A METHODOLOGY FOR DEVELOPING A WEB-BASED FACTORY SIMULATOR FOR MANUFACTURING EDUCATION

Maged M. Dessouky[1]
Department of Industrial and Systems Engineering
University of Southern California
Los Angeles, CA 90089
Phone: 213-740-4891 Fax: 213-740-1120 email:maged@rcf.usc.edu

Sushil Verma
MarketWiz, Inc.
4830 Williams Road
San Jose, CA 95129
sverma@market-wiz.com

Diane E. Bailey
Department of Management Science & Engineering
Stanford University
Stanford, CA 94305-4026
Phone: 650-723-3821 Fax: 650-723-2826 email: Diane.Bailey@stanford.edu

Jeff Rickel
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292-6695
Phone: (310) 822-1511 Fax: (310) 822-0751 email: rickel@isi.edu

March 20, 2000

[1]**corresponding author**

A METHODOLOGY FOR DEVELOPING A WEB-BASED FACTORY SIMULATOR FOR MANUFACTURING EDUCATION

ABSTRACT

Historically, manufacturing engineering education has focused on teaching mathematical models using simplifying assumptions that can mask the realities of complex manufacturing systems. Recent pedagogical approaches to manufacturing education have focused on developing a more holistic view of the manufacturing enterprise. In this paper, we describe the contents and development methodology of a Virtual Factory Teaching System (VFTS) whose aim is to provide a workspace that illustrates the concepts of factory management and design for complex manufacturing systems. The VFTS is unique in its integration of four domains: web-based simulations, engineering education, the Internet, and virtual factories. Evolutionary development of the VFTS is accomplished by separating the simulation model from the graphical interface and user interaction.

1. Introduction

Modern factories are becoming increasingly agile. The components of a manufacturing system (e.g., production, purchasing, design, and management) are integrated today to facilitate rapid and frequent changes in products and processes. To succeed in this dynamic environment, a new engineering college graduate must develop a holistic view of the total business process from design to production to delivery [38]. Hands-on teaching methods are ideal for conveying the complexity of the modern factory, but traditional pedagogy in manufacturing is ill-equipped for the task: on-campus factories face prohibitive cost and space considerations, and student experimentation during industrial site visits is infeasible.

To address the manufacturing educational needs of new engineers, we developed a collaborative learning network called the Virtual Factory Teaching System (VFTS) that is accessible via the Internet's World Wide Web. The VFTS is unique in its integration of four domains: web-based simulations, engineering education, the Internet, and virtual factories. At the core of the VFTS lies a Visual SLAM [34] simulation factory model. The VFTS allows students, working alone or in teams, to build factories, forecast demand for products, plan production, establish release rules for new work into the factory, and set scheduling rules for workstations. An animated panel displays jobs progressing through their factory simulation, complete with queue counts, finished goods counts, graphs, and reporting functions [10].

In the next section, we briefly describe some of the related work in each of the four domains united by the VFTS (i.e., web-based simulations, engineering education, the Internet, and virtual factories) that has helped guide our pedagogical viewpoint and, as a result, our development decisions. Then we present the contents and development methodology of the VFTS, as well as an example of the use of the VFTS. We conclude with our future development plans, in particular the addition of pedagogical agents. Readers interested in exploring the VFTS

can access it through our web page at http://vfts.usc.edu/ by registering as a guest with the password "guest."

2. Related work in the four domains

2.1. *Web-based Simulation*

Recent directions in simulation include developing web-based simulators [16] . Buss and Stork [7] list two benefits of web-based simulators: (1) posting of the simulation model on a web-server that can be executed by any user with a Java-enabled browser and (2) accessibility of the model anywhere in the world. Allen [1] argues that web-based simulators are ideally suited to support virtual environments, interactive simulations, and multiuser interaction. In some instances, they can be cost effective since the simulation can be run on a remote location, while visualizing the results on a low cost local machine [37].

Page [32] developed a home page on the Internet surveying web-based simulators. Java is the predominant language used for web-based simulators because of its portability, reusability, object-orientation, and graphical capabilities [31]. Examples include Simkit [7], Simjava [27], and the CPU-disk simulator [17]. These systems do not support concurrent multiuser interaction.

An alternative approach is to completely separate the simulation model from the graphical interface and user interactions [30]. Examples include the CMOS integrated circuit simulator [6], OASISNET – an OASIS network simulator [41], the network-centric simulation object system [9], and interactive simulation for airbase logistics systems [31]. The VFTS uses this separation approach where the graphical and user interface is developed using Java while the simulation model is developed using Visual SLAM.

2.2. *Engineering Education*

Numerous studies indicate the potential for computer-based learning tools to aid in the classroom instruction of students ([13] and [18]). Beyond developing particular tools, a few

recent efforts have been aimed at developing more integrated approaches to manufacturing education. For example, Sheater, Martin, and Harris [40] describe a manufacturing management program that employs 'hands-on' training in workshop technologies, a simulated factory environment for teaching CIM and FMS technologies, self-directed computer-based learning techniques, and matrix teaching structures for linking streamed theoretical subjects with practical case study material. Other research points to the benefits of collaborative work. Rada *et al*. [35] found that students working in collaborative groups have been better able to formulate concrete ideas and avoid misconceptions. These results are encouraging, as learning to work as part of a team, perhaps one spanning the globe, is expected to be an important skill for new engineers and managers [8].

In recent years, there also has been a surge in interest in applying multimedia technologies to the instruction of manufacturing processes, system design, and production management. The approach of Jackson, Muckstadt, and Jenner [22] is distinguished by its use of manufacturing games and individual factory modules. Tufekci *et al*. [43] have set up a laboratory in which they model a factory via a network of computers where each computer adopts an object found on the factory floor. Such systems show the possibility for modeling factories in the educational context.

2.3. *Internet*

Research investigating the educational potential of linked computer systems, and in particular of networks such as the Internet, has truly exploded in the past few years. There are literally hundreds of examples of current networked educational tools. We mention a few of the projects: a web-based instructional technology course for education masters' students [24], an on-line computer environment for computer conferencing, e-mail, tutoring, and assignments in a management education program [24], the use of Internet media such as Gopher, Ftp, and the

World Wide Web in an undergraduate course in computer science [15], the use of the web for training computational scientists in parallel processing [14], and automated processing and just-in-time delivery of course content using a modified web client-server architecture [23]. We cannot hope to review all such projects here, and opt instead to mention a few main points.

Much of the interest in network-enabled education is aimed at distance education [33]. Bailey and Cotlar [4] note that Internet use in education stimulates deeper learning and improves teacher-student interaction and student-student interaction. There are obstacles, however, to network education. Tomlinson and Henderson [42] highlight a number of issues concerning the value, viability, and development of distributed computer-supported collaborative learning software. One of their main findings is that software development is limited by existing hardware platforms. Technological considerations must include system support for interoperability in a heterogeneous distributed environment [5].

2.4. *Virtual Factories*

Virtual plants are increasingly used within manufacturing industries as representations of physical plants (e.g., TMA's CAESAR system for representation of integrated circuit factories [3]). Virtual manufacturing environments can assist a company in its efforts to rapidly and effectively react to changes in market conditions and technology [28]. The major benefit of a virtual factory is that physical system components (such as equipment and materials) as well as conceptual system components (e.g., process plans and equipment schedules) can be easily represented through the creation of virtual plant entities that emulate their structure and function. These entities can be added to or removed from the virtual plant as necessary with minimal impact on other system data. The software entities of the virtual plant have a high correspondence with real system components, thereby lending validity to simulations carried out in the virtual system meant to aid decision-makers in the real one [12].

3. Current VFTS

Based on our review of previous work, we determined to build a collaborative learning network that would permit students to create and experiment with their own factories. Our design enables students to participate in the functioning of a virtual factory by assuming the roles of various factory personnel in small team settings. This concept of virtual worlds and role-playing is common in the entertainment industry [21]. Our intent is to apply this concept to manufacturing education. Through acting out these roles, students can witness the range of decisions an engineer or a manager makes and their effect on the performance of a company. Student teams may even span institutional boundaries, thereby facilitating virtual teaming and allowing information-sharing among students and professors at a diverse group of universities across the nation and around the world. The system allows many different students to access the same factory simulation at different locations, with a built-in timing capability that allows students to view the same simulation animation at different or the same speeds.

Our constructivist approach to education is evident in how a factory is created. Students enter factory parameters in special input fields in one window, then a graphical representation resulting from their choices is immediately displayed in a neighboring window. For example, if the team opts to set up a factory with four workstations in series, a layout diagram representing the workstations rapidly appears. Immediate, visual interpretation of data input serves to confirm that the students and the machine share the same image of the factory world.

The VFTS lends itself to a mosaic of development possibilities. Engineering education is imparted through a variety of techniques, depending upon the specific curriculum. Pedagogical tools used, degree of student participation, and material presentation are only some of the many variables involved. A virtual factory can be represented via a simple simulation model, or it can be made more complex with dynamic representation of the human decision-making process. The

communication network can range from a local area network to the global Internet, with a university-wide intranet or a city-wide education network serving as intermediate possibilities.

To effectively handle such a range and diversity of possibilities, we adopted an evolutionary design and implementation philosophy for the VFTS. The balance of this section presents the design and implementation of the current VFTS. The software components of the VFTS are integrated within a layered software architecture such that they can be modified and upgraded without affecting the rest of the system. Figure 1 shows the different software components that together define the VFTS. The system consists of three different layers: the real factory world, the virtual factory world, and the student world.

3.1. *Real Factory World*

The real factory world interfaces with the virtual factory world through models created by students as guided by the professor in classroom lectures and discussions. Site visits to physical factories help students form the linkages between the real and virtual worlds. The connection between the real and the virtual factory is currently off-line. That is, the instructor and students visit a factory to develop an understanding of the actual factory settings. Factory data that form the basis of the virtual factory model are provided to the instructor and students. The virtual factories are then created within the VFTS by either the instructor or students. It is possible that in later phases a live connection can be made on the Internet with the real factory. We envision that the shop floor MES (Manufacturing Execution System) can provide a snapshot of the real factory at a fixed point of time. This snapshot forms the boundary condition for further simulations to be performed in the VFTS.

3.2. *Virtual Factory World*

The virtual factory is a representation of a real or imagined factory; it is virtual in that its components (e.g., machines, jobs, even operators) do not exist as physical entities, but only as

objects in computer memory. Embedded in a virtual factory is typically a simulation model. The VFTS uses the Visual SLAM simulation package [34] as its core. We next describe the other components of the virtual factory world in the VFTS.

Student Workspace	Provides private storage space for students and faculty. Only the authorized owner of the space is able to use it.
WWW Server	Enables communication to and from the Internet.
Document Manager	Manages VFTS documents. The document manager is a collection of Common Gateway Interface scripts, Java applets, Perl scripts, etc.
Virtual Factory Capability Documents	Guide the user in establishment of suitable simulation parameters. In their simplest form, they serve as a repository of simulation features and programs, with description of limitations and assumptions involved in each program that is used by the ensuing VFTS simulation.
Simulation Configuration Agents	Allow the user to customize the simulations by linking necessary data files, programs, user permissions, and location for outputs. They constitute a collection of programs and editors.
Simulation Engine	It forms the computational core of the teaching system. It interfaces with the simulation documents, configuration programs and input/output mechanisms to ingest a coherent factory model and simulate it according to the instructions provided.
Simulation Documents	Act as input/output to the simulation engine.
Virtual Factory Documents	Consist of actual data files, video clips, etc. used by the system.
Data Interface Agents	Contain the code needed to read/write or communicate with the external world and different versions of the simulation programs.

The virtual factory model within the VFTS currently contains three standalone modules: scheduling, planning, and forecasting. Currently, these modules are not integrated; each module has its own factory model representation. (In the concluding section, we discuss our plans for integrating these modules.) The *scheduling* module is the only module that currently operates in a simulation mode. That is, a student interacts with the module via a Visual SLAM simulation model running in the background. The other two modules are not evaluated in a simulation environment. We later discuss our plans for integrating these two modules in the simulation environment.

The term "factory model" encompasses a number of parameters. In principle, the model contains all the information needed to simulate a real factory. The factory layout is naturally the first level of information. Most factories fall into the category of job shops or flow shops. In modern flexible environments, however, hybrid shops are increasingly common. The VFTS currently represents a hybrid flow shop. In a traditional flow shop, all part types visit the manufacturing work centers in the same sequence. In a hybrid flow shop, each manufacturing work center may have multiple identical machines. This type of configuration was selected because it is common in many industries (e.g., electronics, garment, and chemical) and because it provides a natural environment in which an instructor can show factory behavior as a function of factory control rules. The student creates a factory by entering the number of work centers and the number of machines at each work center. The student enters job data, which includes information such as processing times, due dates, priorities, and demand size. The student also enters whether the input parameters of the factory will be either deterministic or stochastic.

Two factory control policies are part of the scheduling module: dispatching and release. *Dispatching* rules select which part(s) to produce next at a machine when the machine becomes idle. For each work center, the student selects a dispatching rule from a list that includes shortest

processing time, least slack, earliest due date, highest priority, and first-come-first-serve. The performance of each dispatching rule depends on the performance criteria. For example, the shortest processing time rule tends to minimize work-in-process (WIP) while the least slack rule tends to minimize the lateness of the production jobs [29].

Release rules determine when to release a new lot of raw material to the factory floor. Release rules can be based on monitoring the status of the bottleneck machine [44] or on global factory status [11]. The bottleneck machine monitoring rules tend to be based on pull-type systems such as Kanban while the global factory status rules tend to be based on push systems such as MRP-based systems [20]. The student selects either a push release or one of two pull systems, zero inventory or workload regulating. The zero inventory rule is a strict pull system. Workload regulating, developed by Wein [44], monitors the total workload measured in machine processing time units that is upstream from the bottleneck. If the workload is a below a threshold, new material is released to the factory. Push systems tend to control throughput and measure work-in-process (WIP) inventory while pull systems tend to control WIP and measure throughput [20]. Hence, there is a trade-off between the release rules that factory schedulers must consider, with no single rule dominating in all manufacturing situations. Furthermore, the dispatching and release rules are interrelated depending on many factors including the factory layout, processing times, demand pattern, and machine reliability.

After the factory model is developed, the student can test the different control rules by running a simulation. A Visual SLAM simulation model is automatically generated from the input fields specified by the student. During the execution of the simulation, an animated panel displays jobs progressing through their factory, with queue counts, finished goods counts, graphs, and reporting functions all available.

In addition to the scheduling module, the forecasting and planning modules are part of the VFTS. Several forecasting techniques (e.g., exponential smoothing, regression, and moving average) are available to forecast constant, trend, or seasonal demand data. Students test various input parameters or techniques to find the best forecasts for a given set of demand data. The planning module allows students to determine the appropriate resource (work force and machine) utilization, production volumes, and inventory levels to meet given demand levels for various company objectives. This module allows students to use either a linear program-based planner or a demand matching planner. The linear program model is automatically generated in LINDO format [39] from the input fields specified by the student.

3.3. *Student World*

The student world interfaces with the system using browsers that are capable of running web client processes with dynamic content. The interface contains Java applets/scripts, plug-ins for any display of video, and the ability to invoke external applications. As a result, students are able to freely use the VFTS simulations and external applications like editors and talk systems. They can use the built-in chat room to communicate with other team members at different sites. Figure 2 shows the process of building and using a model in the VFTS from the students' perspective. As the figure shows, the students' interaction with the VFTS depends on whether they are building a factory model or running a factory model. The former interaction is typically during the design time while the latter interaction is during the run time.

3.3.1. *Design time*

During the students' design time, the users create a factory model to be used either for forecasting, planning, or scheduling purposes. The student enters a different factory window for each of the different modules. For the forecasting module, the student inputs the parameters relevant to the selected forecasting technique. For the planning module, the student interacts

with the built-in planning window and does not have to be proficient in the LINDO software package. The same is true for the scheduling module, where the function of building the Visual SLAM simulation model is transparent to the student. In this manner, the student can focus on learning the production concepts instead of the software packages.

3.3.2. *Run time*

All the modules present the outputs in text and graphical format with plots. Of the three modules within the VFTS, only the scheduling module is currently a dynamic model. During the run time, a student is able to execute simulations, change decision parameters, and interact with other team members in conversations mediated by the system over a network.

At run time, a complete factory model is fed into the simulation engine with the simulation parameters and a description of any decision-making roles that are to be enacted by the students. To facilitate role-playing among team members, the VFTS has a hierarchical-level security system. That is, certain data fields and windows are only accessible through a given password. For example, the process engineer on the team may have access to only equipment related data while the scheduler may have access to only the dispatching and factory release rules fields. In some cases, the team members may be given conflicting goals. Expanding this area of role-playing is part of our future direction and is discussed in more detail in the concluding section.

During the simulation run, an animation of the factory is shown to the student. The animation shows the impact of the different scheduling policies on factory performance. The student has options to slow, speed-up, or pause the animation. In addition to the animation, the student can interactively access numerous reports and plots on factory performance regarding inventory levels and job status. They then can study the reports, perhaps deciding to change certain parameters in the factory model.

A special feature of the animation capability within the VFTS is the catch-up feature. Note that the simulation engine is located in a remote location on the server. Hence, students on the same team may interact with the system on client machines at different locations, causing possible synchronization problems in viewing the animation at the different sites. The speed of data transmission of the Internet and the non-uniform computing profiles of the computers involved can cause two different problems. The client may not see the simulation at the server speed, and the speed may vary with time. Also, different clients may see the same simulation appearing at different times. We provide two features to control these problems. First, we permit the incoming simulation data to be buffered at the client side without getting displayed. This allows the client animator program to read the buffer content at a fixed speed. Second, the clients can adjust the granularity of the client-side simulation tick according to the speed of the computer so that they can collaborate in real-time with other remote clients on different computers. If they fall behind, they can also do a quick catch-up.

The timing problem may be due to many reasons including: network access speeds are different, students run the animation at different speeds, and a student at one site pauses the animation. The catch-up capability allows the animation at the slower location to catch up with the faster location. In this manner, the students at the different locations can be at the same point in time during the simulation run.

3.4. *Communication Architecture*

All communication with the server is designed to occur using Java applets. The user first downloads the applet, the applet runs and communicates with the server and spawns other applets. All client-server communication over the TCP/IP is managed by a Java sockets package. A socket is a channel that enables communication with a network address over a particular port. We use both datagram and stream sockets in our implementation. Because of the

nature of the application, a number of threads of communication are created between each client and the server. Each client maintains a separate thread for overall management of the session. A simulation management thread is created to control the simulation's progress over the server. This thread includes the ability to pause the simulation, modify the simulation parameters and restart the simulation. Another thread is a data intensive live channel that carries the dynamic simulation content. This thread drives the concurrent simulation graphics. This live channel is designed according to the buffering technology, similar to the ones used in audio and video live streaming data. A special client is enabled to permit remote system management. The client allows the teachers to manage the student groups, give selective permissions to students based upon their roles, add new members, etc.

The server mainly consists of two segments: the simulation program and the client manager. The two pieces themselves communicate using Windows Socket technology. Multiple simulation programs can exist concurrently on the server to allow multiple student groups. The client manager is a Java based application consisting of three parts: session manager, applet manager and a simulation data manager, each corresponding to distinct functionality on the client side. The architecture allows the simulation server to be separated from the client manager and be put on a dedicated server. At any given time, different simulation engines could run on various different computers and connect via the client manager to a network of students over the Internet.

4. Example of use of the VFTS

We aim to show the many different ways the VFTS can prove beneficial, including providing opportunities for students to learn teamwork, role-play, solve complex problems, and work remotely. Each of these skills is learnable in isolated contexts using traditional methods. For example, standard simulation packages can be used to help students grasp complex

problems, and teamwork can be built via group assignments. The VFTS provides an environment in which students can learn all these skills in an integrated, and hence more real-life, fashion. An example may help to demonstrate how the VFTS can aid students in developing conceptual understanding to complement their analytical thought.

In our example, six workstations exist in series, each with a different number of parallel identical machines operating within it. Table 1 displays data for the initial setup. (Obviously, students can create factories even larger and more complex than this one, which is after all one of the main goals and benefits of the VFTS. For example, different types of machines in each workstation and different processing times for different types of jobs can be specified. Our example is meant for illustration purposes.) One student plays the role of the factory manager, who is in charge of the factory layout and overall specifications. Other students (up to six) supervise the six workstations divided amongst them. The goal of the student group is to work as a team to achieve a targeted throughput that minimizes the overall number of machines per workstation while keeping other metrics under control. Moreover, the members of the factory team are not all physically located in the same place. One supervisor telecommutes; another is traveling on business and logging in from a hotel. Furthermore, the last workstation and its supervisor are located in an offshore factory. (Products are not shipped to this remote site, but digital images of a feature are transferred there electronically for testing purposes.) All members login into the same simulation website to work together across distances.

Table 1. Information for factory set-up

Workstation	1	2	3	4	5	6
Number of Machines	3	4	6	2	4	5
Processing Time on Machine	1	5	2	2	7	2

The factory floor manager first creates this initial factory in the VFTS. Figure 3 shows the first two VFTS windows. The student manager enters the module type in the first window. In our example, the student enters the scheduling module. The second window allows the student to create a new factory or to edit (or start running) a pre-existing factory listed under "Registered factories." It also lists any factories that are currently running on the server in the "Factories in Session" box. The VFTS allows students to join a currently running factory. These two windows encapsulate the Virtual Factory Capability Documents. In an expanded version, more capabilities may be added, with illustrative documents explaining each capability.

Figure 4 shows sample input windows for defining the factory listed in Table 1. In the definition window, the student manager enters the factory name, number of work centers, and whether the factory is deterministic or stochastic. If stochastic is selected, then the inter-arrival times, processing times, time between machine failures, and machine repair times are assumed to be exponential, normal, exponential, and exponential random variables, respectively. In the Factory window, the student manager enters the characteristics of each work center regarding number of machines, time between machine failures, and machine repair time. When a machine fails, the job currently at the machine returns to the beginning of the input queue, and the first idle machine at the work center services the remaining processing time of the job. The student manager enters the characteristics of each job type regarding priority and routing in the Jobs window. Here, the variability of the processing time is entered as a percentage of the mean. Next, the arrival data is entered in the Orders Window. The "number of orders" field sets the maximum number that will be released to the factory; the "mean volume" field specifies the lot size. The due date is set to be the sum of the processing times for the job multiplied by the due date factor. In this example, the due date factor is 1.5. Last, the student manager enters the release rule of the factory and the dispatching rule of each work center. The simulation

parameters collectively are an example of the Simulation Documents from the input perspective. The programs that control the simulation models are examples of the Simulation Configuration Agents. The actual screens are generated using programs that belong to the class of Data Interface Agents.

The methodology adopted for solving this problem is to start with a large number of machines in each workstation that easily provide the targeted throughput. Starting from this base, the supervisors together identify the machines with very low utilization that can be removed. All the while, the WIP needs to stay within specified bounds. In the process of removing machines, students learn the effect of dispatching rules, release rules, and push versus pull systems.

The supervisors also can run the factory to answer a series of questions for which they are first told to think what the answer might be based on the understanding they have developed thus far. They can confirm their ideas by running a simulation on the VFTS. For example, for this problem, students can begin by assuming no breakdowns and no transportation time. Starting with a push system with FIFO scheduling, they can determine the rate of each workstation, the bottleneck, the throughput rate, and the makespan. Then they can establish two levels of job priorities, think about which aspects of the system dynamics will change, and rerun the simulation. They can switch from a push system to a pull system, again thinking about expected changes before running the system. Similarly, they can increase the transportation value. Here, students might correctly guess ahead of time that the makespan will increase and the throughput decrease, but they may fail to realize that the WIP will decrease, too. The simulation reports can help them to determine why this change in WIP occurs. They can introduce stochastic machine failure to a single machine first (a non-bottleneck machine versus a bottleneck one, for example),

and then to every machine. They can use the system to verify proven manufacturing dynamics, such as the minimization of mean flow time via the shortest processing rule.

Figure 5 shows a sample animation window for this example. The animation is automatically displayed when the Start Factory button is clicked. (See Figure 3.) Different colors are employed to indicate different machine status (e.g., green is busy, blue is idle). The boxes with numbers are the queues. The first queue is controlled by the factory release strategy; the last box represents the finished goods inventory. During the execution of the simulation, the animation displays jobs progressing through the factory, with queue counts, finished goods counts, graphs, and reporting functions all available. The black dots in the animation window represent jobs. As the animation progresses, the dots travel through the factory from one workstation to the next. The animation shown here is an example of a Virtual Factory Document that can be stored and redisplayed at a future time.

It is important to realize that there is only one simulation, which occurs on the server. All remote parties see a reflection of this simulation in their web browsers. There are two types of simulation ticks displayed in the animation window. The first tick shows the current time of the animation display; the second tick shows the current time of the simulation at the server. In the example, the delay is 20 time units. The simulation data contained in the time difference interval are stored in local buffer. A student could cause this delay by either pausing the animation or setting a slow speed for it. The catch-up capability allows the time of the animation display to match the simulation clock time at the server. This capability provides the students with the flexibility to watch the simulation at their own respective speeds, and then converge to the same time point when desired.

Figure 6 displays some of the graphical capability of the VFTS. There are three types of graphs in the VFTS containing output for the factory, work center, and job, respectively. Plots of

factory output might include the number of jobs completed, WIP, and cycle time. Plots of work center output might include queue sizes and waiting times. Plots for job might include values for tardiness and earliness. For each output, the student can display and graph the current value, average, and standard deviation. This plots are another example of Simulation Documents, in this case from the output perspective. They are prepared in the background by the Data Interface Agents.

The student supervisors ultimately devise a plan that seems to achieve good throughput while keeping WIP low. They notify the factory manager using email. The student manager connects online. After reviewing their solution, the manager explains in the collaboration window why the proposed solution does not go far enough. The manager, prompted perhaps by the professor, sets a target number of machines for the team and challenges them to come up with a plan to achieve it. The manager also may be instructed by the professor to set private and conflicting goals for the supervisors in addition to restricting the number of machines overall. Initially, the students may find the challenge too great. Perhaps after reading more about flow shops or hearing a class lecture, they come back with new ideas. As they pursue them, they begin to develop an intuitive understanding of how release rules and dispatching strategies affect throughput. They also learn how to play their roles to achieve team-oriented results.

5. Future development plans

We next outline our development methodology for expansion of the VFTS including the incorporation of pedagogical agents.

5.1. *Engineering Education*

As more functionality is added to the VFTS, student roles can be expanded, allowing students to tackle more complicated, realistic problems. The problems themselves can be garnered from real factories, with site visits helping students form the linkages between the real

and virtual worlds. In the more complex problems, neither the final solution nor the means of achieving it will be unique. Moreover, multiple objectives may be operative.

Because the production planning and control course is a common one found in many universities, collaboration among faculty to arrange "virtual" teams will be feasible. Student teams can even cross university boundaries, thereby facilitating virtual teaming and allowing information-sharing among students and professors at a diverse group of universities across the nation and around the world. In this manner, students can learn how to organize and complete work in the absence of face-to-face communication. Faculty members may even team teach the course. Two virtual factories created at two different universities could be linked as two economic units to study their codependency. Eventually, a network of virtual factories could be developed by linking universities through communication networks. This concept of using the VFTS for virtual teams will be tested in the fall of 2001 at the University of Virginia, Tennessee State University, San Jose State University, and the University of Southern California.

5.2. *Virtual Factory*

The current VFTS limits the model to that of a hybrid flow shop. Moreover, the various modules (i.e., forecasting, planning, and scheduling) are not integrated. We plan to steadily expand the domain to general factory layouts, as well as to a variety of business processes, such that we will model an entire enterprise in its essential elements. The basis of our expansion will be a hierarchical framework consisting of three levels: strategic planning, management control, and operations control [2]. Outcomes at the higher level will be treated as inputs to the next lower level. Strategic planning decisions, made at the highest level of the organization, concern issues such as what products should be marketed and where factories should be located. Decisions made at the management control level include resource utilization (work force and machines), inventory levels, production volumes, and demand satisfaction. Sample output at this

level is what, when, and how to produce. Operations control decisions are made at the factory floor level. Typical decisions made at this level include detailed scheduling of individual customer orders on each resource, as well as expediting and processing of orders.

As previously stated, the scheduling module is the only simulation portion of the system. We plan to integrate the higher level decisions in the Visual SLAM model so that students can make higher-level decisions in a dynamic mode. Many such decisions can be broken down into concrete parameters for a simulation model. For example, a management decision to give highest priority to the jobs of strategic customers can be trickled down in many ways. The VFTS system can be used to study the overall effect of such a policy on the performance of the system. It also can be used to find the best way to implement the policy so as to minimize its adverse impact. Financial decisions about when and where to add capacity in the factory similarly can be broken down into models that can be simulated in an enhanced version of the VFTS. In terms of the example presented in Section 4, the forecasting and planning modules, when integrated into the simulation model, can be used to derive due dates for the jobs or to increase capacity.

5.3. *Internet*

Currently, students access the VFTS server through a standard WWW browser. The server consists of a Java-based manager and a Visual SLAM based factory simulator. Note that the current version uses Java Version 1.0, sockets, and Abstract Windowing Toolkit (AWT). To make use of all the Java-based utilities, a new interface to the VFTS will be developed using Java Version 1.1, remote method invocation (RMI), and Swing. We plan to evolve a distributed simulation design for the enterprise model spanning a number of computers, networks, and universities via a collection of peer servers, one dedicated to each division of the enterprise. We envision that each module will be part of a single but distributed object oriented simulation model whose parts will communicate via messages sent over the Internet. The current VFTS has

an architecture that allows expansion along these lines. Moreover, it will be used to calculate the bandwidth capacity requirement of the proposed design before it is implemented, which should shorten the design cycle.

5.4. *Pedagogical Agents*

Pedagogical agents are useful in interactive simulations to guide student activity in productive directions. For example, when students do not know how to complete a given task and lack a clear understanding of it, they may perform that task incorrectly [19]. Agents may help to remedy this problem, and so we plan to incorporate pedagogical agents into the VFTS.

A distinct thread of agent research has focused on the development of autonomous agent technologies for synthetic environments. Although in many instances autonomous agents are controlled using simple mechanisms such as state transition networks or potential fields, there has been increasing interest in the use of artificial intelligence (AI) architectures for controlling agents in synthetic worlds. AI techniques are appropriate when the agents must perform complex behaviors in a wide variety of situations. For example, agent architectures such as Soar [25] have been used to create agents that can pursue goals yet respond dynamically to changing environments, as is necessary for agents that are expected to operate effectively in interactive virtual environments. These technologies come together in the form of animated pedagogical agents. For example, Rickel and Johnson's [36] pedagogical agent named Steve trains operators of complex equipment in virtual environments.

The main task will be to define the types of interventions we expect of such agents. The primary role for agents will be to recognize student learning opportunities and ensure that students recognize them as well. Agents will be designed to recognize particular combinations of factory management decisions and their expected simulation results. When such situations

arise, they provide a context in which the agents can discuss important principles with students, thus grounding theoretical concepts in concrete applications.

6. Conclusions

Pedagogical tools for transferring hands-on knowledge to engineering students have traditionally required physical laboratories. However, factory experimentation through full-scale on-campus laboratories is an infeasible alternative for engineering programs due to the high expense associated with development and maintenance. A virtual factory for educational purposes is freed from many of the problems that inhibit the creation of physical laboratories. With equipment sets represented by abstractions, entire manufacturing processes and technologies can be represented without creating full-scale factory environments. This paper has presented the details of a Virtual Factory Teaching System, VFTS, to teach modern manufacturing problems, practices, theory, and techniques to engineering students.

In summary, the VFTS provides a mediating medium for student teams. Multiuser access and operation are handled by cloning the common programs. The system allows sharing of read-only sections and permits editing of various specified parameters.

Acknowledgements

The research reported in this paper was partially supported by the National Science Foundation under grant CDA - 9616373. We thank Dr. Sadashiv Adiga of Control Factory Systems and Professors George Bekey and Edward Kazlauskas from the University of Southern California for their contribution and input to the project. We thank Ajay Singh, Mohammad Reza Kolahdouzan, Parthiv Patel, H. O. Reed, and Zachary Baker for their help in developing the VFTS. We also would like to thank the members of the advisory board consisting of Mr. Jack Ferrell, Dr. Patricia Heffernan-Cabrera, Dr. Elizabeth Morris, and Dr. A. Alan B. Pritsker for providing input in the content of the VFTS.

References

[1] Allen, R. (1998) The web: interactive and multimedia education. *Computer Networks and ISDN Systems*, **30** (18), 1717-1727.

[2] Anthony, R. N. (1965) *Planning and Control Systems: A Framework for Analysis*. Harvard University Press, Cambridge, MA.

[3] Axelrod, V., Granik, Y., and Jewell, R. (1995) Caesar - the virtual IC factory as an integrated TCAD user environment. *Microelectronics Journal*, **26** (2-3), 191-202.

[4] Bailey, E.K. and Cotlar, M. (1994) Teaching via the Internet. *Communication Education*, **43** (2), 184-193.

[5] Blair, G.S., Coulson, G., and Davies, N. (1994) System support for multimedia applications: An assessment of the state of the art. *Information and Software Technology*, **36** (4), 203-212.

[6] Bogliolo, A., Benini, L., and Demicheli, G., and Ricco, B. (1997). Gate-level power and current simulation of CMOS integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI)*, 5 (4), 473-488.

[7] Buss, A. H., and Stork, K. A. (1996) Discrete-event simulation on the World Wide Web using Java. *Proceedings of the 1996 Winter Simulation Conference*, edited by J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, Winter Simulation Conference Board of Directors, San Diego, CA, 780-785.

[8] Chidambaram, L. and Jones, B. (1993) Impact of communication medium and computer support on group: perceptions and performance - A comparison of face-to-face and dispersed meetings. *MIS Quarterly*, **17** (4), 465-491.

[9] Colvin, K. and Beaumariage, T. (1998) Network-centric simulation using NCSOS. *Simulation*, **70** (6), 396-409.

[10] Dessouky, M. M., Bailey, D. E., Verma, S., Adiga, S., Bekey, G. E., and Kazlauskas, E. J. (1998) A virtual factory teaching system in support of manufacturing education. *Journal of Engineering Education*, **87** (4), 459-467.

[11] Dessouky, M. M., and Leachman, R. C. (1994) An optimization based modeling methodology for operational level scheduling. *Production and Operations Management*, **3** (4), 276-295.

[12] Dessouky, Y. M., Roberts, C. A., and Beaumariage, T. G. (1995) Object-oriented simulation architecture with real-time capabilities. *International Journal of Production Research*, **33** (9), 2471-2492.

[13] Duplessis, J.P., Vanbiljon, J.A., Tolmie, C.J. and Wollinger, T. (1995) A model for intelligent computer-aided education systems. *Computers & Education*, **24** (2), 89-106.

[14] Dwyer, D., Barbieri, K., and Doerr, H.M. (1995) Creating a virtual classroom for interactive education on the Web. *Computer Networks and ISDN Systems*, **27** (6), 897-904.

[15] Fay, D.Q.M. (1994) Internet and the electronic classroom. *Microprocessing and Microprogramming*, **40** (10-12), 847-850.

[16] Fishwick, P. A. (1996) Web-based simulation: some personal observations. *Proceedings of the 1996 Winter Simulation Conference*, edited by J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, Winter Simulation Conference Board of Directors, San Diego, CA, 772-779.

[17] Fishwick, P. A., Belk, M., and Spatz, B. (1997) An interactive web simulation of CPU/Disk performance. At: http://www.cise.ufl.edu/~fishwick/CPUDisk/, University of Florida, Gainsville, FL.

[18] Haag, M., Maylein, L., Leven, F. J., Tonshoff, B., and Haux, R. (1998) Web-based training: a new paradigm in computer-assisted instruction in medicine. *International Journal of Medical Informatics*, **53** (1), 79-90.

[19] Hill, R.W. and Johnson, W.L. (1995) Situated plan attribution. *Journal of Artificial Intelligence in Education,* **6** (1), 35-67.

[20] Hopp, W. J. and Spearman, M. L. (1996) *Factory Physics: Foundations of Manufacturing Management*. Irwin Publishing, Homewood, IL.

[21] Hughes, C. E., and Moshell, J. M. (1997) Shared virtual worlds for education: the ExploreNet experiment. *Multimedia Systems*, **5** (2), 145-154.

[22] Jackson, P. L. , Muckstadt, J., and Jenner, M. J. (1994) Course Materials for Manufacturing System Design, at http://www.orie.cornell.edu/~jackson/aseehtml.html, Cornell University, Ithaca, NY.

[23] Johnson, W.L., Blake, T., and Shaw, E. (1996) Automated management and delivery of distance courseware. *WebNet '96 – World Conference of the Web Society Proceedings,* 225-230.

[24] Kazlauskas, E. (1997) Personal communication, materials available at http://www.usc.edu/dept/itp/lacoe.html, University of Southern California, Los Angeles, CA.

[25] Laird, J.E., Newell, A., and Rosenbloom, P.S. (1987) Soar: An architecture for general intelligence. *Artificial Intelligence*, **33** (1), 1-64.

[26] McConnell, D. (1994) Managing open learning in computer-supported collaborative learning environments. *Studies in Higher Education*, **19** (3), 341-358.

[27] McNab, R., and Howell, F. A. (1996). Using Java for discrete-event simulation. *Proceedings of the Twelfth UK Computer and Telecommunication Performance Engineering Workshop*, University of Edinburgh, 219-228.

[28] McGehee, J., Hebley, J., and Mahaffey, J. (1994) The MMST computer-integrated manufacturing system framework. *IEEE Transactions on Semiconductor Manufacturing*, **7** (2), 107-116.

[29] Morton, T. E. and Pentico, D. W. (1993) *Heuristic Scheduling Systems*. John Wiley & Sons, New York.

[30] Narayanan, S., Schneider, N. L., Patel, C., Carrico, T. M., and DiPasquale, J. (1997) An object-oriented architecture for developing interactive simulations using Java. *Simulation*, **69** (3), 153-171.

[31] Narayanan, S., Malu, P. G., Ashili, K. P. B., Di Pasquale, J., and Carrico, T. M. (1998) A web-based interactive simulation for architecture for airbase logistics system. *International Journal of Industrial Engineering*, **5** (4), 324-335.

[32] Page, E. (1998). A survey of web based simulations. At http://ms.ie.org/websim/survey/survey.html.

[33] Peraya, D. (1997) Quoted from http://tecfa.unige.ch/edu-ws94/contrib/peraya.fm.html.

[34] Pritsker, A. A. B, and O'Reilly, J. J. (1999) *Simulation with Visual SLAM and AweSim,* 2nd Edition, John Wiley & Sons, New York, New York.

[35] Rada, R., Acquah, S., Baker, B. and Ramsey, P. (1993) Collaborative learning and the MUCH system. *Computers & Education*, **20** (3), 225-233.

[36] Rickel, J., and Johnson, W. L. (1999) Animated agents for procedural training in virtual reality: perception, cognition, and motor control. *Applied Artificial Intelligence*, forthcoming.

[37] Rodrigues, M. A. F., Gillies, D. F., and Charters, P. (1998) Modeling and simulation of the tongue during laryngoscopy . *Computer Networks and ISDN Systems*, **30** (20-21), 2037-2045.

[38] Sackett, P. and McCluney, D. (1992) Inter enterprise CIM - A mechanism for graduate-education. *Robotics and Computer-Integrated Manufacturing*, **9** (1), 9-13.

[39] Schrage, L. (1991) *LINDO: An Optimization Modeling System,* Fourth Edition, Scientific Press, San Francisco, CA.

[40] Sheater, G., Martin, R., and Harris, D. (1993) Partners in excellence: A new model for cooperative education. *Educational & Training Technology International*, **30** (1), 5-31.

[41] Tian, Y., and Gross, G. (1998) OASISNET – An OASIS network simulator. *IEEE Transactions on power systems*, **13** (4), 1251-1258.

[42] Tomlinson, H. and Henderson, W. (1995) Computer-supported collaborative learning in schools - a distributed approach. *British Journal of Educational Technology*, **26** (2), 131-140.

[43] Tufekci, S. (1999) Quoted from http://gurgun.ise.ufl.edu, University of Florida, Gainsville, FL.

[44] Wein, L. M. (1988) Scheduling semiconductor in wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing,* **1** (3), 115-130.

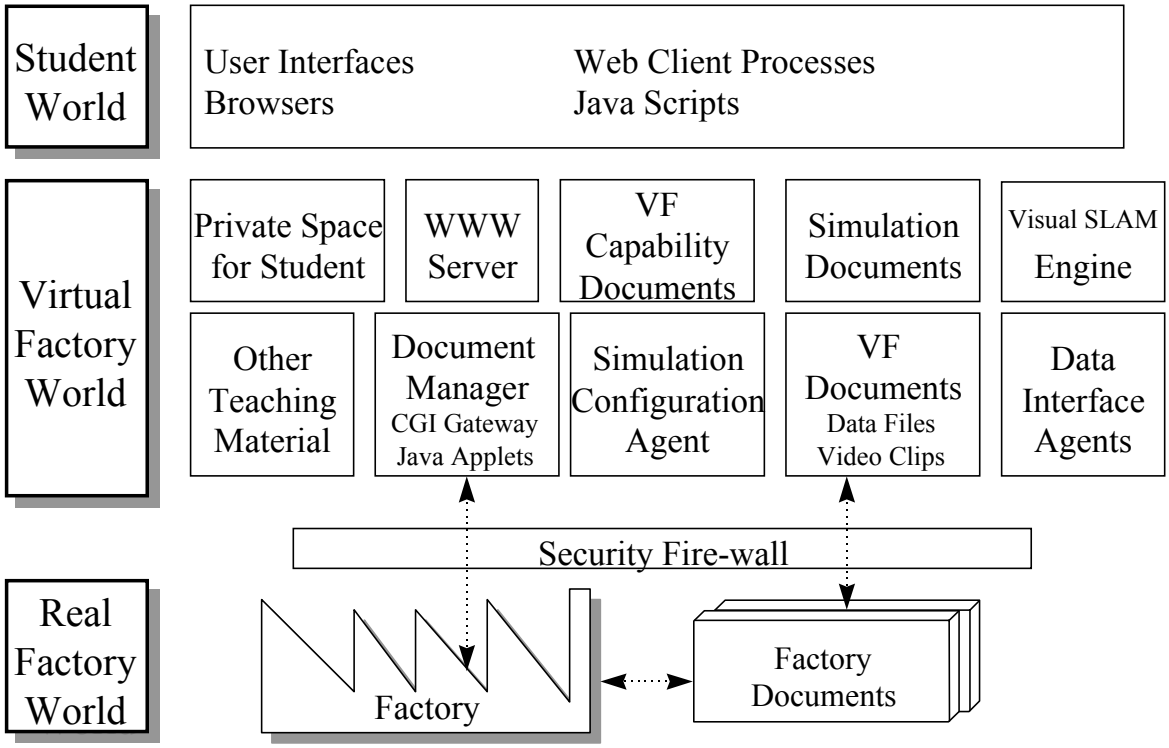


Figure 1. Software Components of the Virtual Factory Teaching System

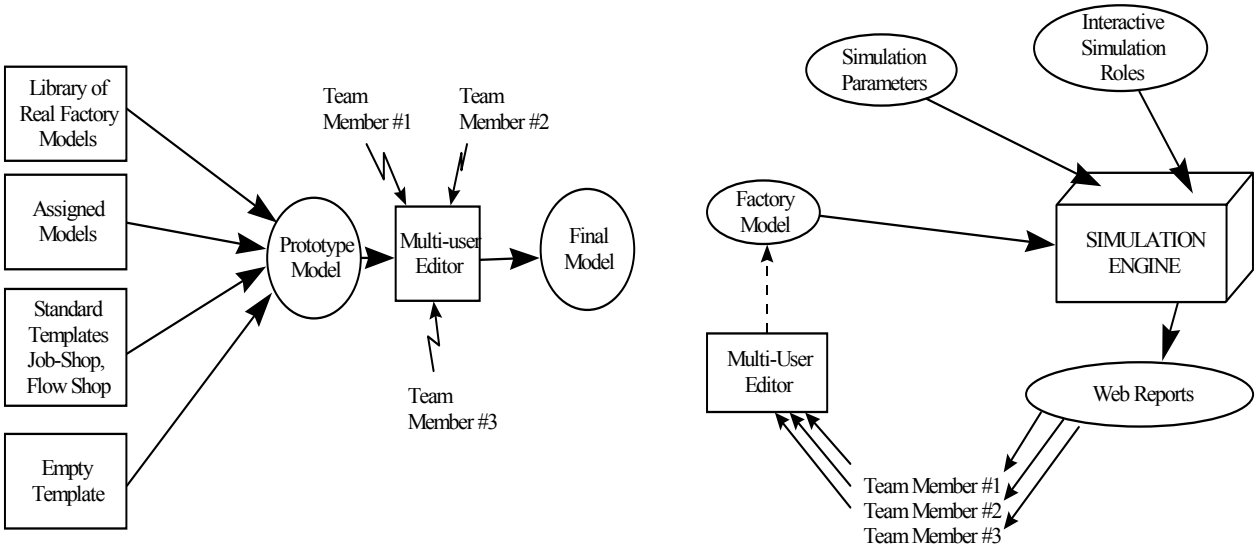


Figure 2. (i) Building a Factory Model (ii) Using a Factory Model

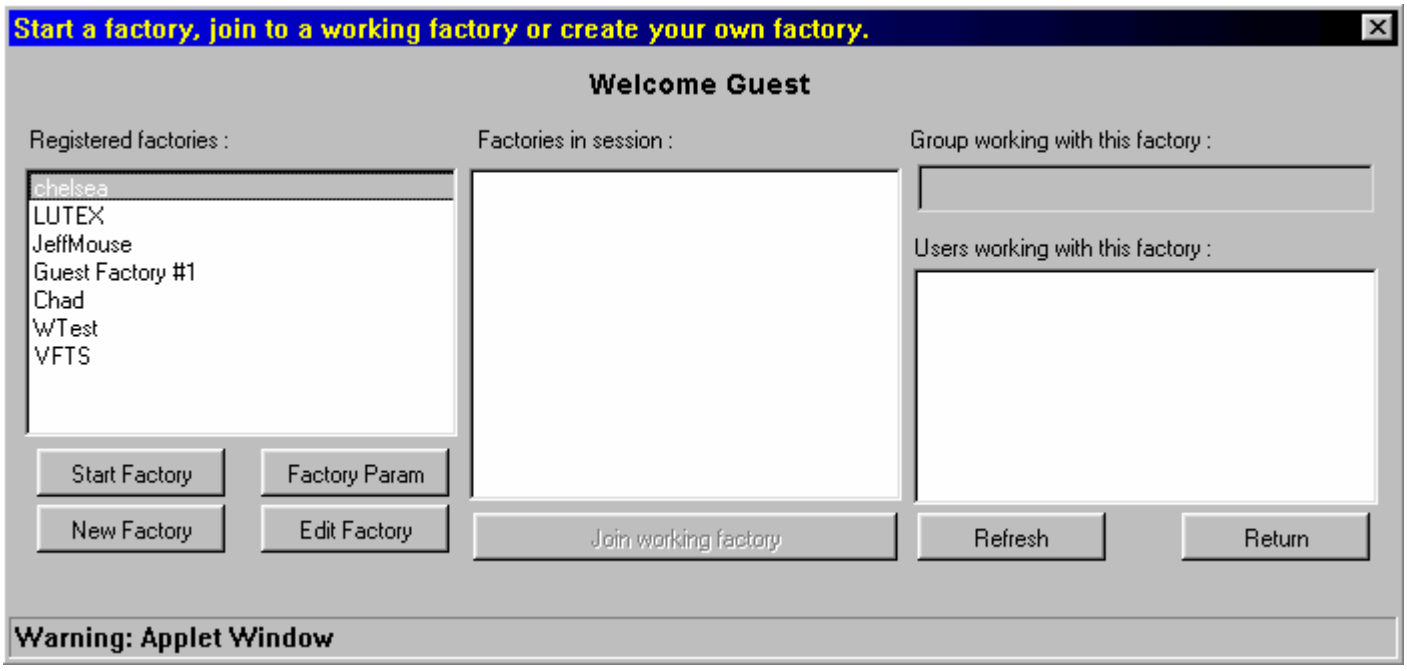
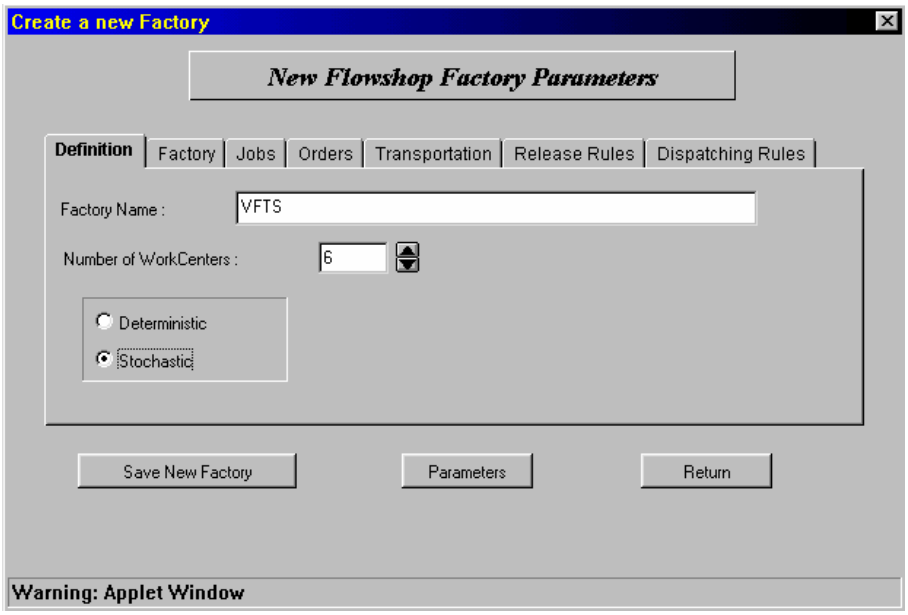
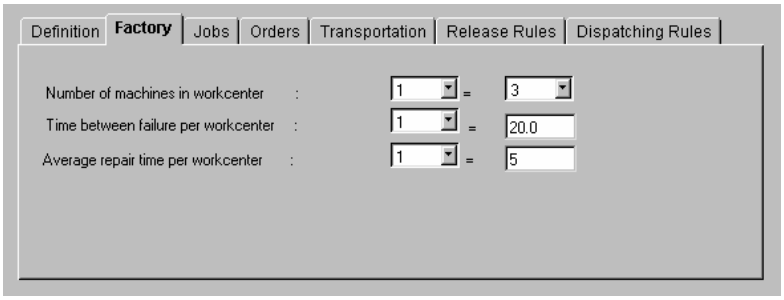
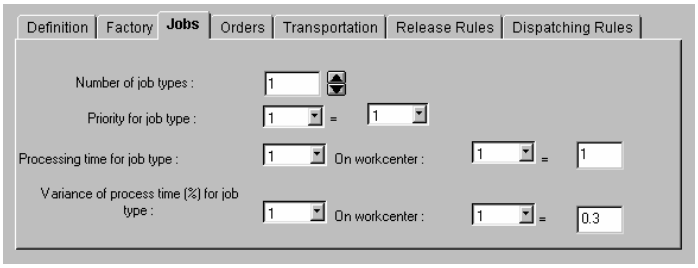
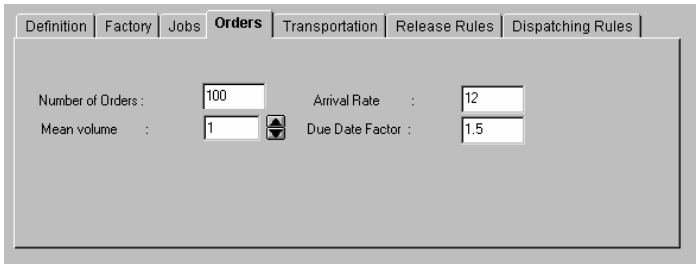


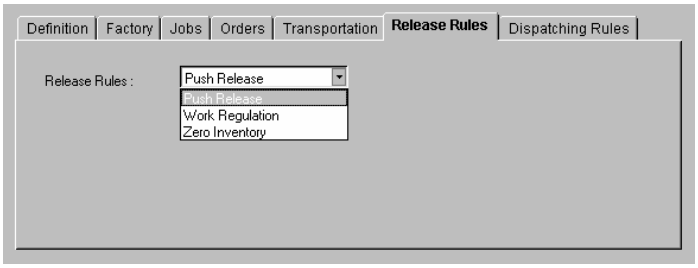
Figure 3. VFTS Factory Start Windows











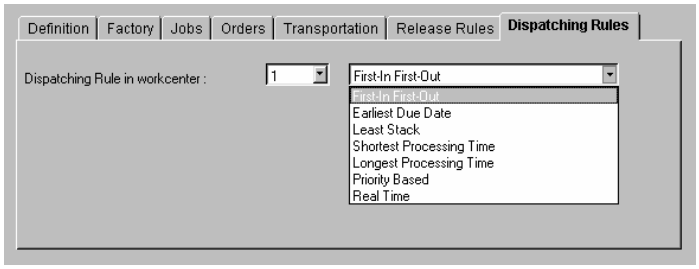


Figure 4. VFTS Factory Definition Windows

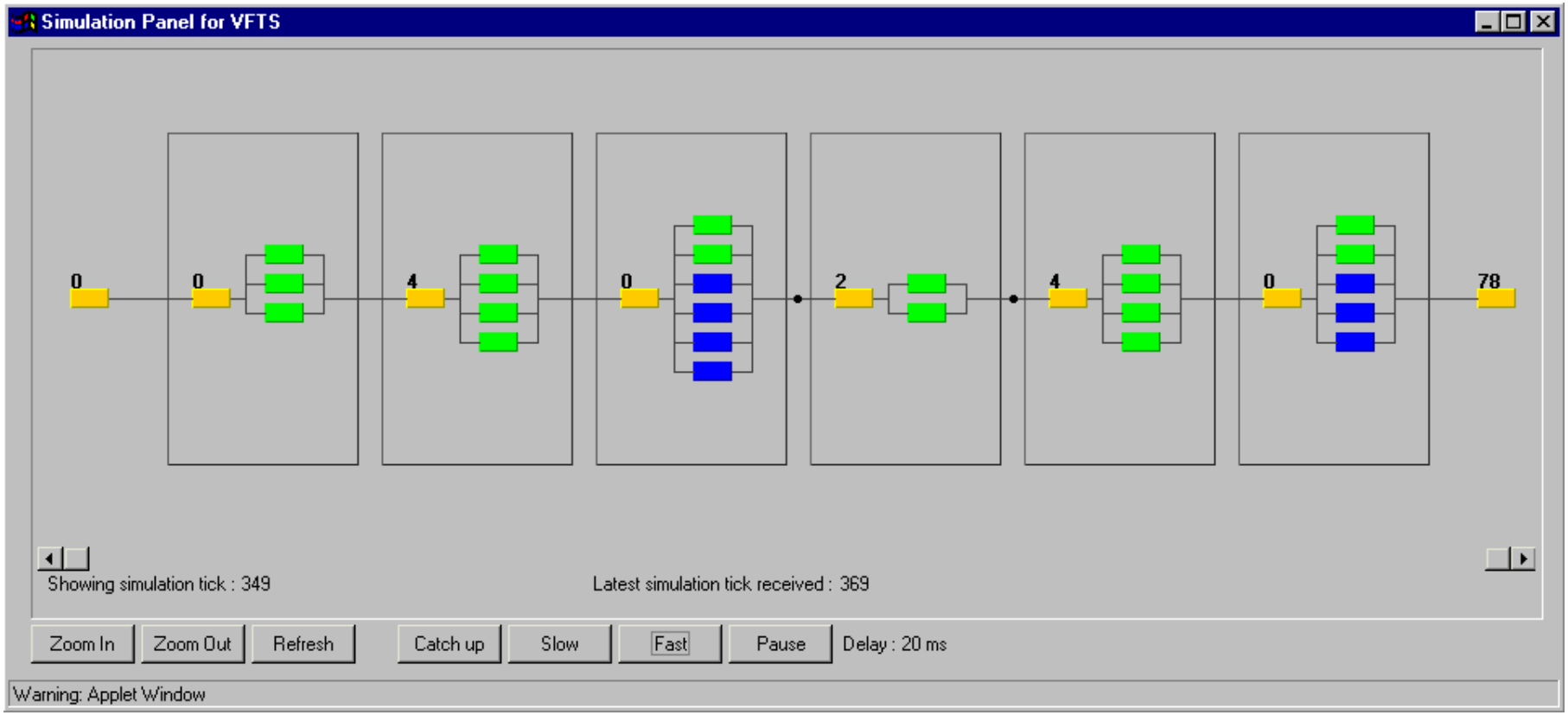


Figure 5. VFTS Animation Window

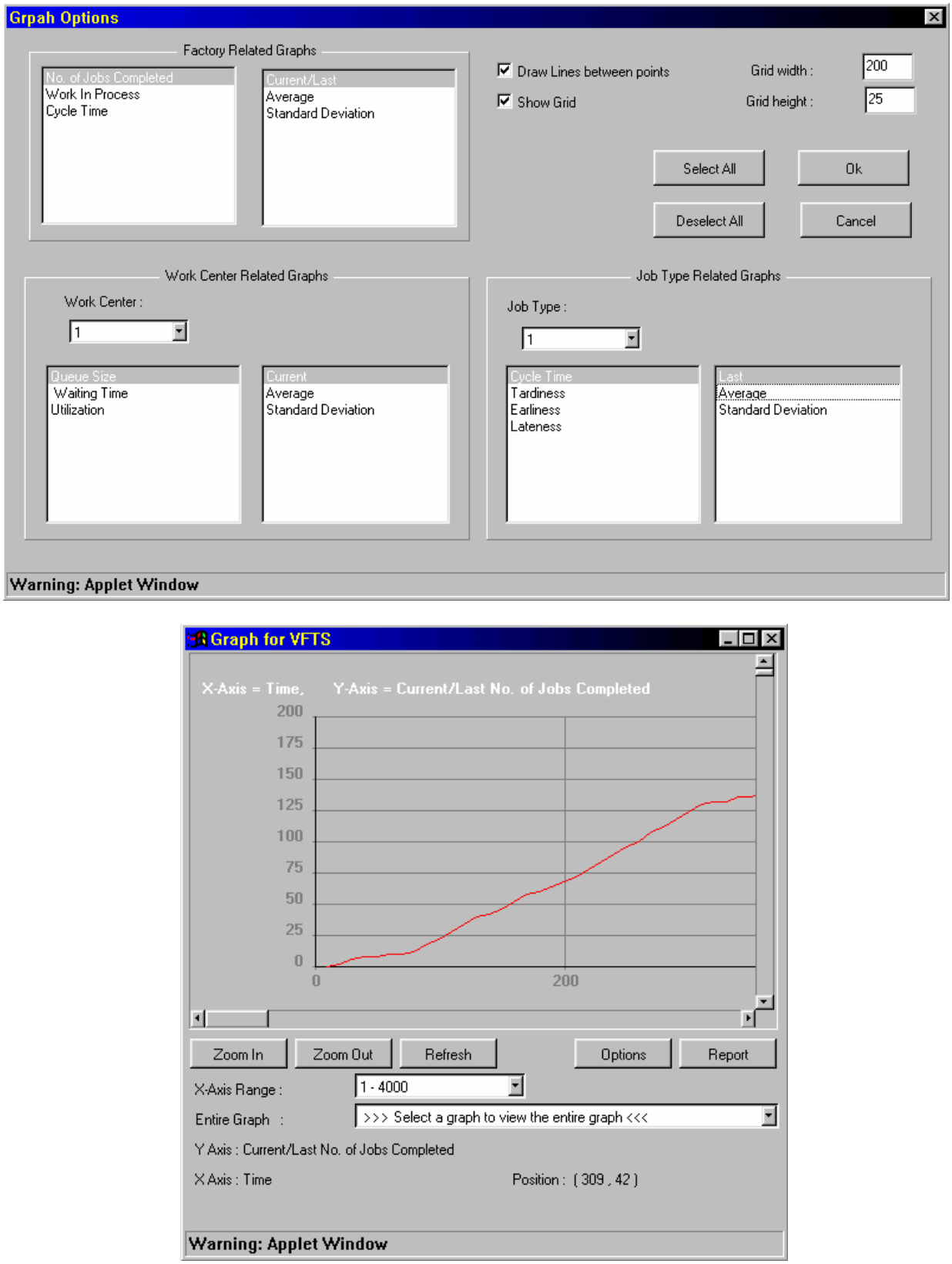


Figure 6. Graphical Windows