader is referred to Tirpak, Deligiannis, and Davis [1992] for more details. The original simulation model for the AMRF was reprogrammed in Java in order to make greater use of Web-based computing procedures.

3 THE ON-LINE SIMULATION EXPERIMENT

In implementing our on-line simulation experiment, the model for the AMRF was used to perform a real-time emulation of the system. In this mode of application, the model is executed as a general simulation model, but with one major exception: no event is processed until real-time equals the event time. Specifically, after each event is processed, the next event is removed from the event calendar and its event time is recorded. The emulation then pauses until the real-time is equal to the event time for this next event. When this condition is satisfied, the next event is processed. The emulator can also employ time scaling which permits its clock to advance faster than real-time. Typically, we operate the emulator at 10 to 100 times real-time.

The real-time emulation is actually unnecessary to the experiment. That is, we would prefer to replace the real-time emulation with a real-world system. As shown in Figure 3, the real-time emulator (or real-world system) posts its state with the server as each change in state occurs. Alternately, the system's current state could be posted at regular time intervals (e.g. every 10 seconds). Hence, the client serve becomes the central repository for the current state information of the emulated (or real-world) system.

As shown in Figure 3, several remote clients can log into the on-line simulation experiment. Whenever a remote client logs into the central client/server, a set of viewing applets is downloaded to the remote site. The first of these applets produces the Timed Simulation Display, shown in Figure 4, which displays the detailed current state

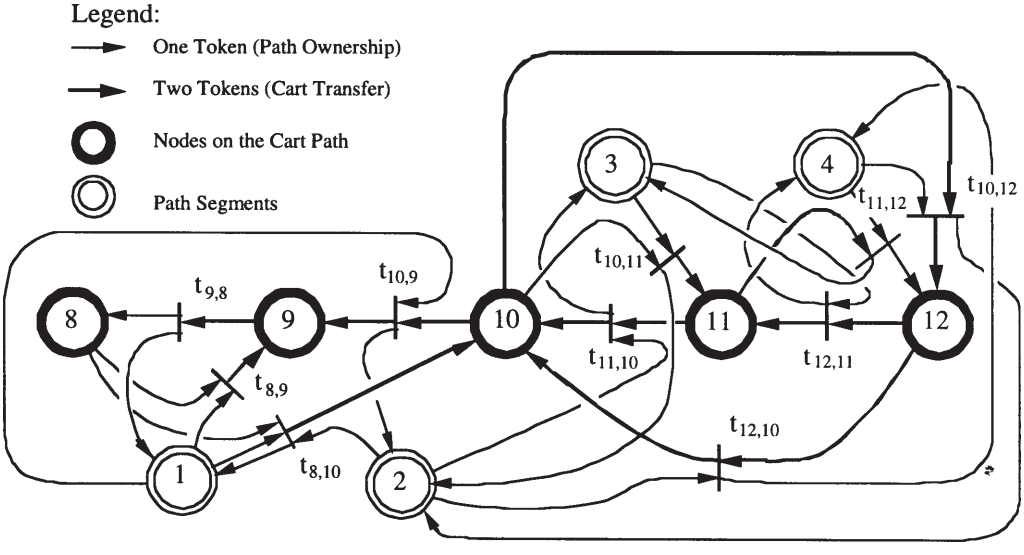

Figure 2: Petri-Network Control Logic for Allocating the Path Segments during AGV Transitions

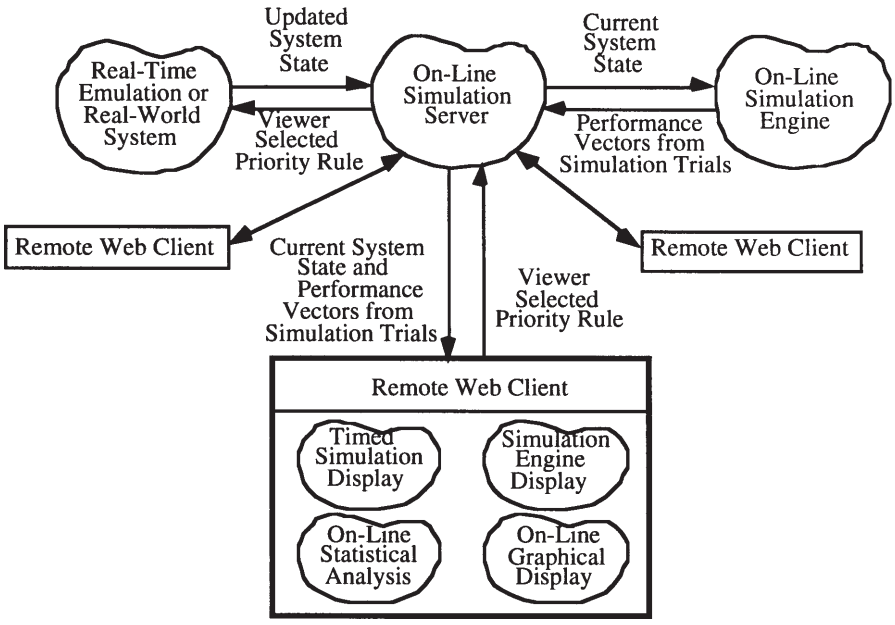

Figure 3: Organization of the Distributed Computing Objects that Comprise the On-Line Simulation Experiment

information for the emulated (or real-world) system.

At the top of the display, the Time Now field (=2926.656) gives the amount of emulated time that has past since the beginning of the experiment. Next, the current number of jobs that are in the system (=48) is given along with the minimum job number (=212) and the maximum job number (=264) for the jobs residing in the system.

The next major section provides detailed state information for each of the four primary workstations. For each workstation, the number in its input and output queues is recorded along with the job numbers of the first two jobs in the input queue and the output queue based upon the priority scheme employed to order the jobs in each of the respective queues. In this case, there are 14 jobs in the output queue for Station 1 with jobs 226 and 227 occupying positions 1 and 2 in the buffer. For each workstation, the number of the job, if any, that is currently being processed is given. For example, Station 2 is currently processing job 246.

Below the display for the state of the workstations is the detailed state information for the AGV system. To the left of this section of the display is the detailed state information for each of the AGVs. Included in this state information, is the current node where the AGV resides, the destination node to which it is traveling, and the next node that it will visit as it moves toward the destination node. There are two additional state variables defined for each AGV. The first of these is the AGV's job status variable, which defines the next job that the cart has been assigned to process. The second is the variable that defines the job that is currently on-board the AGV. If there is a job on board, then this state variable will have the same value as the status variable, and the AGV delivers the on-board job to the specified destination node. If no job is on board, then the AGV is traveling to its destination in order to pick up the job number specified in its status field.

Below the AGV state field, detailed information is provided for the status of each segment of the cart path. If a given segment of the cart path is currently owned by a given AGV, it will be so noted. If a given segment is currently owned by one AGV and is being requested by another AGV, then the AGV that is requesting that path segment will also be noted. To the right of these state fields, the contents of the queue of jobs requesting transportation is given. The total number of jobs that are awaiting transport is given along with the first seven jobs in that queue based upon the priority scheme that is being employed to order the jobs in this queue. The reader will note that there are currently 45 jobs in the system that are awaiting transportation. We have deliberately provided a state where the entire system has become congested primarily due to an inefficient priority rule for assigning the AGVs to the requesting jobs.

In the bottom left portion of the Timed Simulation Display, the retrospective performance statistics for the current experiment are given. That is, for the 216 (=264-48) jobs that have already been processed during the period prior to the current emulated time contained in the Time Now field (=2926.956), four performance criteria are evaluated. The first criterion is the mean time that each of the completed jobs resided in the system. The second criterion is average productivity for the completed jobs. In order to compute this, the ratio of the total processing time

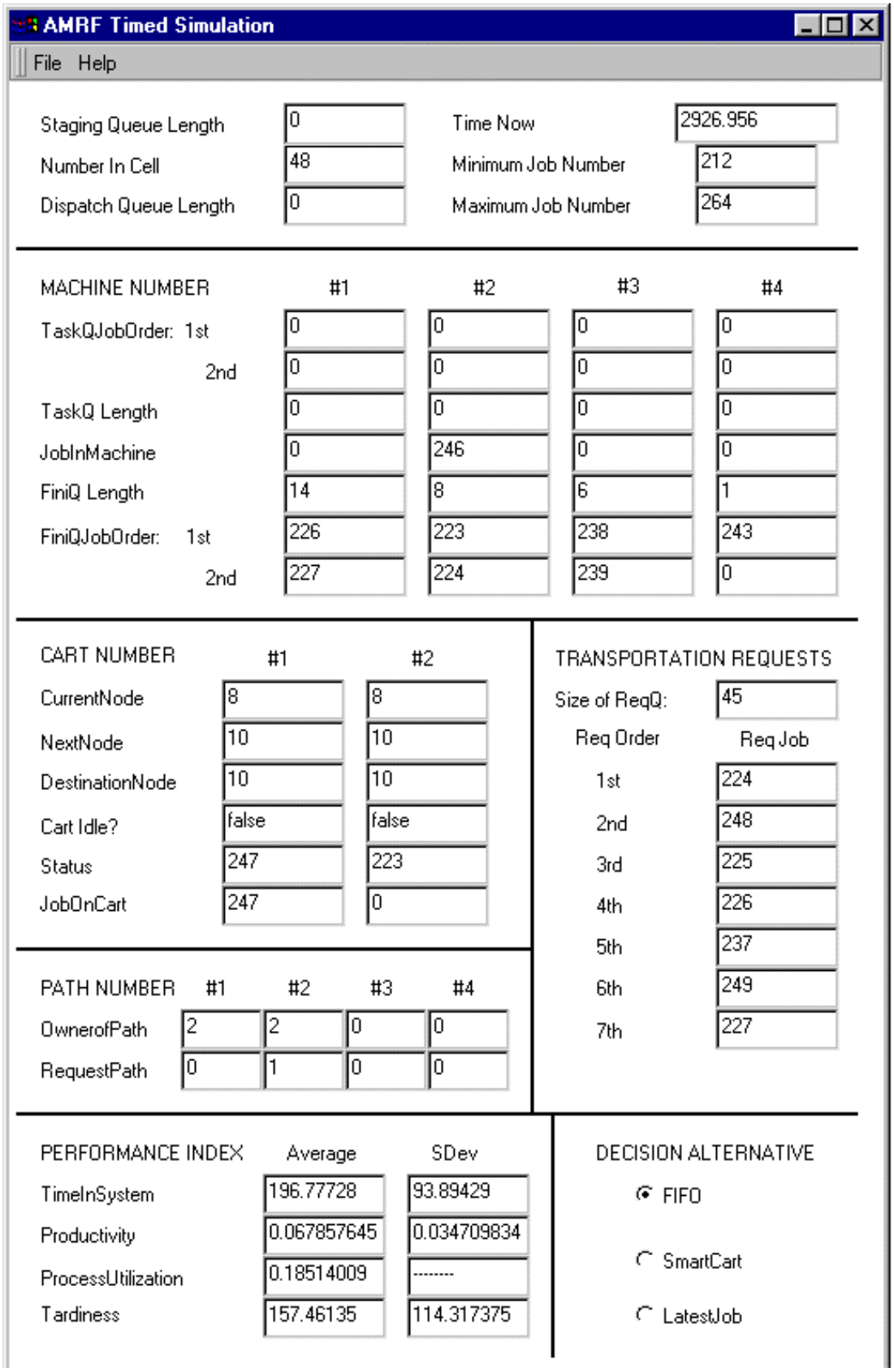

Figure 4: Timed Simulation Display for the State of the Real-Time Emulation

dedicated to each job over the total time that the job resided in the system is first computed in order to determine the individual job's productivity. The productivity ratios for all the jobs are then averaged in order to compute a mean value for the productivity criterion. The third criterion is the average process utilization. Here, the process utilization is first computed for each workstation by computing the ratio of the total time that the process has been busy over the total time for the emulation. The individual utility values are then averaged over the four workstations in order to compute a mean value for process utilization criterion. The fourth criterion is the average lateness of each completed job. As each job enters the AMRF, a due date is assigned for its completion. Based upon the emulated completion time, the tardiness of each completed job is then computed. The individual tardiness values are then averaged for all the completed jobs in order to compute the mean lateness criterion. In addition to the average value for each criterion, the standard deviation is also computed over the sampled values that were used to compute the means. Other statistics such as the minimum and maximum values for each criterion could also be computed if desired.

It is important to observe that all the information contained within the Timed Simulation Display either deals with the current state or the past performance of the system. There is one particular exception to this statement, however. At the lower right section of the Timed Simulation Display, there are three radio buttons which can be selected by the remote viewer. Attached to each radio button is a given priority rule for ordering the jobs that are currently requesting transportation. In this experiment, three priority rules have been included. The First-In, First-Out rule assigns priority based upon the time the given job requested to be transported. The Latest Job rule orders the jobs based upon their assigned completion date. Finally, the Smart Cart rule prioritizes the transportation requests in a manner that minimizes the cart movement necessary to handle the request given the current state of the system.

The radio buttons depict which priority scheme is currently being selected. As shown, the FIFO strategy has currently been selected, and its inefficiency has resulted in the congested system state depicted in Figure 4. Any remote viewer can select another strategy for immediate implementation. For example, a viewer might elect to implement the Smart Cart rule which would likely eliminate the congestion. We could also have included viewer-selectable priority rules for managing the other queues in the system. However, for the purpose of this demonstration, we elected to keep the situation as simple as possible while demonstrating all potential capabilities for the experiment.

Returning to Figure 3, we can begin to appreciate the complexity of the on-line experiment. As illustrated, the real-time emulated or the real-world system is constantly posting updated state information to the Java-based server for the experiment. Whenever an update in the system state occurs, this information is broadcast by the server to any remote viewing applet on the Web. The remote applet then displays the updated state information in its dedicated Timed Simulation Display window. However, the remote viewer can also select any other priority rule for ordering the transportation request queue. If a remote user selects a different rule for implementation, the remote applet sends a message to the central server which then forwards the message to the real-time emulated (real-world) system. When the emulated (real-world) system receives that message, it then changes the priority rule that it used to order the transportation request queue. Note that when this change occurs, the state of the system will also change in reference to the priority rule that is currently being used to order the transportation request queue. This updated state information is then posted with the server which, in turn, broadcasts the information to the remote clients. In this manner, the remote client who requested the change receives feedback that his/her request for a change in the implemented priority rule has been effected.

As shown in Figure 3, the updated state information is also sent to the Simulation Engine object. Earlier, we stated that there were multiple uses for the simulation model of the AMRF. The model was first employed in the real-time emulation. The second use of the model occurs at the Simulation Engine object that addresses the future response of the system. Using the Java-based simulation model, the Simulation Object then projects the future performance of the system (the AMRF) while it operates under each of the three potential rules for prioritizing the transportation request queue. In the demonstration, the simulation projects the future performance of the system as it completes the next 100 jobs that will go through the system. The consideration of the next 100 jobs is totally arbitrary. For example, we could have considered the performance of the system over the next 24 hours. The on-line simulation analyst must specify the planning horizon that is to be considered in the experiment.

As each simulation trial for the next 100 jobs is completed for a given priority rule, the mean Time in the System, the mean Productivity, the mean Process Utilization and the mean Lateness criteria are evaluated for the given trial. (Note that these computed averages can be referred to as prospective performance statistics in contrast to the retrospective performance statistics that are given in the Timed Simulation Display as previously discussed.) After a simulation trial for each priority rule is completed, the computed means for the performance criteria are sent to the server. The simulation object then computes another set of simulation trials for each priority rule where each simulation trial is initiated to the most recent system state.

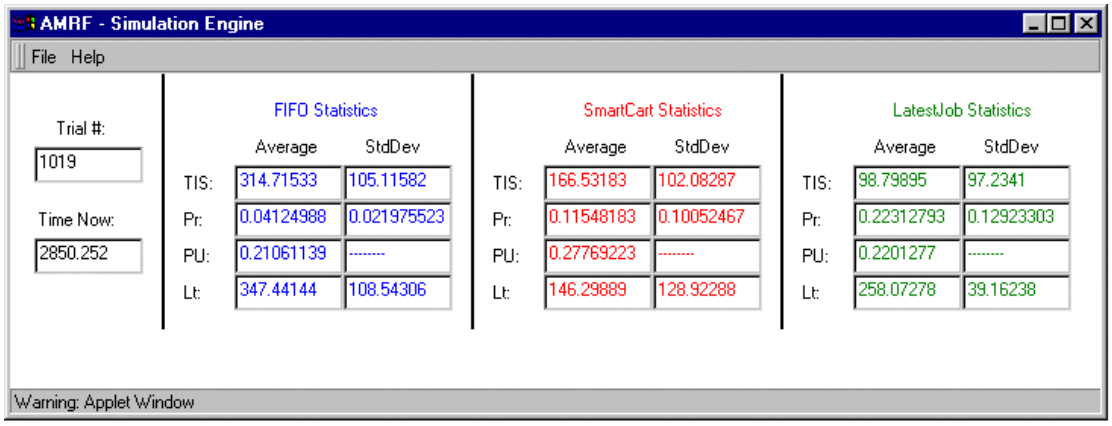

Figure 5: Graphic Display for the Simulation Engine

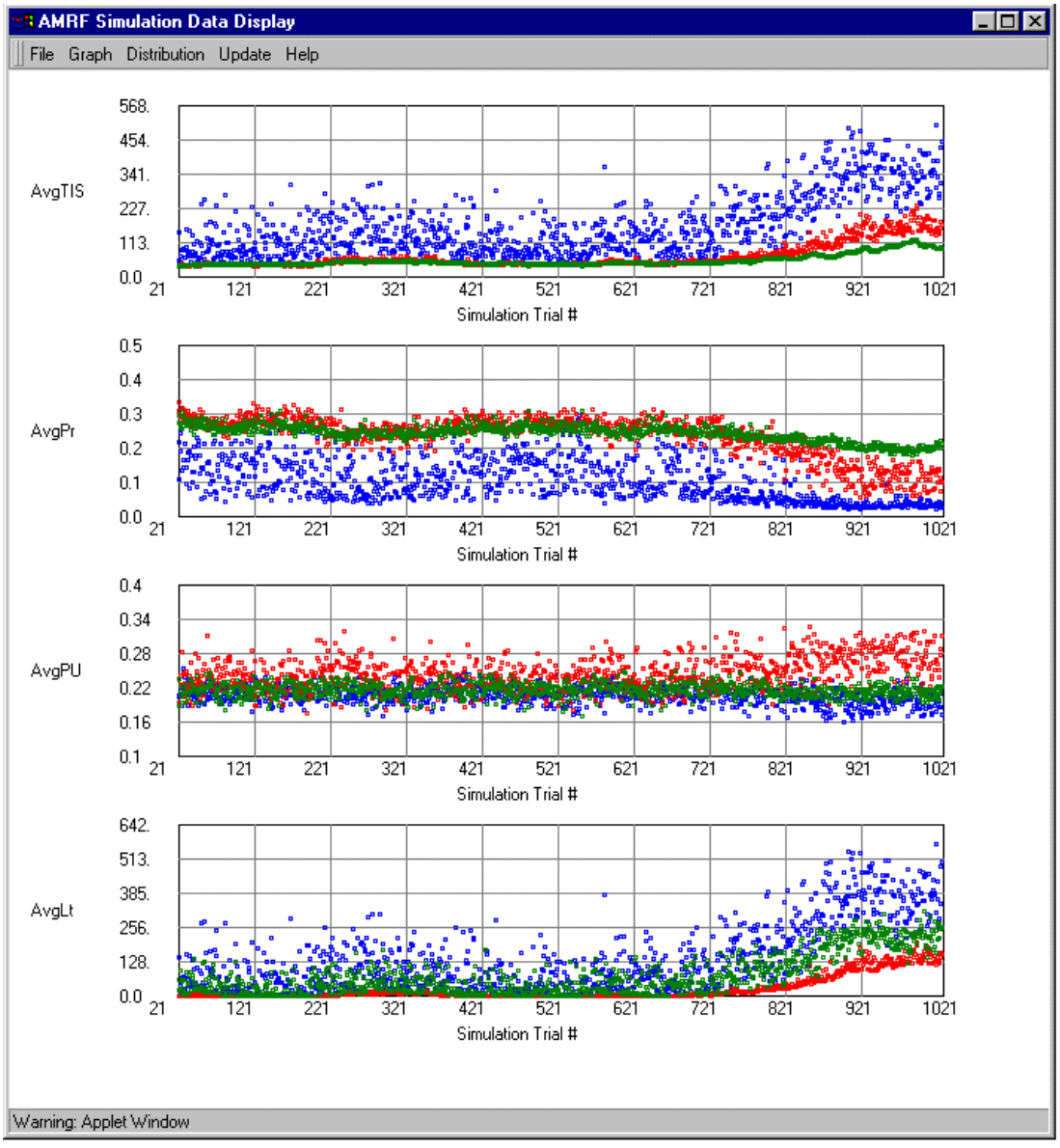

Figure 6: Graphic Display Depicting Performance Criteria Values Stored in the Push Down Stacks

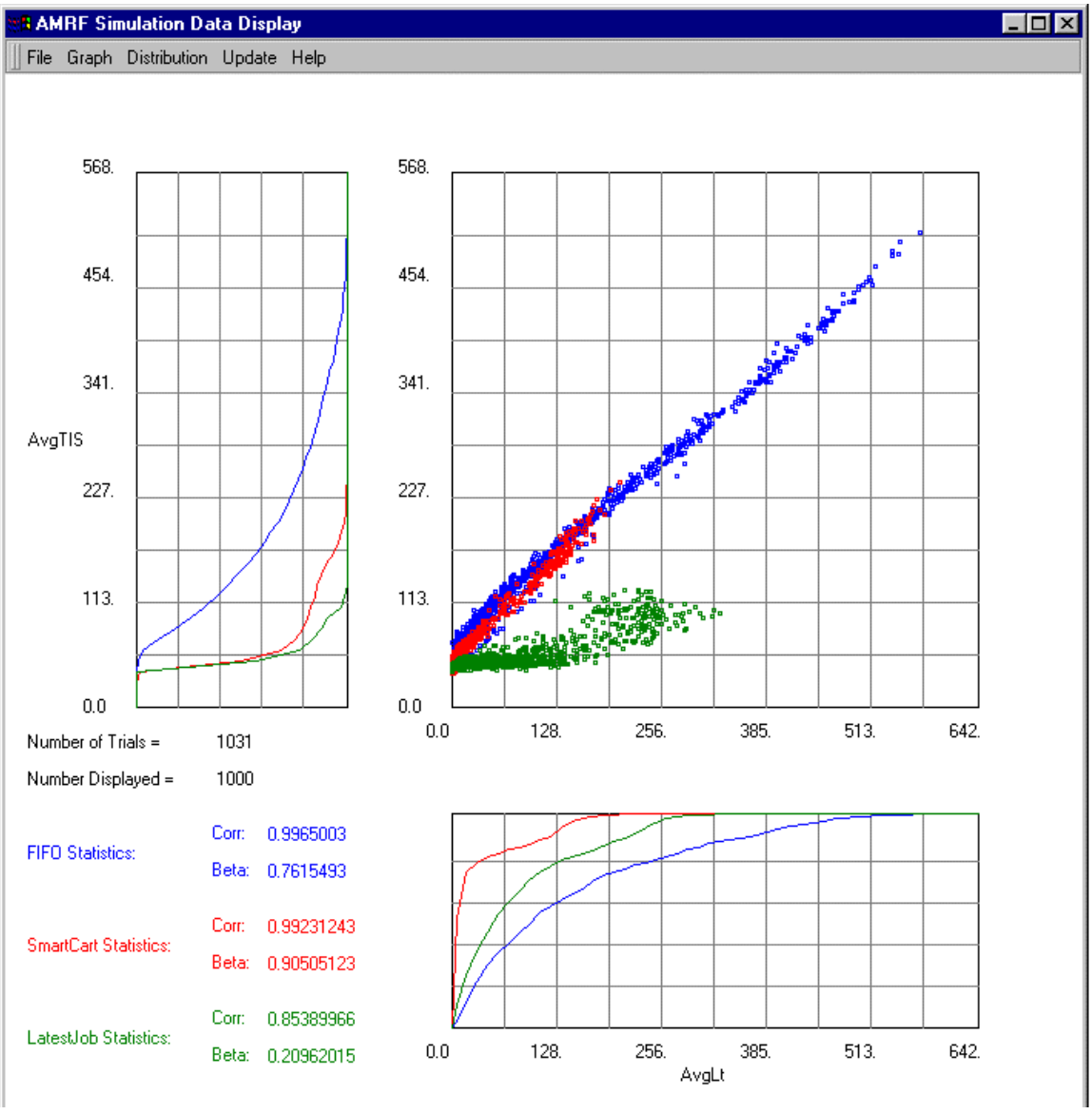

Figure 7: Simulation Data Display Window Depicting Statistical Distribution of Lateness versus Time In System

This process is repeated constantly throughout the on-line simulation experiment.

Whenever the server receives the prospective performance vectors associated with another set of simulation trials from the Simulation Engine Object, the performance vectors are broadcast to each remote site. At the remote site, the applet contains a dedicated display for viewing the output of the Simulation Engine Object (see Figure 5). This display gives the simulation trial number (=1019) and the average value and standard deviation for each performance criterion associated with each priority rule. The information contained in this display is of little practical utility because it provides the performance measures for a single simulation trial. Its primary use is to depict the status of the on-line simulation analysis only. That is, it provides information on the number of simulation trials have been generated since the on-line simulation experiment has been initiated. It also provides visual evidence about the rate at which simulation trials are being generated.

As noted above, the performance measures derived from a single simulation trial are of limited utility. Our desire is to assemble a collection of simulation trials associated with the system operating under a given priority strategy in order to make a more comprehensive statistical estimation of the future performance of the system if a given priority scheme is adopted. To this end, the remote applet collects the results from the past simulation trials that have been sent to it from the central server. In reality, the performance vectors resulting from each simulation trial for each considered priority rule are stored in sepa-

rate push down stacks, one for each priority rule. In this stack, a fixed number of performance vectors are stored with the most recent trials stored at the top of the stack. When the stack becomes full, the oldest performance vectors are removed from the bottom of stack and disposed. Hence, after the stack it is filled, it holds a fixed number of the most recent simulation trials for a given rule. This situation is best illustrated in Figure 6 where the last 1000 simulation trials (e.g. from trial 22 to 1021) for each performance criteria are displayed.

Each remote applet also contains a Java-based, on-line statistical analysis object for computing the statistical results associated with the simulation trials contained in each stack. Specifically, for each performance criteria contained within the performance vectors, the average and the standard deviation of the mean performance values are computed. (Recall that the Simulation Engine Object computes mean values for each simulation trial, e.g. the mean-time in system for the next 100 jobs to be processed by the system.) The recorded values from the set of stored performance vectors are also used to compute an empirical cumulative distribution function for the trial mean values of each performance index. As new simulation trials are added to the stack (depicted in Figure 6), all of the computed statistics are updated in real-time.

Each remote applet also contains a window object (the Simulation Data Display Window) to view the statistics. Here, the remote viewer can specify which display he or she currently wishes to view. In Figure 7, we have provided a snapshot of the display for viewing average Time In System (AveTIS) versus the average Lateness (AveLt) performance criteria. In the upper right quadrant of this window, the AveTIS and AveLt values for each of the last one thousand simulation trials are plotted as a single point in the AveLt versus AveTIS Cartesian space. In this diagram, a total of 3000 points are plotted: 1000 points for each priority rule. In order to distinguish the points associated with a given priority rule, the points have been color-coded. However, it is difficult to distinguish these here as the gray-scale (not color) figures are presented in this paper. To the left of, and below, this primary scatter plot, the empirical cumulative distribution functions for the average values of the displayed performance criteria that have been plotted. Again, a single empirical cumulative distribution function is provided for each priority rule. In the lower left quadrant of the display, we provide the correlation coefficient among the displayed performance criteria as well as the slope of the best linear regression line through the plotted data. Again, the values are computed for each priority rule.

Assuming that we are considering a total of four performance criteria (i.e. average Time In System, average Productivity, average Process Utilization, and average Lateness), a total of six such plots can be generated–one for each possible pair of performance indices. The Graphical Display window permits the user to choose which pair of performance criteria he or she wishes to view at any given moment.

4 FUTURE RESEARCH

In this paper, we have provided only a basic description relating to the basic operation of the distributed on-line simulation experiment as it is implemented upon the World-Wide Web. Another paper could be written on the software implementation of the experiment. Currently the real-time emulation, the on-line simulation engine, and the central server are all executed upon a single personal computer operating under the Windows-NT operating system. If more speed is needed, these functions could be distributed across a network of personal computers in our laboratory.

All the software that is needed to implement the experiment was written in Java by programmers in our lab, including the server's programs which manage the distributed computational environment. It may have been desirable to employ commercial software for several of the functions (including the implementation of the server), but our research budget would not permit us to make such purchases.

Obviously, the current experimental setup is still in the prototype phase. Future research must develop a standardized architecture for implementing these experiments. It must also make use of existing standards whenever they are available. The goal of this experiment was not to solve the general implementation problem for all future on-line studies. Rather, the goal was simply to demonstrate the evolving simulation capabilities that can be provided in the future

The area of on-line simulation is certainly in its infancy and needs much further investigation. In Davis [1998], future research needs are discussed in much greater detail. Clearly, considerable effort is needed to provide better statistical analysis tools for analyzing transient response. Returning to Figure 6, it is clear that the statistical performance being projected by the most recent trials is significantly different from that of the earlier trials. Thus, we conclude that the system is undergoing a transient behavior which could possibly lead to its instability as the system becomes more congested.

However, a major source of this instability is the inefficient priority scheme (i.e. First-In, First-Out) that is employed to allocate the AGVs to the jobs that require transportation. To use a more efficient priority scheme is likely to return the system to a more stable configuration.

The selection of the priority rule for implementation opens an entirely new area of research associated with the development of an on-line intelligent controller for man-

aging the system. Such a controller would first need to determine which control strategies should be considered for possible implementation, perform the essential on-line simulations that are needed to assess their potential performance, compare their projected responses in order to determine which strategy should be implemented and then implement the selected strategy. The need to address all of the requirements is constant. Thus, all of these functions must be addressed concurrently while the system is operating. The structure for such an intelligent controller has not yet been specified and represents a major research need. In short, we can conclude that much more research is needed. On the other hand, it is obvious that there is a bright future for research in the simulation area.

REFERENCES

Davis, W. J. 1998. On-Line Simulation: Need and Evolving Research Requirements. In *Handbook of Simulation*, ed. J. Banks, 465-516. New York: John Wiley and Sons, Inc.

Tirpak, T. M, S. J. Deligiannis and W. J. Davis. 1992. Real-Time Scheduling in Flexible Manufacturing. *Manufacturing Review*, 5(3): 193-212.

AUTHOR BIOGRAPHIES

WAYNE J. DAVIS is a professor of General Engineering at the University of Illinois at Urbana-Champaign. His research areas include distributed planning and control architectures for large-scale, discrete-event systems, computer-integrated manufacturing and simulation.

XU CHEN received his MS degree in Mechanical and Industrial Engineering at the University of Illinois at Urbana-Champaign. He is currently employed at Dell Computer.

ANDREW BROOK is a graduate student in computer science at the University of Illinois at Urbana-Champaign.