Implementing On-Line Simulation With the World Wide Web

Wayne J. Davis, Xu Chen, Andrew Brook and Fayez Abu Awad

SIMULATION 1999 73: 40
DOI: 10.1177/003754979907300106

TECHNICAL ARTICLE

SIMULATION 73:1, 40-54

ISSN 0037-5497/99
Printed in the United States of America

Implementing On-Line Simulation With the World Wide Web

Wayne J. Davis, Xu Chen, Andrew Brook and Fayez Abu Awad
General Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA
E-mail: w-davis@uiuc.edu
URL: http://www-msl.ge.uiuc.edu/

This paper addresses the implementation of distributed, on-line simulation experiments with the World Wide Web. The paper first provides a basic overview of the on-line simulation approach. Next, we discuss the system that was considered in this study. The paper then describes the distributed object computing environment that has been employed in the study, including the specialized displays that can be accessed by a remote participant using the services of the Web. Finally, directions for future research are outlined.

Keywords: Web-based simulation, on-line distributed simulation

1. Introduction

Web-based simulation is a new and evolving research area. Fishwick's paper [1] was one of the first to discuss the topic. Since then, many more papers have been published. Today, there are three basic approaches to Web-based simulations. The first approach executes the simulation at the server, and the remote viewer is typically able to specify certain parameters governing the simulation and view the results of the experiment. The second approach is similar, except the simulation applet is loaded down to the remote site, and the simulation model executes upon the remote viewer's computer. The third approach allows the remote viewer to construct and execute his or her own simulation models using the services of the Web. Languages that support the latter approach include Simkit [2], JSIM [3], SimJAVA [4] and Silk [5]. One of the more recent directions is the implementation of distributed simulations upon the Web (see [6, 7]). The reader is referred to [8] to survey the entire area.

The research presented in this paper did not evolve explicitly as a Web-based simulation. Rather, the goal was to provide a means for demonstrating a new technology to other simulationists using the services of the Web. As this demonstration was constructed, however, elements of several of the above approaches were included. The constructed demonstration clearly illustrates that it is now possible to perform on-line projections of the near-term response for a system while it

operates under one or more alternative control strategies. In addition, we have now demonstrated that it is possible to implement such on-line simulation analyses in a manner that will permit the projected system response to be viewed concurrently by several participants at remote sites on the Web.

We begin by introducing our conceptual approach for conducting on-line simulation analyses. Next, we discuss the system considered in this experiment. Then, the distributed object-oriented computing environment that was constructed for implementing the experiment is detailed. In particular, the displays with which a remote viewer will interact upon the Web and the manner in which the simulating objects provide information to these viewing objects are detailed. The special situations that arose while implementing the experiment are next related. The paper closes by citing future research directions.

2. The On-Line Simulation Approach

On-line simulation is a new technology that few have had an opportunity to experience. In order to rectify this situation, we set out to demonstrate an on-line simulation with the Web. Although this appeared to be a difficult task at the outset, we soon discovered that constructing the demonstration was relatively trivial. The more interesting element of the experiment soon became one of determining how to configure the experiment in terms of the placement of the required distributed computing objects, rather than one of programming the objects themselves. Java and the Web provide a natural environment for performing on-line simulation. Using these capabilities, we were able to integrate the following: (1) one or more human users, (2) a real system, (3) several concurrent simulations of a real system, (3) methods for the on-line analysis of the simulation results, and (4) methods for controlling the system remotely. We will discuss the consequences of this integration later, but we believe that the integration would not be possible without Java and the Web.

Before discussing the development of the demonstration, we need to provide a brief overview of the on-line simulation approach. As shown in Figure 1, we begin by assuming that a real-world system exists and that a simulation model for that system has been created and verified. A critical requirement for the real-time operation of the real-world system is that a control policy has been selected for implementation. As shown in Figure 1, an on-line simulation is employed to project the future performance of the system as it continues to operate under the current control policy. On-line simulations are also performed for other control policies that could potentially be implemented if it is demonstrated that the projected performance for one of the alternatives exceeds that of the control policy that is currently being implemented.

In general, the selected control policy, in conjunction with the system's state transition function, determines the state evolution of the system as a function of time. However, this response is often not deterministic due to stochastic system elements and because the system must also respond to inputs outside its control. One

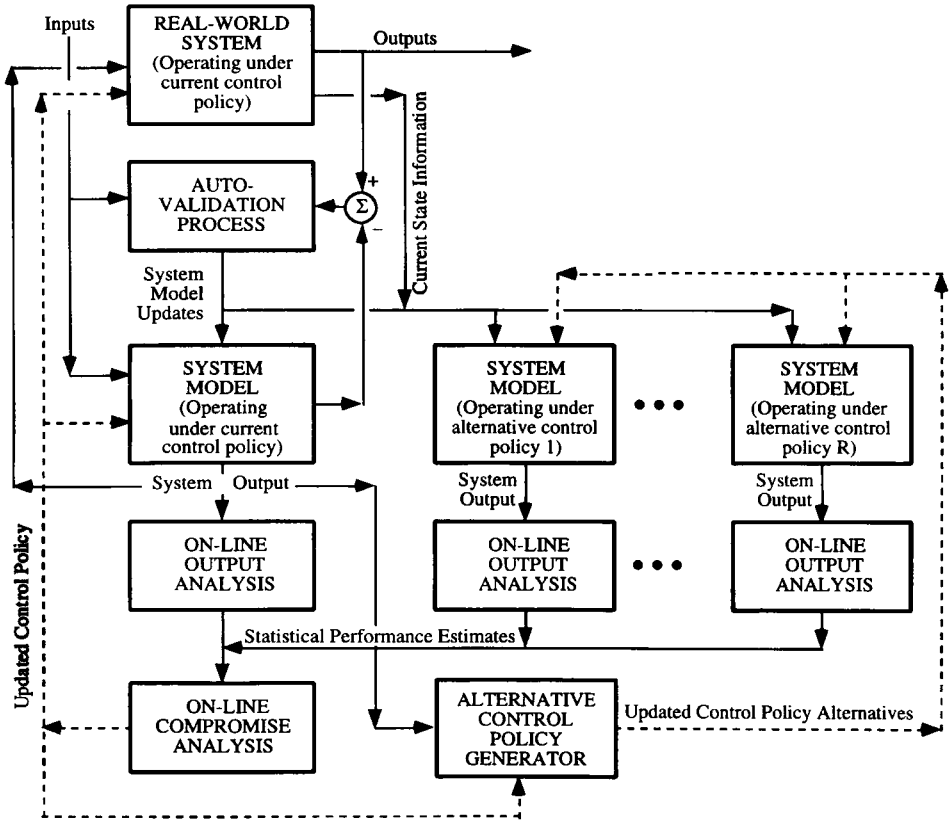

Figure 1. Schematic for the on-line planning/control process using real-time simulation

may view the system inputs as consisting of two components. The input to the system depicted in Figure 1 represents the *exogenous* control input over which the system has no control. The second component is the *endogenous* control input, which is dependent upon the selected control policy. A control policy can generate the control inputs in either of two ways. If an open-loop control policy is employed, then the control inputs generated under that policy are determined on an *a priori* basis and will not be explicitly dependent upon the evolution of the system state. For a closed-loop control policy, however, the issued control inputs are dependent upon both the selected control policy and the current state of the system.

The current system state provides another critical input for on-line simulations. The models that are employed in on-line simulations must be able to initialize their response to the current system state and then project the performance of the system as it continues to evolve from that state while responding to both the exogenous inputs from its environment and the endogenous inputs arising from the application of a given control law.

Although the system model has presumably been validated, the validation process must continue under the on-line planning/control scenario. Real-world systems are seldom stationary, that is, their operating characteristics constantly change with time. Typically, these changes are slow, but they can also be abrupt. For example, a new type of cutting tool with an extended tool life is introduced into a machining cell, and the number of tool replacements may be significantly reduced. There could also be improved processing techniques which significantly modify processing times.

Whether the system characteristics change slowly or abruptly, the system model must continually be updated to reflect these changes. To this end, Figure 1 includes an *auto-validation* process which compares the output projected by the model against the measured output from the system and updates the model to improve its accuracy. This auto-validation process functions in a manner similar to the system identification element which is contained in most conventional, self-organizing controllers for time-varying, continuous-state systems, (see [9, 10, 11]). Hence, the auto-validation process can be viewed as a system identifier for a discrete-event system. Unfortunately, robust procedures for implementing this auto-validation capability do not currently exist. However, there has been a recent push (particularly within the artificial intelligence community) to develop new learning technologies, such as explanation-based learning, that could address this concern. We also believe that it is possible to develop new modeling paradigms that will better explain the system behavior and simplify the validation process. One recent approach has been to employ the simulation model to physically control the modeled system (see [12, 13, 14, and 15]). In these approaches, the system model and the controller architecture are the same. We believe that auto-validation will become a major research concern in the near future as more attempts are made to employ on-line simulations.

Under an on-line planning/control scenario, it is essential that an on-line simulation of the implemented control policy be conducted concurrently with the implementation of the control policy by the real-world system. As stated above, the real-world system response is often stochastic. Because the real-world system operates under a selected control policy, it will realize only one of the potential state trajectories that could occur given the current system state. That is, the current operation of the system can be viewed as a single, statistical experiment that is dependent upon the current system state. Thus, the on-line simulation must necessarily be initialized to the current system state in order for it to be effective. Furthermore, unlike the real-world system that samples a single possible trajectory, the on-line simulation analysis must sample numerous potential trajectories in order to provide meaningful statistics. *In order to provide the essential number of simulation trials for the on-line analysis, the on-line simulation trials must actually be generated at a rate that is significantly faster than real time!*

Statistical estimates for the future performance of the real system, while operating under current control law, have at least four immediate uses. First, these estimates can be viewed as feed-forward information to be employed in the control of the system. Second, the projected response of the system operating under the selected control policy can assist in the generation of alternative control policies for possible implementation. Third, these projections provide a reference level of performance against which the predicted performance of other potential control policies can be compared. Finally, these projections clearly impact the auto-validation process as described above. For example, if the realized system performance is significantly outside the confidence interval for the projected response, it is clear that the system operator should be alerted that a significant discrepancy has occurred and the system model may no longer be valid. That is, control charts such as those employed in quality control might be employed in order to insure that the model is making accurate predictions.

Returning to Figure 1, the reader will note that R additional instantiations of the system model have been included. Each of these R instantiations considers an alternative control law for possible implementation. These simulations also receive the current state of the real system as an input and any updates to the system model derived from the auto-validation process. Again, these on-line simulations generate trials as quickly as possible in order to statistically characterize the future performance of the real system if it operates under the alternative control law.

In order to compute the statistical estimates, the outputs generated by the on-line simulation trials for each system model are passed immediately to an on-line output analysis process. Obviously, there is a correspondence between the on-line output analysis function included in Figure 1 and the statistical analyses performed in conventional off-line simulation studies. There are, however, marked differences in the statistical technologies required to perform on-line and off-line simulation analyses. Most off-line simulation analyses incorporate explicit measures to prevent the consideration of transient effects. On the other hand, on-line simulations focus solely upon the transient phenomena.

The actual statistical comparison of the future performance of a real-world system operating under current and alternative control laws is performed by the on-line compromise analysis function. It is assumed from the outset that multiple performance criteria will be considered and that a compromise solution must be defined. However, the procedures for performing the compromise analysis are not currently known. One approach might be to develop a utility function for real-time performance criteria in order to provide a single, aggregate performance criterion which can then be used in the comparison. There are, of course, limitations to this approach because the contribution of the individual criterion becomes obscured in the aggregation process. There are also other approaches to multi-criteria that have been considered in the operations research literature, (see [16] and [17]). However, these approaches have not presently been extended to consider the stochastic performance measures which are to be evaluated using simulation.

Comparison approaches are also a research issue. For the comparison of stochastic entities, the notion of stochastic dominance has been developed. However, the requirements that are necessary to assert stochastic dominance are so restrictive that true stochastic dominance is often difficult to demonstrate. Stochastic dominance requires that (empirical) cumulative distribution functions for the compared quantities must not intersect. That is, the cumulative distribution function for the performance criterion arising from the operation of the system under one alternative control strategy must be consistently greater than the cumulative distribution function for the same performance measure operating under another control strategy at all probability levels. Furthermore, in order to assert complete dominance in an on-line planning scenario, one control strategy must stochastically dominate all the others with respect to all the performance criteria. This situation seldom occurs, and other forms of statistical comparison are needed. Certainly more research is needed in performing on-line compromise analyses.

If an alternative control policy is demonstrated to provide an improved performance over the control policy that is currently being implemented, then the new control policy is transmitted to the real-world system for immediate implementation. It is also transmitted to the reference simulation model, which then begins to estimate the future performance of the system as it operates under the new control policy. The implemented control policy is also transmitted to the alternative control policy generator to provide a starting point for generating new control policy alternatives. These alternatives subsequently will be subjected to on-line simulation at the other R instantiations of the system model (included in Figure 1). Like many of the other elements discussed above, the procedures for generating these alternative control policies are a research concern. We believe that through the use of performance projections generated by on-line simulation, a whole new generation of planning algorithms will evolve. Such algorithms are already being explored in the development of learning systems within the artificial intelligence community.

Together, the on-line compromise analysis function and the alternative control policy generator implement a planning algorithm that is similar to the design optimization function for the conventional off-line planning scenario. In particular, the alternative control policy generator function can be viewed as one of implementing the search algorithm. There is, however, no direct correspondence to the conventional experimental design procedures that are often employed within off-line simulation-based planning scenarios. Given the discrete nature of the control policies to be evaluated, it is unlikely that an effective response function can be defined over a control policy space. That is, the definition of a response surface requires that a design variable space be defined such that the value of the performance criteria can be functionally related to the potential values of the design variables (factors). For the type of control policies considered during on-line planning analyses, this functional relationship is virtually impossible to define.

Based upon the above discussions, it is obvious that several fundamental research issues remain with respect to the implementation of the various functions in Figure 1. Unfortunately, this paper cannot explore these issues in detail. Rather, the reader is referred to [15], which provides a comprehensive discussion of the basic concepts and issues relating to on-line simulation.

However, the reading of articles pertaining to on-line simulations does not allow the reader to fully visualize the workings of an on-line simulation analysis. Given that on-line simulation analyses fundamentally differ from off-line simulation analyses in so many ways, the reader must see an on-line simulation in order to appreciate the contributions that it will provide for the development of futuristic planning and control technologies. Toward this end, we have constructed an on-line simulation demonstration upon the World Wide Web. The remainder of the paper will discuss the development of that demonstration.

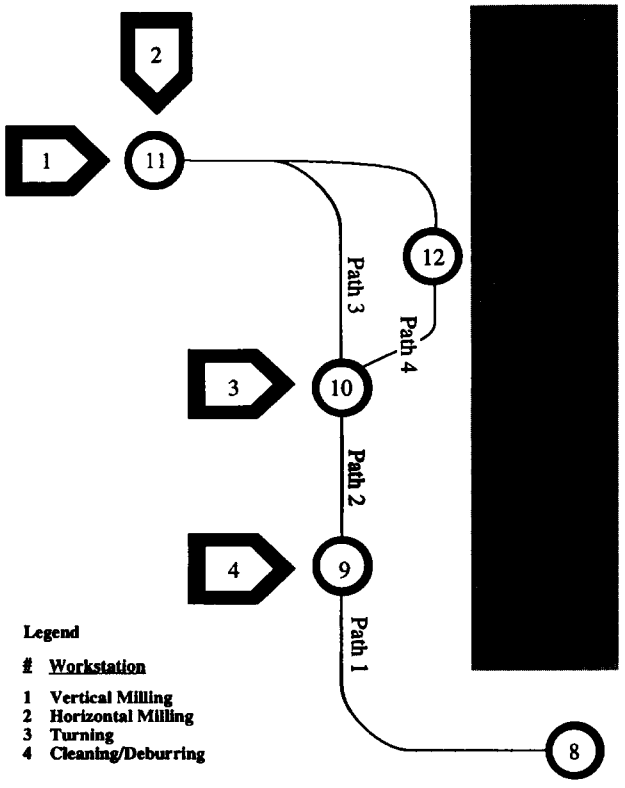

Figure 2. Schematic layout for the AMRF

3. The On-Line Simulation Experiment

The first step in constructing the demonstration was to select a system for the analysis. This experiment employs a model of the Automated Manufacturing Research Facility (AMRF) that was constructed by the National Institute of Standards in Gaithersburg, Maryland (see Figure 2). The AMRF provided 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 via an Automated Guided Vehicle (AGV) system. The AGV's path is illustrated in Figure 2 as the set of arcs connecting nodes 8 through 12. As shown, node 11 represents the entry/exit point to stations 1 and 2. Nodes 10 and 9 represent the entry/exit point for stations 3 and 4, respectively. Node 8 represents the entry/exit point for 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 primary loops to allow the AGVs to pass each other. Therefore, 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 3). 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 1, 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 AMRF simulation model can be found in [18]. The original simulation model for the AMRF that was discussed in the paper was reprogrammed in Java for this experiment in order to use Web-based computing capabilities.

In implementing our on-line simulation experiment, the model for the AMRF was first used to perform a real-time emulation of the system. In this application mode, the model is executed as a conventional 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

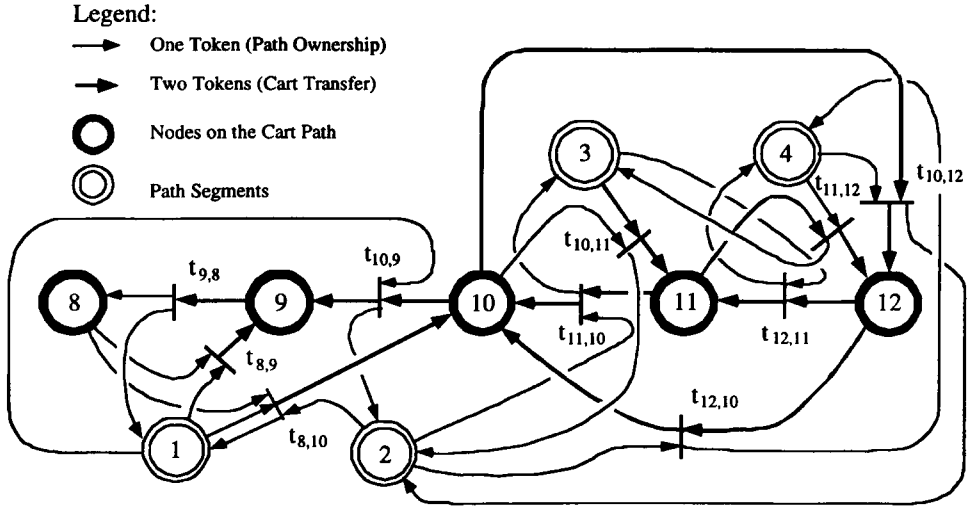

Figure 3. Petri-network control logic for allocating the path segments during AGV transitions

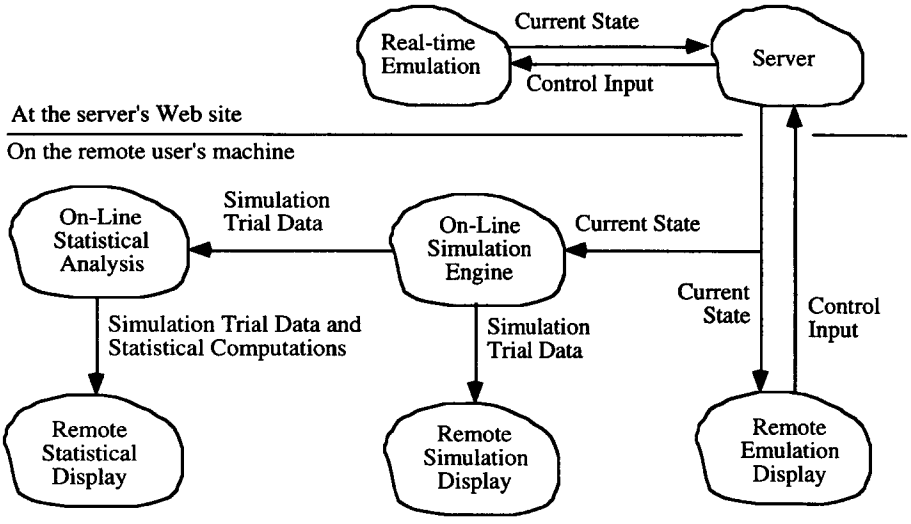

Figure 4. Organization of the distributed computing objects that comprise the on-line simulation experiment

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 the 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 simply to make the experiment more interesting.

The real-time emulation is actually an undesired element of the experiment. That is, we would prefer to replace the real-time emulation with a real-world system. As shown in Figure 4, 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 server becomes the central repository for the current state information of the emulated (or real-world) system. This server is also programmed in Java and must be activated before the Web demonstration can proceed.

Several remote clients can simultaneously log into the on-line simulation experiment. Whenever a remote client logs into the central server, a set of viewing applets is downloaded to the remote site as shown in Figure 4. Each of these applets performs a function within the on-line simulation environment as shown in Figure 1. In Figure 5, the applets depicted in Figure 4 are related to the on-line simulation functions given in Figure 1. Note that the human observer at the remote site executes the On-Line Compromise Analysis function for this experiment.

We will now discuss in detail each of the applets that are executed at the remote Web site in detail. The first of these applets produces the Timed Simulation Display, shown in Figure 6, which displays the detailed current state 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 passed 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 displayed, along with the job numbers of the first two jobs in the input queue and the output queue based upon the priority scheme that is being 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 that queue. 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 two AGVs. Included in this state information is the current node where the AGV resides, the destination node to which it is traveling, and the node that it will visit next 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 travels 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

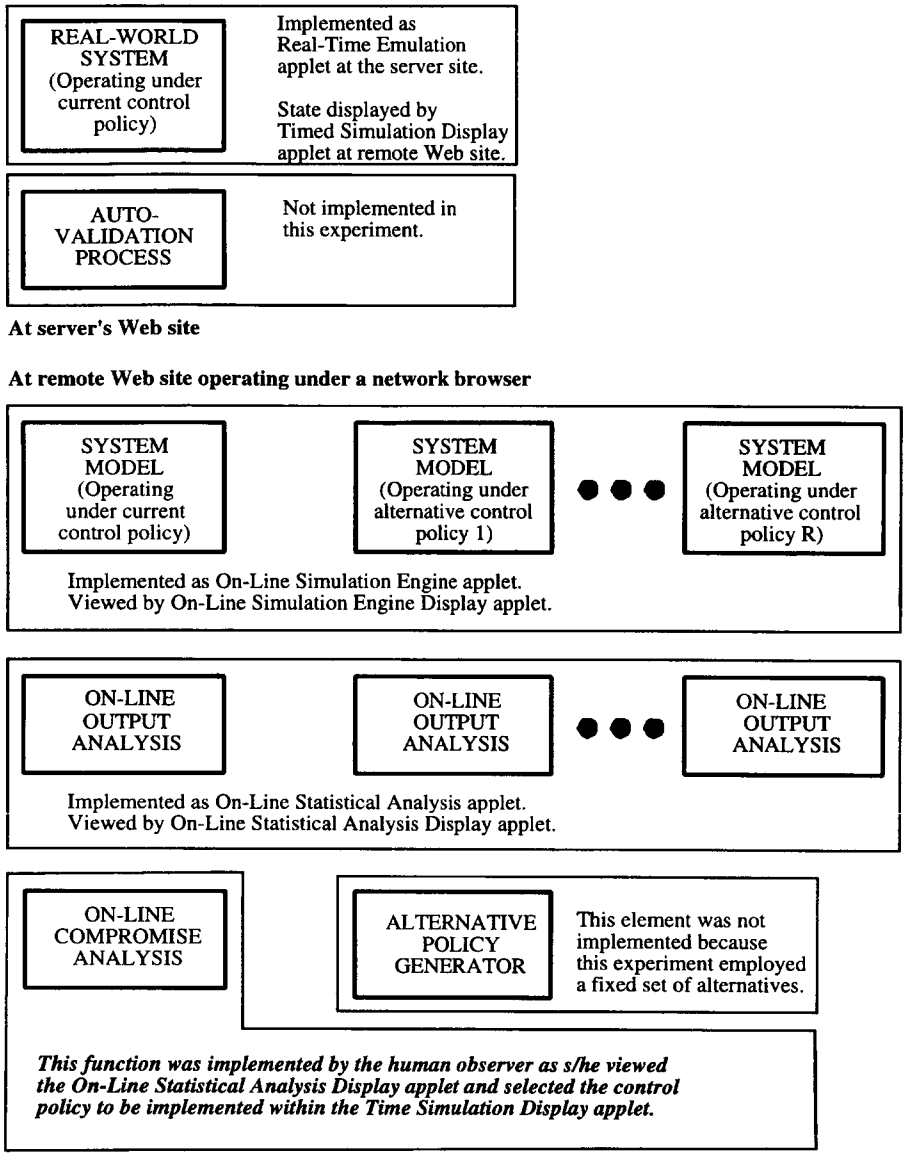

Figure 5. Graphic depicting relationship of the applets in Figure 4 to the on-line simulation functions in Figure 1

segment is currently owned by one AGV or is being requested by another AGV, then the AGV that owns or is requesting that path segment will be recorded in the appropriate field. To the right of these AGV 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 dedicated to each job to 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 emulated AMRF, a due date is assigned for its completion. Based upon the emulated completion time, the tardiness of each of the completed jobs 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 currently require 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 dates. 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 in Figure 6, the FIFO strategy has currently been selected, and its inefficiency has resulted in the congested system state depicted. 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 on-line simulation.

Returning to Figure 4, 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, the server broadcasts this information 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 viewer selects a different rule for implementation, the remote applet sends a message to the central server which then forwards the message to the 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 also changes with respect 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 4, 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 this 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,

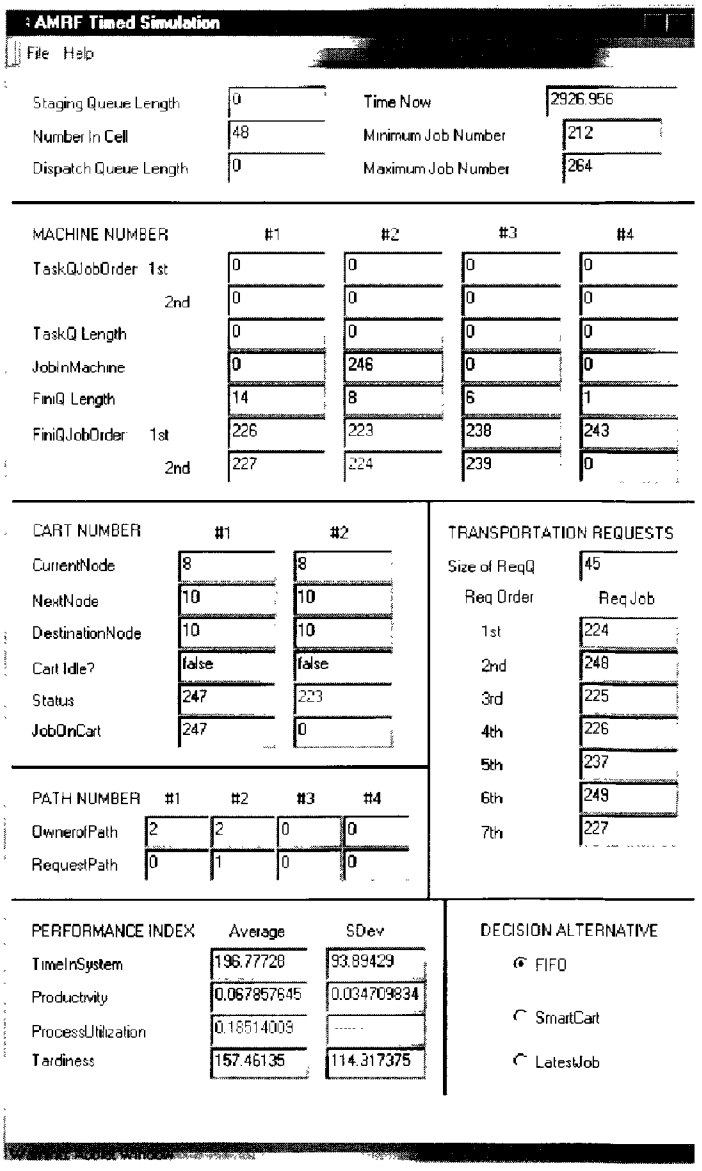

Figure 6. Timed simulation display for the state of the real-time emulation

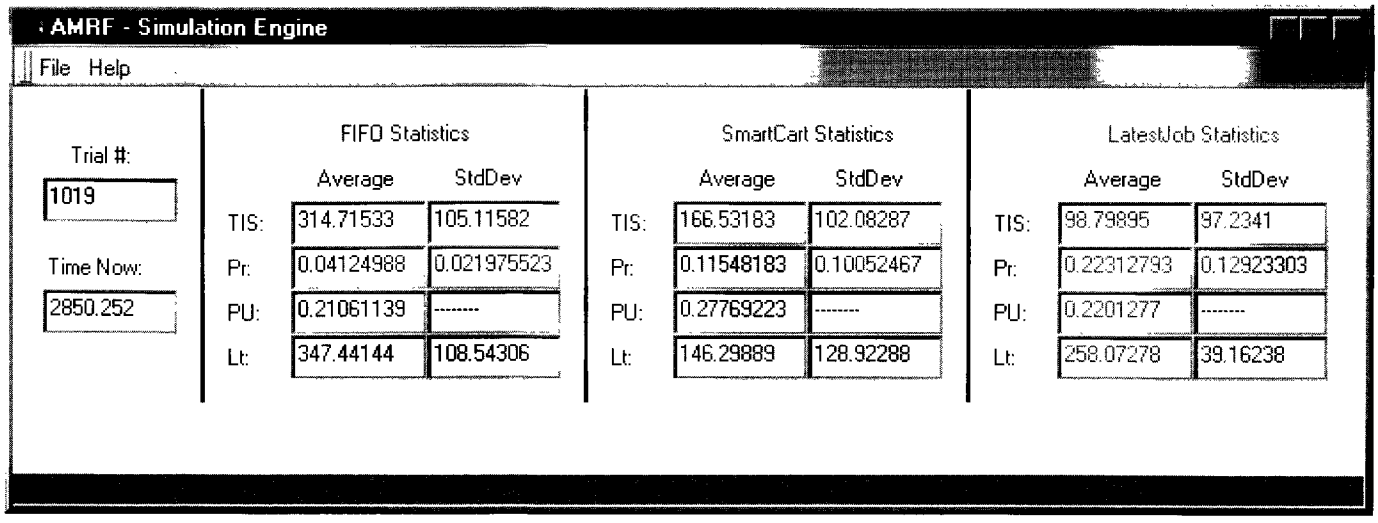

Figure 7. Graphic display for the simulation engine

we could have considered the performance of the system over the next 24 hours. In general, the analyst must specify the planning horizon that is to be considered in the on-line simulation 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 100 jobs considered on that 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, one trial for each priority rule, where each simulation trial is again initiated to the most recent system state. This process is repeated constantly throughout the on-line simulation experiment.

At the remote site, the applet contains a dedicated display for viewing the output of the On-Line Simulation Engine applet (see Figure 7). This display gives

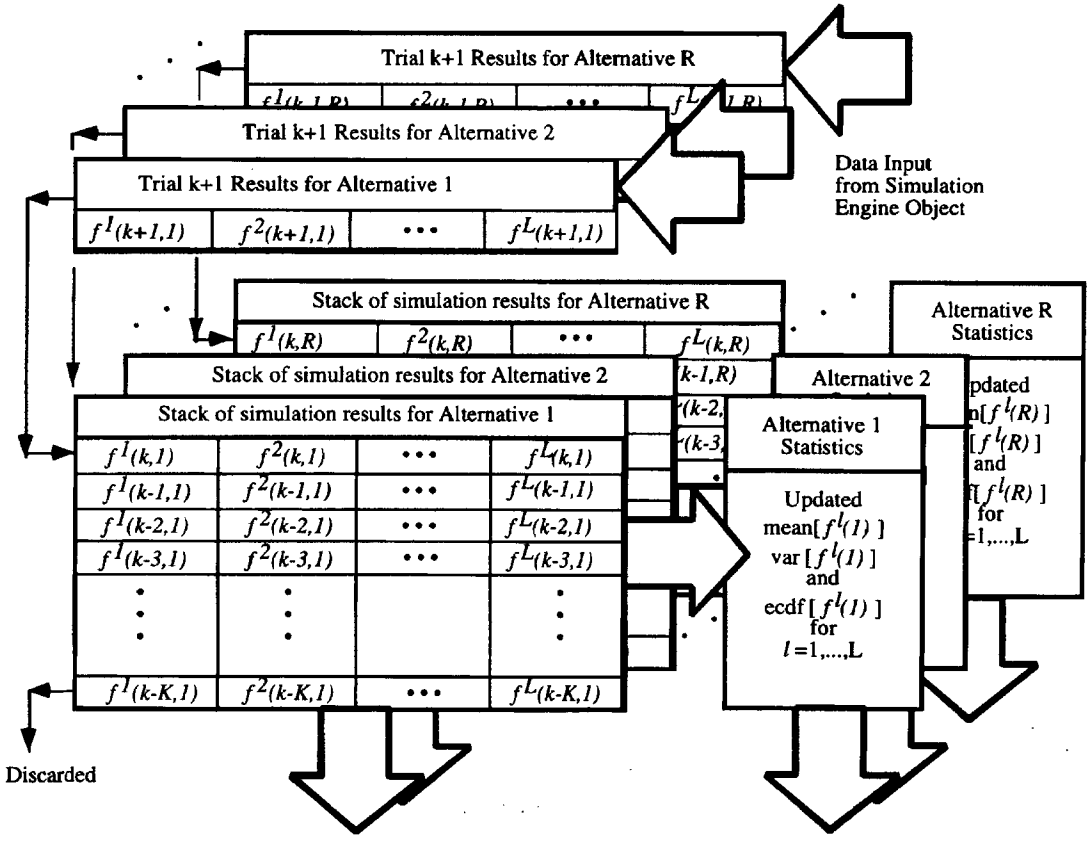

Figure 8. Schematic for the on-line statistical analysis process

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 trial generation process. That is, it provides information on the number of simulation trials that 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.

One contention that many have made with respect to on-line simulation analyses is that computers are not fast enough to perform the computations as needed. In order to address this concern, we now download the simulation model to each remote viewer's site and the on-line simulation trials are computed upon their computers using a Java-based simulation model. We note also that Java in general executes more slowly than most other computer languages because Java is compiled to a virtual machine rather than to the computer's native processor. In most cases, we find that the remote computer can generate another set of simulation trials within one or two seconds. In fact, returning to Figure 5, all of the processes in this diagram associated with on-line simulations of the current and the alternative control policies as well as the on-line statistical analysis are performed at the client's computer. That is, we are implementing a unique distributed computing environment at each remote viewer's site, each operating under the coordination of the Java-based server located in our laboratory. Note also that our server has been defined in a generic fashion that will allow us to place these computing processes anywhere we desire. We will discuss these Web-based implementation concerns in more detail in the next section.

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 generated by its

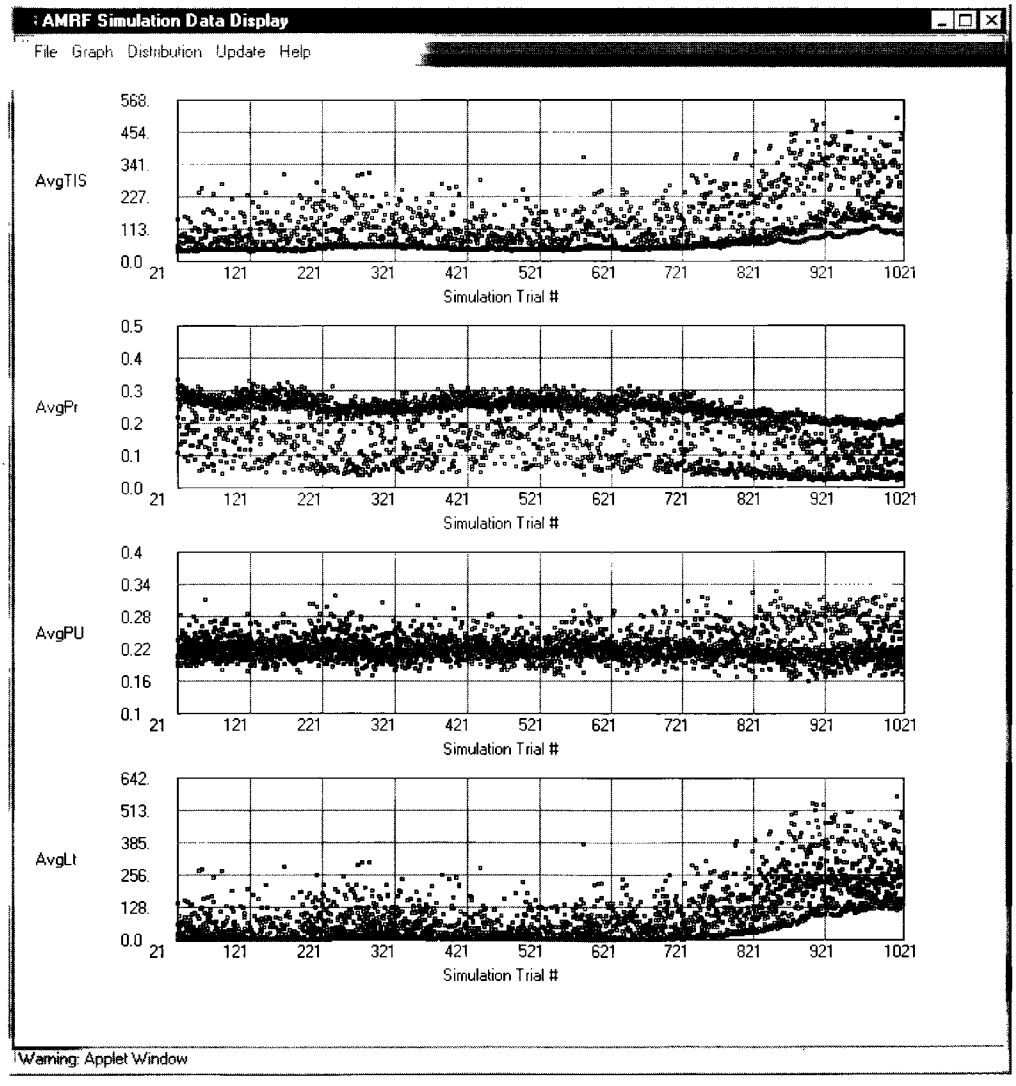

Figure 9. Graphic display depicting performance criteria values stored in the push-down stacks

local simulation engine. As shown in Figure 8, the projected performance vectors (which are comprised of the mean values for the L performance criteria considered in the study) resulting from each simulation trial for each of the R alternative priority rules are stored in separate push-down stacks, one for each priority rule. In this stack, a fixed number of performance vectors (K) 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 the stack and are disposed. Hence, after the stack is filled, it holds the K most recent simulation trials for a given rule. This situation is best illustrated in Figure 9, where the last 1,000 simulation trials (e.g. from trial 22 to 1,021) for each performance criterion are displayed.

Each remote applet also contains a Java-based, online statistical analysis object for computing the statistical results associated with the simulation trials contained in each stack. As shown in Figure 8, for each of the L 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 meantime in system for the next 100 jobs to be processed by the system.) The values recorded 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 8), 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 10, 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 1,000 simulation trials are plotted as a single point in the AveLt versus AveTIS Cartesian space. In this diagram, a total of 3,000 points are plotted: 1,000 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 observe this feature here as

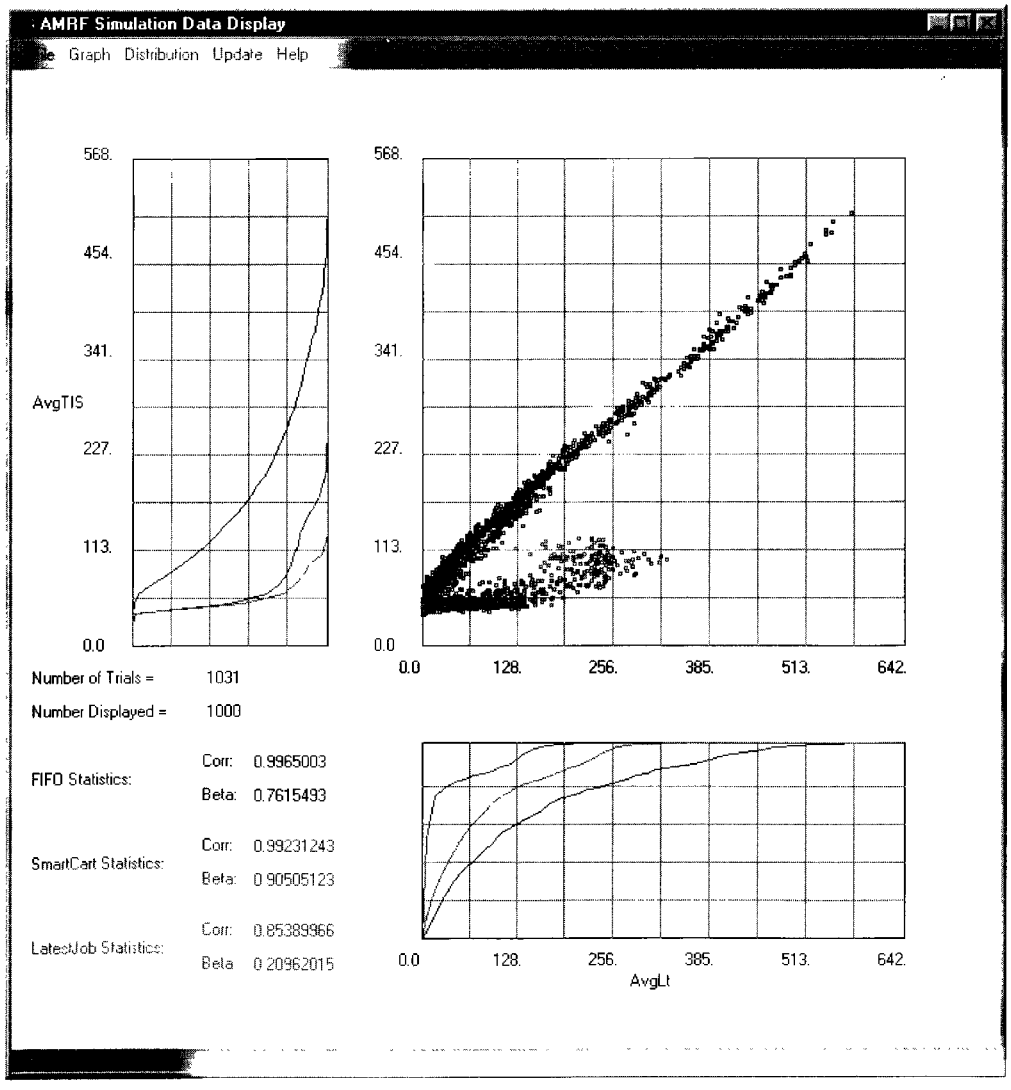

Figure 10. Simulation data display window depicting statistical distribution of lateness versus time in system

grayscale (not color) figures are presented. To the left of and below this primary scatter plot, the empirical cumulative distribution functions for the average values of the displayed performance criteria 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. These values are updated in real time 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. Special Issues in Implementing the Experiment

When we initiated this effort, our primary goal was to demonstrate the workings of an on-line simulation to a much larger audience. However, as we began to develop the demonstration, we quickly realized that Java and the Web are the ideal environment for doing this research. In short, we found numerous advantages and no insurmountable disadvantages. In fact, our entire research program in distributed intelligent control has now been moved to the Web.

In developing this Web-based demonstration, the only dedicated code that we developed explicitly for this experiment was the server. However, we were able to employ standard operating procedures that are published in many texts. The server can best be viewed as a message service. Returning to Figure 4, it is clear that the primary role of the server is to pass messages from one client applet to another. From the server's point of view, it has two primary types of clients. The first client type is the real-world system or the real-world emulation of that system. From this client, the server receives system state updates, and transmits incoming control from the remote on-line analysis clients.

The second type of client is the remote on-line client. When an individual logs into the server from a remote Website, his/her remote address is recorded. Note that we could easily add a security device here which prevents an unauthorized individual from logging onto the server. Once the remote on-line analysis client is logged onto the server, the server downloads all of the applets defined in the previous section. Each of these applets typically communicates with the server and not with each other. This is important because it allows the server to direct the essential information that is needed by each type of applet. For example, updates in system state go to Remote Emulation Display applet and the On-Line Simulation applet at each remote site. If one watches the demonstration carefully, one can observe that the updates for the TimeNow variable in these two applets can occur at slightly different times because the server is communicating with each applet as a separate entity.

We decided to treat the applets as distinct entities because the server will be receiving different types of information from each of these applets. Through the Remote Emulation Display, the viewer can send control signals to the server, which it then forwards to the real-world system (or its emulation). Note that we could install security measures at the server that could prevent a person who is not authorized to change the control law from requesting the change. In Figure 4, we have shown a simplified situation where the On-Line Simulation Engine applet communicates directly with the On-Line Statistical Analysis applet at the same Website. If several viewers are simultaneously viewing the on-line simulation demonstration, the potential arises for sharing the on-line simulation trials that are being generated at each site. That is, as a given On-Line Simulation Engine applet generates the performance vectors associated with a given set of simulation trials, it can forward them to the server, which then relays them to every On-Line Statistical Analysis applet that is currently active. In the current implementation of the experiment, the simulation engine does transmit the trial performance data to the server, which then bounces the information back to the On-Line Statistical Analysis applet at the same site. The ability to share the simulation data among all remote viewers also exists. However, we do not usually employ this feature in most demonstrations because it complicates the discussion of the experiment. Note, however, that the On-Line Simulation Engine applet does send the trial performance data that it generates directly to the Remote Simulation Display applet at its site. In this manner, the remote viewer can verify that its simulation is currently operating.

The On-Line Statistical Analysis applet receives simulation trial performance data from the server that was generated either by the On-Line Simulation applet at the same Website or by the entire collection of active On-Line Simulation applets. The output from the On-Line Statistical Analysis applet is then forwarded to the Remote Statistical Display at the same Website. (Thus, the On-Line Statistical Analysis applet does not send any data to the server.)

Based upon this discussion, it is clear that the server can be written in a general manner. Earlier in this section, we said that the server will interact with two basic types of clients. Rather than classify these clients as we had done earlier in this section, we can simply classify these clients as being permanent on-line clients (the real-world system) or as temporary clients (the remote sites performing the on-line simulation analysis). The server downloads the same set of applets to each temporary client each time they log in. Again,

this approach has its advantages because a single version of each applet is maintained at the server. In this manner, we can be sure that all viewers are using the same (most current) version of the applet.

Changes in the experimental setup are also easy to accomplish. One needs only to change the recipient list for the incoming messages from each type of client. For example, if we wanted the On-Line Statistical Analysis applets to use only the simulation trial data generated by the On-Line Simulation Engine at the same Website, then one would specify at the server that any incoming messages for a given On-Line Simulation Engine applet be forwarded to the On-Line Statistical Analysis applet at the same Website. However, if we desire to share simulation trial data, then the incoming messages from any On-Line Simulation applet would be forwarded to all active On-Line Statistical Analysis applets.

We have also developed the demonstration to maximize its reusability. If one desires to change the target system for the experiment, then one needs only to change the simulation model within the On-Line Simulation Engine applet (and the Real-Time Emulation applet if it is employed) and the Remote Emulation Display applet for viewing the state of the system. Again, these changes are made once at the server.

One might be concerned with the amount of information that is being passed across the network. Again, we have been very careful in our design. For example, the Remote Emulation Display applet receives only updates in the state rather than the entire state description. Every screen display is constructed locally. Although we have not found a need to do so, we can buffer the information at the server in order to pass it in larger packets, but at less frequent intervals. As a validation of the small amount of information that is being passed, we have exhibited this demonstration at several conferences using conventional phone lines and a modem.

To date, we have never experienced any large delays in information transmission. In fact, we have had viewers from three different continents viewing the demonstration at the same time. The concern here was to establish voice links among the concurrent viewers so that we could explain to the viewers what they were observing. We have had as many as ten viewers watching the demonstration at the same time. We believe that we could easily handle fifty or more simultaneous viewers. To test the robustness, we even launched three remote viewers from the same computer. The applets slowed down because they were sharing the same processor but continued to operate in the normal fashion.

Given the flexibility of the experimental design, we are now employing it in most of our experimental development related to on-line, distributed intelligent control. We have also begun implementing some of this technology upon real-world systems. Recently, we developed a collection of applets for a major electronics manufacturer where real-time state information was collected at several facilities and forwarded to a central server in our laboratory. This remote data could then be viewed at any remote site using the services of the Web. In fact, one of the sites continuously logged onto the Web in order to allow the line workers to monitor the current production data for their line. Given that we had the real-time data at our lab for several of the manufacturer's lines, it would have been a trivial next step to perform the on-line simulation needed to project the system's future performance. Unfortunately, the manufacturer has not agreed to take this next step.

As discussed above and in [15], we are now employing new modeling paradigms that will provide simulation models that are actually capable of controlling the systems that they model. We have began to employ this modeling paradigm in writing a new set of controllers for a flexible manufacturing cell operated at a military depot. Based upon our experience, we have elected to do this development in Java. Although there are several programmatic reasons for selecting Java for this development, one of primary reasons was the expanded capability for interacting with the clients during the development. The clients can log into the Website and monitor our progress. As we can develop interfaces, they can immediately review our efforts and suggest changes. By using the developed control software in a training mode, they can also interact with an emulated version of the system in order to verify that the resulting system dynamics are what they expected, before the controller is installed. Through this process, we expect that the client will be fully familiar with the controller before it is brought on-line.

Once we have developed the controller based upon our modeling paradigm, we can immediately perform on-line simulation with the same model. The next logical step is to develop the on-line intelligent controller that will assist in making on-line scheduling and resource allocation decisions for managing the systems. Here again, Java and the Web will be critical. On-line simulations generate enormous amounts of data with which the human operator must interact. As shown in Figure 5, the person performs the On-Line Compromise Analysis function within the intelligent controller (depicted in Figure 1) by viewing the displays from the On-Line Statistical Analysis applet and then decides which rule to implement by selecting that rule in the lower right-hand corner of the Timed Simulation display, shown in Figure 6.

For this experiment, only the rules for prioritizing the transportation request queue were considered. For a real system, there will be many more degrees of freedom that can be considered in generating alternative control policies for managing the system, given its current system state. It will be essential to have the user's input in the design of the displays for monitoring the on-line analyses of these alternatives. Java and the Web should assist in this process.

5. Future Research

In this paper, we have provided only a brief description of the on-line simulation methodology and an experimental implementation of this technology upon the World Wide Web. Obviously, the current experimental setup is still in the prototype phase. Future research should provide a standardized distributed object-computing environment for implementing these experiments. The goal of this experiment was not to provide a solution for all future on-line studies. Rather, the goal was simply to demonstrate the future capabilities of on-line simulations.

The area of on-line simulation is certainly in its infancy and needs much further investigation. In [15], future research needs are discussed in much greater detail. Clearly, considerable effort is needed in order to provide better statistical analysis tools for analyzing transient response. As in 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. The use of a more efficient priority scheme is likely to return the system to a more stable configuration. The issue is to develop on-line statistical analysis procedures that will make this situation apparent to the operator.

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 managing 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.

Based upon the above experiment, it is clear that Java and the Web will provide critical capabilities in this future development. Using the current capabilities, it should be possible to implement a fully operational on-line intelligent controller for a real manufacturing cell within the next five years. With this development, a manager should be able to view on-line operation of his/her system and make management decisions from any Website. Furthermore, when fully implemented, other managers should be able to view the same on-line operation of the system. The potential also exists where managers from several distinct manufacturing facilities within the corporation should be able to interact with each other in a real-time manner such that each manager sees the other systems' current state and make appropriate decisions to coordinate the production of his/her system with the other systems. The only real uncertainty here is what new Web technologies will be available in the next five years for us to exploit. Whatever these capabilities are, they will radically change the way we view simulation and control. Simulation will never be the same.

6. To Access This Simulation

The on-line simulation discussed in this article is accessible at the URL *http://www-msl.ge.uiuc.edu/*. However, the Java server must be enabled before one initiates the demo. Thus, if anyone is interested in seeing the demo, he or she should send an e-mail to Professor Wayne Davis at *w-davis@uiuc.edu* requesting that the Java server be started so that they can view the demo.

At this same URL, there is another demo where on-line simulation is being employed to generate a ship's course as it attempts to avoid incoming torpedos. This demostration clearly illustrates the expanded planning and control capabilities that on-line simulation can provide. This demo can be executed without contacting Professor Davis.

Based upon past experience, the demos seem to operate best with Internet Explorer 4.0 or later running on Win-based PC.

7. References

[1] Fishwick, P.A. "Web-Based Simulation: Some Personal Observations." *Proceedings of the 1996 Winter Simulation Conference*, J.M. Charnes, D.J. Morrice, D.T. Brunner, J.J. Swaim (eds.), pp 772-779, SCS, San Diego, 1996.

[2] Buss, A.H., Stork, K.A. "Discrete Event Simulation on the World Wide Web Using Java." *Proceedings of the 1996 Winter Simulation Conference*, J.M. Charnes, D.J. Morrice, D.T. Brunner and J.J. Swaim (eds.), pp 780-785, SCS, San Diego, 1996.

[3] Nair, R.S., Miller, J.A., Zhang, Z. "Java-Based Query Driven Simulation Environment." *Proceedings of the 1996 Winter Simulation Conference*, J.M. Charnes, D.J. Morrice, D.T. Brunner, J.J. Swaim (eds.), pp 786-793, SCS, San Diego, 1996.

[4] McNab, R., Howell, F.W. "Using Java for Discrete-Event Simulation." *Proceedings of Twelfth UK Computer and Telecommunications Performance Engineering Workshop*, pp 219-228, University of Edinburgh, UK, 1996.

[5] Healy, K.J., Kilgore, R.A. "Silk: A Java-Based Process Simulation Language." *Proceedings of the 1997 Winter Simulation Conference*, S. Andradottir, K.J. Healy, D.H. Withers, B.L. Nelson (eds.), pp 475-482, SCS, San Diego, 1997.

[6] Page, E.H., Moose, Jr., R.L., Griffin, S.P. "Web-Based Simulation in Simjava using Remote Method Invocation." *Proceedings of the 1997 Winter Simulation Conference*, S. Andradottir, K.J. Healy, D.H. Withers and B.L. Nelson (eds.), pp 468-474, SCS, San Diego, 1997.

[7] Fishwick, P.A. "Web-Based Simulation." *Proceedings of the 1997 Winter Simulation Conference*, S. Andradottir, K.J. Healy, D.H. Withers, B.L. Nelson (eds.), pp 100-102, SCS, San Diego, 1997.

[8] Fishwick P., Hill D., Smith R. (eds.). *Proceedings of the 1st International Conference on Web-based Modeling and Simulation*, SCS, San Diego, 1998.

[9] Astrom, K.J., Wittenmark, B. *Computer-Controlled Systems: Theory and Design*, Prentice-Hall, Upper Saddle River, NJ, 1997.

[10] Eykhoff, P. *System Identification: Parameter and State Identification*, John Wiley & Sons, New York, 1974.

[11] Ljunj, L. *System Identification: Theory for the User*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[12] Smith, J.S., Wysk, R.A., Sturrock, D.T., Ramaswamy, S.E., Smith, G.D., Joshi, S.B. "Discrete Event Simulation for Shop Floor Control." *Proceedings of the 1994 Winter Simulation Conference*, J.D. Tew, S. Manivannan, D.A. Sadowski,A.F. Seila (eds.), pp 962-969, 1994.

[13] Drake, G.R., Smith, J.S., Peters, B.R. "Simulation as A Planning and Scheduling Tool for Flexible Manufacturing." *Proceedings of the 1995 Winter Simulation Conference*, C. Alexopoulos, K. Kang, W.R. Lilegdon, D. Goldsman (eds.), pp 805-812, SCS, San Diego, 1995.

[14] Peters, B.A., Smith, J.S., Curry, J., LaJimodiere, C. "Advanced Tutorial—Simulation Based Scheduling and Control" *Proceedings of the 1996 Winter Simulation Conference*, J.M. Charnes, D.J. Morrice, D.T. Brunner, J.J. Swaim (eds.), pp 194-198, SCS, San Diego, 1996.

[15] Davis, W.J. "On-Line Simulation: Need and Evolving Research Requirements." In *Handbook of Simulation*, J. Banks (ed.), pp 465-516, John Wiley & Sons, New York, 1998.

[16] Fishburn, P.C. *Nonlinear Preference and Utility Theory*, Johns Hopkins Press, Baltimore, 1987.

[17] Wolfram, S. "Fundamentals of Multicriteria Optimization." In *Multicriteria Optimization in Engineering and in the Sciences*, S. Wolfram (ed.), Plenum Press, New York, 1988.

[18] Tirpak, T.M., Deligiannis, S.J., Davis, W.J. "Real-Time Scheduling in Flexible Manufacturing." *Manufacturing Review*, Vol. 5, No. 3, pp 193-212, 1992.

7. Additional Reading

Kreutzer, W., Hopkins, J., Mierlo, M.V. "SimJAVA—A Framework for Modeling Queueing Networks in Java." *Proceedings of the 1997 Winter Simulation Conference*, S. Andradottir, K.J. Healy, D.H. Withers and B.L. Nelson (eds.), pp 483-488, SCS, San Diego, 1997.

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.

Fayez Abu Awad is a Graduate Student in Industrial Engineering at the University of Illinois at Urbana-Champaign.