Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

SIMULATION, ANIMATION, AND SHOP-FLOOR CONTROL

Cynthia Erickson
Antonie Vandenberge
Trevor Miles
Systems Modeling Corp.
248 Calder Way, Suite 300
State College, PA 16801
(814)238-5919

ABSTRACT

It is often desirable to link shop-floor hardware directly to a discrete-event simulation model or graphical animation. Four situations in which this type of arrangement may be beneficial are identified here, and several mechanisms for implementation are discussed. A primary application involves testing the planned control logic for a specific manufacturing system. In this case, a simulation model, linked directly to one or more programmable controllers, provides the controller(s) with system scenarios under which they are expected to perform and produce a response. A second application is system emulation, where real-time data is used to drive an animation. In this way, a visual representation of system status is provided to monitor shop-floor activity. This may be especially useful in situations where the system is monitored from a remote or central control area. The ability to simulate ahead from current shop-floor status is the foundation for the final two applications that are considered here. Used for contingency control, the effects of alternate control strategies which may be imposed when some critical event occurs, e.g. machine breakdown or expedited orders, may be studied and evaluated. Finally, alternate production schedules may be simulated and compared when the initial schedule is prepared, or updated and re-simulated as the shop-floor situation changes.

1. INTRODUCTION

Simulation analysis and, more recently, animation have become powerful tools in manufacturing systems analysis. Animation now supplies the analyst with a means to graphically depict the simulation of the system's operations and their interactions. Simulation analysis can be applied to such diverse situations as distribution systems spanning across a country, scheduling of parts and resources on a shop floor, and control of physical and electronic devices needed to guide a part through an automated manufacturing system. To date, animations have been "driven" by some form of a simulation system. What is depicted on the graphics screen is influenced only by the simulation, not by the real system.

Emulation has been used to describe graphical systems displaying the current status of the manufacturing shop floor. This type of emulation can be accomplished by directly interfacing the graphics to the logical control sequences of the shop floor controllers. A logical step is to combine this technology with state-of-the-art simulation capabilities, such that the data needs of a simulation run are derived from both the shop-floor devices being modeled and the underlying simulation (language) itself. This method requires an interface between a simulation language and factory-floor devices. The simulation language and the shop-floor hardware must both have a means (such as an RS 232C interface) to communicate with other programs or devices. These interfaces currently exist and are considered "off the shelf" items by many of the programmable controller manufacturers. Some of these manufacturers sell emulation systems as well, and have long since broken the communications barriers among shop-floor devices. So, the hardware required to tie a simulation language to one or several factory-floor devices already exists; what remains is to modify a simulation language in such a way that "events" in the simulation are generated not only by the simulation itself, but also by the shop-floor devices.

This paper identifies several scenarios in which such a link between factory-floor devices and a simulation language may be desirable. Several mechanisms for implementation are also discussed here.

2. TESTING CONTROL LOGIC

Linking a simulation directly to a programmable logic controller (PLC) provides a means to test the control logic of the PLC. This would ideally be applied when the design of the manufacturing system has been completed, but has not yet been implemented. Once the control logic for the PLC has been written, it must be debugged and tested. Currently, much of this verification takes place on the shop floor once the manufacturing system is in place. A startup phase is usually planned, during which the manufacturing system operates at low capacity, in order to debug the control software. To verify

PLC logic using simulation, a model of the physical system must be developed; however, the timing of some events would be generated by the PLC. In this manner, the effects of certain control scenarios on the rest of the system may be studied using the simulation model prior to implementation.

Inferences about the control logic of the PLC may be made based on the simulation's statistical output and animation. In addition to determining whether the programmed logic is viable, the simulation analysis provides a measure of performance for comparison with other control scenarios. Other control schemes may be tested by reprogramming the PLC ladder logic and rerunning the simulation.

The implementation of this first application is twofold. The simulation vendor must provide the software portion of the interface between the simulation language and the shop-floor devices. Second, a simulation model of the physical system must be developed. This model may retrieve data directly through a hard-wired interface into the I/O of the shop-floor devices, in much the same manner the PLC interfaces with the shop-floor devices. For example, a limit switch is tripped as a part is detected; this input is detected by the PLC's I/O scanner which then updates a register value (sets a bit high or increments a counter) in the PLC's memory. It is then up to the simulation program to read this register value, and process this change as a possible event generation in the ongoing simulation. This may become quite cumbersome if the PLC logic is to be tested before the system and the I/O have been implemented. In this case, the I/O must be completed before any simulation may be attempted.

A more viable alternative is to take advantage of the PLC manufacturers' numerous interface cards available today. Most PLC's can accept additional cards in their I/O racks for purposes such as networking PLC's together or polling register values from PLC memory for transmittal over an RS 232C interface. The hard-wired I/O may be circumvented by having one or more of these interface cards poll or receive information into their respective areas of memory. These then may serve as a database for the simulation. The manufacturing system analyst would be responsible for setting up the interface cards, which would be the same hardware required for any shop-floor report generation. The simulation software must be able to receive data over an RS 232C port and use this data to drive the ongoing simulation.

3. MANUFACTURING SYSTEM EMULATION

A second application of linking simulation and animation to shop-floor control is emulation. Rather than testing logic of individual PLC's, emulation graphically depicts the current status of the manufacturing system. This status is updated in real time as the simulation language uses the shop-floor interfaces to detect changes in the system as processes are completed or new jobs arrive. The primary use of emulation is for remote monitoring of the system's functions, in which a graphical display on an office desk might provide information about system faults, switches sticking, buffers overflowing, etc.

A potential problem in using the simulation/shop-floor interface for emulation arises if data requirements become very large. When an event occurs in the real system, such as the completion of part processing at a workstation, information about the event is gathered in the PLC or an interface card's memory, to be transmitted to or read by the simulation. The machine running the simulation must retrieve this data and interpret it for use by the executing simulation. The simulation model then acts upon the data by changing model variables and updating the graphical animation. If the time to process events is large and many events occur in a short period of time, the simulation model may not be able to provide a real-time display of the physical system.

Emulation systems alone may be difficult to cost-justify. However, when combined with simulation, emulation capabilities form the basis for using the current status of the shop floor as a snapshot from which to simulate ahead, which is the focus of the discussion in the following section.

4. SIMULATING AHEAD FROM CURRENT SHOP-FLOOR STATUS

In addition to verifying PLC logic and emulating shop-floor activity, the interface between PLC and simulation model affords the opportunity to execute simulation runs based on the current shop status rather than on some arbitrarily initialized or empty status. In this way, the simulation model may be used to evaluate sequencing and scheduling alternatives, and/or crisis management techniques. In the latter case, alternate solutions for dealing with breakdowns, expedited orders, and other randomly occurring contingencies may be evaluated as an aid to decision-making. In the former case -- scheduling and sequencing -- the simulation may be executed, perhaps at the

beginning of a shift, to test the effects of alternate scheduling rules or order sequences. Though similar in premise and implementation, the two cases will be discussed individually here.

4.1 Scheduling and Sequencing

The appeal of simulation in the solution of scheduling and sequencing problems lies in the ability to study system performance in a relatively short period of time, while capturing all the nuances unique to a particular system. In general, optimal sequencing solutions (relative to some performance measure) have been found for only a small group of manufacturing systems; furthermore, these solutions rely on assumptions which, in practice, are usually violated. For instance, typical assumptions may include orders with sequence independence or no setups, machines organized as pure flowshops or random jobshops, machines with infinite queue space, or no queue space at all.

Some algorithms have been developed to address very specific configurations, e.g. pure flowshops with infinite buffers. However, many real systems cannot be categorized so simply; rather, they must be described as some combination of factors, making the optimal solution of the scheduling problem an impossible one. In some cases, rules which produce advantageous results when applied to simple systems may also be applied with favorable results to variations of the simple case. However, some performance criteria are subject to such system-specific variations so as to preclude the use of generalized or extendable solutions. For example, tardiness calculations are particularly sensitive to system configuration in the dynamic job shop problem [Baker].

For most systems with multiple constraints or a unique configuration, then, the scheduling problem persists. Simulation has been presented, and successfully used [Miles, Erickson, and Batra], as an option for tackling this problem. Simulation will not necessarily produce an optimal solution -- it merely provides the analyst with a tool to quickly evaluate alternatives. In this way, the traditional use for simulation in the solution of the scheduling problem involves the testing of dispatching rules under steady-state conditions. To accomplish this, a simulation is run repeatedly over a very long period of time -- each time using a new rule or algorithm. The results, relative to some criterion (e.g. average flowtime or average tardiness) are then compared, and a rule for general use in that system may be established. For instance, the results using the Shortest Processing Time (SPT) rule may be compared with the results obtained using First Come First Served (FCFS) rule, where the rule producing the lowest average flowtime is considered to provide a generally advantageous solution. The problem with this approach is that the best solution over the long run may not be the best solution for a specific scheduling period. An even better solution may exist for some finite production window, given specific starting conditions for the system.

So, a more ideal approach to the solution of the scheduling or sequencing problem is to resimulate the alternatives at the beginning of some production window, using the actual starting conditions of the system. Since the simulation need only be executed for the duration of the production window, this can be accomplished in a relatively short period of time. The interface between PLC and simulation model provides a simple mechanism to gain immediate access to the current status of orders and equipment on the shop floor. Since the status is continuously updated in real time, the schedule may be reevaluated as often as is deemed necessary. The simulation model may be used to test the difference between particular sequencing rules or heuristics, or may be used to evaluate specific sequences provided by the analyst. In the absence of well-defined rules or heuristics, an analyst with an intimate understanding of the jobs and the production system often can produce several possible sequences. The simulation of these sequences affords the analyst the opportunity to choose the alternative which best satisfies the performance measure.

4.2 Contingency Control

For the purposes of this paper, consider contingencies to be random events such as machine failures, expedited orders, raw material changes, and other problems that are unavoidable regardless of good system maintenance and scheduling. In fact, these contingencies cannot be considered in a sequencing algorithm because of the random nature of the failures. An exact sequence cannot be generated based on an unknown occurrence. Therefore, resequencing may be necessary in the face of contingencies.

The underlying premise is the same for the simulation of both sequencing and contingency control alternatives -- the current state of the machines, orders, and inventory is captured through emulation, providing the starting point for the simulation and evaluation of alternatives. This affords the opportunity to ask the "What if?" questions, reducing the risk of implementing any particular control strategy. In situations where there are no interim options available to resolve the problem, at least the effects of the contingency over the rest of the scheduling window may be ascertained.

4.3 Implementation

The implementation of the shop-floor/simulation interface for purposes of simulating ahead is the same for each of the applications discussed above. The simulation model must include a mechanism to poll the registers of the PLC to detect changes in the shop-floor status (like the emulation application discussed earlier). Communication is strictly one-way -- the simulation model gathers information and updates the simulation (and animation) without providing any input to the PLC. When a simulation is required, e.g. a breakdown or shift change occurs, a "snapshot" of system status is taken to be used as the starting point for the simulation trials.

Two difficulties arise at this point. First, the production system does not necessarily freeze at the moment that we wish to use the simulation model to simulate ahead. How do we keep track of shop-floor changes while the microcomputer is being used for simulation rather than for gathering data (emulation)? Multitasking presents itself as a possible solution to this problem, although this may degrade the speed with which the simulations are executed. Currently, the absence of multitasking capabilities in microcomputers dictates that separate processors must be employed for the two phases, emulation and simulation.

The second difficulty encountered in simulating ahead is one of model functionality. The simulation models discussed so far, for PLC testing and system emulation, have relied on the logic embedded in the PLC. The simulation model has incorporated only the physical system, without any of the decision-making logic. Furthermore, this type of model can be exercised only in real time since it is subject to the real-time performance and timing of the PLC. Since the value in simulating ahead to test strategies or sequences depends on the ability to do so quickly, the decision-making logic must therefore be embedded in the simulation model rather than relying on the PLC to provide it. This logic must be incorporated in the model in such a way as to be transparent during the emulation phase, but accessible during the simulation phase. While this may increase the modeling effort, ideally a detailed simulation model would already exist, having been prepared to aid in analysis during system design.

5. CONCLUSIONS

The interface between shop-floor hardware and simulation software provides at least four applications as discussed here: PLC program verification, emulation, scheduling and sequencing, and contingency control. PLC program verification offers a powerful solution to the costly and painstaking process of PLC debugging. While emulation may be better provided by other monitoring systems, it may be an added bonus when simulation is applied for the other purposes enumerated above. Simulating ahead for the purposes of sequencing and contingency control provides a useful tool for adapting rules and strategies to the current conditions of the shop floor. While the results are promising, the implementation of any of these applications will require continued investigation, and basic additions to simulation software.

REFERENCES

Baker, Kenneth R. (1974). Introduction to Sequencing and Scheduling. John Wiley & Sons, New York.

Miles, T.I., Erickson, C.J. and Batra, A. "Scheduling of a Manufacturing Cell with Simulation." *Proceedings*, 1986 Winter Simulation Conference, Washington, D.C.

AUTHORS' BIOGRAPHIES

Cynthia Erickson is a systems engineer at Systems Modeling Corp., State College, PA, where she provides consulting in the design and analysis of simulations for manufacturing systems. She earned her Bachelor of Science degree in Industrial Engineering at the Pennsylvania State University, where she is currently pursuing her Master's Degree in Industrial Engineering. Her areas of interest include the use of flexible manufacturing and group technology in the computer-integration of manufacturing systems; the integration of simulation with shop-floor control; and the use of simulation in the planning, design, and scheduling of manufacturing and material handling systems.

Antonie Vandenberge is a systems engineer at Systems Modeling Corp., State College, PA. He received BS and MS degrees in Industrial Engineering from Purdue University. He joined Systems Modeling in July 1987, where he is involved in manufacturing systems analysis and design through simulation consulting services. His current interests include the use of shop-floor information systems and Local Area Networks to aid in the simulation and scheduling of manufacturing and material handling systems. He is a member of IIE and SME/CASA.

Trevor Miles is a software engineer at Systems Modeling Corp., State College, PA, where his responsiblities include simulation consulting, and programming, particularly to update the SIMAN software and as a member of the Cinema development team. He is also a PhD candidate in the

Department of Industrial Engineering at the Pennsylvania State University; his research interests include optimization of stochastic systems, on-line scheduling of FMS systems with simulation, and the interface of programmable controllers with simulation systems. Miles received his MSc degree in engineering from the University of Witwatersrand, Johannesburg, South Africa and his BSc in chemical engineering from the University of Cape Town, Cape Town, South Africa.