A MULTI-PASS SIMULATION-BASED, REAL-TIME SCHEDULING AND SHOP FLOOR CONTROL SYSTEM

by

Young Jun Son, Héctor Rodríguez-Rivera, and Richard A. Wysk

ABSTRACT

Significant shop floor scheduling and control research has been conducted in the past several decades. Moreover, real-time shop floor scheduling and control research also has been conducted in the past 10 years. However, there have been no reports of automated, real-time, online, multi-pass simulation-based shop floor control and scheduling being used in practice.

This paper documents the implementation of a multi-pass simulation-based, real-time scheduling and shop floor control system in the Computer Integrated Manufacturing (CIM) lab at The Pennsylvania State University. This research is based on technologies developed in the RapidCIM project at the Pennsylvania State and Texas A&M Universities, as well as developments described herein.

A multi-pass simulation is composed of two simulations -- a real time simulation and a preview simulation, which runs in the fast mode. The real-time simulation acts as a real-time shop floor controller, whereas the fast mode simulation acts as a real-time scheduler. At each decision point in real-time simulation, a deterministic fast mode simulation is evoked to see what control policy impacts the current system most favorably. This control policy is then fed to the real-time simulation for execution on the system.

The multi-pass simulation system is implemented using Arena 3.01. The implementation scheme issues were investigated in great detail. A Visual Basic Application (VBA) embedded in Arena 3.01 plays an important role in making this implementation possible.

Experiments to evaluate the performance of the proposed multi-pass simulation have yielded throughput increases of 138% over single-pass, single-rule dispatching procedures. For different system instances, there may be even greater improvements in throughput performance.

Keywords: Shop Floor Control, Simulation-based Control, Real-time Scheduling, and Simulation
1.0. INTRODUCTION

1.1. Background

Since the inception of flexible manufacturing systems (FMS), manufacturing companies have been investing large sums of money in advanced production facilities. These facilities include computer hardware and software, robotics, automated material handling, machine tools, and data management and communications equipment.

The primary motivation for these investments has been the perception that profits will increase. This increase would be the result of better quality, less direct labor, faster throughput, greater machine utilization, and a more flexible production environment. Unfortunately, these expectations have not been met in many cases. One of the primary reasons for this is the difficulty involved in scheduling and controlling manufacturing activities in a dynamic environment on the shop floor. Moreover, the increased flexibility provided by these new manufacturing technologies has magnified, rather than reduced, planning, scheduling and control problems.

Traditionally, scheduling problems have been formulated using analytical methods like mathematical programming or network theory. These methods often provide optimal solutions for small sized problems under simplified assumptions that do not reflect actual practice. New scheduling approaches rely on the symbolic processing power of computers and thereby enable the incorporation of qualitative information. A current much studied approach has been the use of simulation technology for real-time scheduling. Real-time scheduling can produce good schedules that are reactive to the system status.

"Real-time" means instantaneous response to any event. In a manufacturing process an event could be an arrival of a part, a machine breakdown, preventive maintenance, or completion of a process. Harmonosky and Robohn (1991) defined real-time as a "...response that occurs rapidly enough to react to the system without interrupting its operation." The idea of real-time scheduling is to make a decision fast enough so as not to disrupt the operation of the system.

Shop floor scheduling deals with a dynamic environment characterized by various unexpected events, such as machine breakdowns and frequent changes in order quantities and due dates. These unexpected changes in demand and quantity often necessitate the revision of the whole schedule. One of the challenging issues in revising the current schedule is to be able to predict the effect of a local decision on the global schedule. Provided that the simulation model
reflects the current system status, it can be used in decision support mode with look-ahead system assessment capabilities.

1.2. Simulation for shop floor control

Traditionally, simulation has been applied to long-term planning and design of manufacturing systems. These models have been termed "throw away models" because they are seldom used after the initial plans or designs are finalized (Thompson 1994, Drake et al. 1995). In this development cycle, the same control logic is developed twice: once for use in the simulation, and then again for the control system.

Wysk et al. (1992) proposed that once the system design has been finalized, the simulation that was used for evaluation can be used as the basis for the control system. This approach has been developed as part of the RapidCIM project [Wysk et al. 1992] and has been implemented at The Penn State CIM lab and Texas A&M Computer Aided Manufacturing (TAMCAM) lab. The work is intended to identify the implementation problems and document the research issues associated with simulation-based control.

1.3. Definitions

For clarity, definitions of terminology are provided in this section.

Definition 1. Scheduling problem: selecting the best dispatching strategy, among feasible alternatives, for the next short time period $dt$, in order to meet a system criteria (e.g. minimize tardiness, mean flow time, total completion time, etc)

Definition 2. Multi-pass scheduling algorithms: scheduling algorithms that deal with the form of scheduling problems defined in Definition 1.

Definition 3. Simulation-based scheduling: employing discrete event simulation to evaluate certain dispatching strategies for a short planning horizon. According to Definition 2 and Definition 3, to use multi-pass simulation based scheduling is to employ discrete event simulation to evaluate a set of feasible dispatching strategies for a short planning horizon.

Definition 4. Real-time scheduling: making production decisions fast enough so as not to disrupt the operation of the system.

Definition 5. Simulation-based shop floor control: using a simulation model for design, performance analysis and planning, also for direct shop floor control. In this way, the same
control logic is reused and need not be implemented multiple times.

**Definition 6.** According to Definitions 1 to 5, multi-pass simulation-based, real-time scheduling and shop floor control is to employ discrete event simulations (one in real-time and a preview simulation in fast-mode) to evaluate a set of feasible scheduling procedures fast enough so as not to disrupt the operation of the system and then to control the shop floor based upon the best operating policy found.

1.4. Objectives

Significant shop floor scheduling and control research has been conducted in the past 20 years. Moreover, real-time shop floor scheduling and control research also has been conducted in the past 10 years. However, there exists no automated, real-time, on-line, multi-pass simulation-based shop floor controller and scheduler.

The objective of this research is to finalize the design and implement a multi-pass simulation-based, real-time scheduling and control system in the computer integrated manufacturing (CIM) lab at The Pennsylvania State University. This research is based on the RapidCIM's project models and technologies developed by the Pennsylvania State University, Texas A&M University, and Systems Modeling Corporation as presented in Wysk et al. (1992), and Smith et al. (1996).

1.5. Problem Description

*Figure 1* shows the development cycle of a multi-pass simulation-based, real-time shop floor controller. As shown in *Figure 1*, simulation can be used for two purposes: analysis and shop floor control. This paper focuses on the last two parts -- real-time simulation and rapid previewing using simulation.

In this paper, real-time simulation implementation issues, which were part of the RapidCIM work, will be explained to help readers understand simulation-based shop floor control. Then, multi-pass simulation-based scheduling and control, our main contribution, will be discussed in detail.
1.6. RapidCIM Overview

The RapidCIM project was intended to facilitate the process of making a fully automated, computer controlled, flexible manufacturing system (FMS) operational once the hardware is in place. The specific project focus was on the automatic generation of execution software required for the control of such systems, and tools to assist in the development of the FMS control software.

RapidCIM developments provide generic control elements or modules with well-defined interfaces such that the control modules can be developed independently, i.e. plug-ins. These well-defined control elements, generated using RapidCIM concepts, are very flexible and reconfigurable to adapt to changes in the environment through changes to data and/or regeneration of control software.

In the RapidCIM approach to FMS control software development, the controller functions are partitioned to separate the execution functions from the decision making (planning and scheduling) functions. This separation allows modular development of the execution and decision making functions. In addition, the first phase of the project focuses on reducing the controller development time further by using the simulation model for the decision making function. Thus, the simulation model can be developed only once and used both for the purpose of simulation and then for the purpose of real-time control. This dual use of the simulation model reduces the time needed to build the decision-making module of the controller.
2.0. LITERATURE REVIEW

Wu and Wysk (1988) created a "Multi-pass Framework" for simulation based control. They combined a learning system with simulation. The manufacturing control system learned from its own historical performance. Wu and Wysk (1989) later presented a multi-pass scheduling algorithm using a mechanism controller and a flexible simulator. Multi-pass scheduling algorithms are defined as the scheduling algorithms that deal with the scheduling problem of selecting the best dispatching rule, among rules in an alternative space. Studies have shown that combinations of dispatching rules over a system's production cycle can produce better performance than a single rule alone [DarEl and Wysk (1982), Wu and Wysk (1988), and Drake et al. (1995)].

Smith and Joshi (1992) describe the structure of the control software for the RapidCIM project. The authors explain how the controller can be derived from a graph specifying the behavior of the system from a part's perspective. Wysk, Peters, and Smith (1994) describe how process plans are used in a shop floor control environment. A generic vision of process planning for control is presented. Flexibility in developing and testing operating policies for manufacturing systems requires the ability to change a simulation model to incorporate the new policy quickly and easily. To achieve this goal, Smith and Medeiros (1995) propose separating the model of the system control strategy from the model of the physical system.

Generally, scheduling is difficult, both theoretically and practically speaking. Operation Research (OR) tools and Artificial Intelligence (AI) concepts have been applied to the scheduling problem for a long time. Dutta (1990) presents a knowledge-based methodology to automatically take corrective action when exceptions (interruptions) occur. Kim (1990) compares job shop dispatching rules when a job shop has alternate routings. Maley et al. (1988) conceptualized a closed-loop control structure for the scheduling and control of a CIM system. Perkins and Kumar (1989) achieved real-time implementation by allowing each machine to make localized scheduling decisions based on their buffer levels.

Other researchers (Cho and Wysk 1993) have suggested using neural networks to identify candidate rules for multi-pass simulation analysis. They define five types of scheduling problems in the context of an automated workstation. At each decision point, the neural network generates candidate rules for each problem type, and these rules are then evaluated through simulation.

In recent years, there has been an increasing trend towards integrating the simulation
model of a manufacturing system with the shop-floor control system. Computer technology has significantly reduced computation time for evaluating scheduling methods, and now simulation can be used as a tool for real-time evaluation in many circumstances. Simulation can be a part of the Real-time, decision-control system of a manufacturing system. It allows the incorporation of dynamic changes on the shop floor into the scheduling function. It also has look-ahead capacities, and it helps to determine the effect of decisions on the future performance of the system.

Harmonosky and Robohn (1991), reviewing the work in the area of Real-Time scheduling, examined implementations using general techniques, simulation, Artificial Intelligence and a combination of simulation-AI techniques. The benefits of simulation for the real-time and decision-control of a manufacturing system were first proposed by Davis and Jones (1988). On the basis of Davis and Jones' work, Kim and Kim (1994) built a model in which dispatching rules were dynamically varied based on evaluation of the candidate rules. Harmonosky and Robohn (1995) investigated the application of simulation to real time control. The study focused on the effect of various physical attributes of a manufacturing system on the CPU time needed for simulation.

Church and Uzsoy (1992) addressed the problem of rescheduling production systems in the face of dynamic job arrivals. Using simple single and parallel machine models to gain insight, the authors provided worst-case and computational analyses of periodic and event-driven rescheduling policies.


A different scheduling and control environment was proposed by Kim, Bae, Song, and Lee (1995). The authors suggest a market-like framework for scheduling and shop floor control in computer-controlled manufacturing systems. In this environment, each resource agent and part agent acts like an independent profit maker.

Tunali (1997) presented a simulation model of a job-shop type FMS developed to investigate how the performance of scheduling decisions (i.e., mean job flow time) is affected by the use of flexible or prefixed part process plans in case of a machine breakdown.

Smith et al. (1994) examined the application of discrete event simulation to shop floor control of a flexible manufacturing system. In their research, simulation is used not only as an
analysis and evaluation tool, but also as a task generator to control the physical equipment on the shop floor. A simulation of the physical system acts as a decision-maker to determine what task should occur next and sends that task to the system's execution software.

In spite of significant research of real-time scheduling and control for last 20 years, there exists no automated, real-time, on-line, multi-pass simulation-based shop floor controller and scheduler.

3.0. SIMULATION BASED SHOP FLOOR CONTROL

In this section, simulation-based, real-time control implementation at the Penn State CIM lab is described. Discussion of the multi-pass simulation will be postponed until Sections 4 and 5.

3.1. Introduction to Simulation-based Control

The Penn State CIM Lab is currently being used for implementing a real-time supervisory controller. The control software drives the shop in a completely automated manner. The software can also interact with human operators.

In this research, an Arena 3.01 simulation with the RT (real-time) template is used to model the operating logic of the system and to act as a real-time shop floor controller. Arena is interfaced with individual equipment controllers and a communication system developed as part of the RapidCIM project.

3.2. Penn State CIM Lab

The Penn State CIM lab consists of material processors, material handlers, a material transporter, and buffers interacting together under the control of an Arena simulation following specifications from a database to produce different parts.

*Figure 2* shows the resources that form part of the CIM Lab as it is currently laid out. Raw materials are stored in the KARDEX vertical material storage and retrieval system. The GMF A0 material-handling robot is used to store and move parts to and from the KARDEX. The parts are transported among the different stations using the CARTRAC material transport system. The simulation controls the carts in the CARTRAC system to move around the system continuously after delivering parts in order to be available and avoid deadlocks.
3.3. Overview of the Simulation-based Control System

The Arena simulation model obtains part orders and part process plans using an SQL connection to an interactive database built in Microsoft Access 97 -- an ODBC compliant database system. The database keeps track of part orders and how many parts in each order are finished. The real-time multi-pass simulation system, explained in the following section, will eventually replace the current First In First Out (FIFO) dispatching rule in allocating resources over time. The simulation controls the manufacturing system by sending and receiving messages using an Ethernet communication link to a high-level task executor, known as the BigE. The BigE performs the shop-level execution functions and keeps track of the status of each material processor, material handler, and the storage system. The BigE receives instructions (messages) from the simulation and based on the system status, sends messages to the equipment level controllers. After an order is sent from the BigE, both the BigE and the simulation wait for a "completion_ok" message from the equipment level controller that received the message. Once the BigE receives the "ok" message, it sends a similar message to the simulation, and the simulation knows that the current task was completed. The task generator and execution modules communicate through the task input queue (TIQ) and the task output queue (TOQ). These queues facilitate the explicit separation of the decision-maker from the execution module. This control architecture appears in Figure 3.
3.4. Arena Model Specifics

The simulation model is made up of three submodels:

- The first submodel creates an entity every minute. This entity will read the database for new orders to be produced. Visual C++ code was written to interface between the real-time simulation model and the database. This code is dynamically linked to the model through the dynamic link library (DLL) file and essentially reads the orders and recipes (process plans) from the database (implemented with Microsoft Access 97).

- The second submodel represents the CIM Lab and contains real-time blocks to interact with the control programs that control the material handlers and material processors.

- The third submodel controls interaction between material transportation and the second submodel. Here the requests are assigned to each transporter. The flowchart in Figure 4 summarizes the logic for the third submodel. When a cart reaches a requesting or receiving station, it asks whether the next operation is a "pick" or a "put". If it is a pick, it updates the status of the part's former position to empty. If it is a put, this submodel signals the BigE the
Once the cart finishes the requested operation at a station, it determines whether any other request has been placed on it at that same station. If there is a second request that can be performed at the same station, the cart stays there until the next pick or put operation is completed. If there is no other request from that station, the cart determines whether it can accept any new requests. If a new request is accepted, it again determines whether the request is for the current station. If not, it leaves the station. Before a cart leaves a station, it verifies all the requests placed on it and goes to the closest station from which there is a request. This section of the simulation is considered to have the most room for research and improvement. This logic is encoded within the Arena Simulation, and messages are passed via the execution function created via the MPSG (message-based part state graph) defined by the RapidCIM requirements.

Figure 4: Flowchart for the third Submodel
When an entity enters the first submodel, the database is read and an entity representing an order is passed through the submodel. That entity is later duplicated as many times as needed to create one entity for each part in the order. Every entity has attributes to store the information needed to produce the order that it represents. The names and numbers for the attributes in the simulation are listed in Table 1.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Attribute name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Process Plan Pointer</td>
<td>This is a pointer to the current step in the structure that represents the process plan in memory.</td>
</tr>
<tr>
<td>101</td>
<td>Part Type</td>
<td>This is the order type. A number that indicates the process plan to be used to produce the order.</td>
</tr>
<tr>
<td>102</td>
<td>Order Detail ID</td>
<td>The order identification number</td>
</tr>
<tr>
<td>103</td>
<td>Order Size</td>
<td>The size of the order that is going to be produced.</td>
</tr>
<tr>
<td>104</td>
<td>Conv_loc</td>
<td>Destination for the CARTRAC</td>
</tr>
<tr>
<td>105</td>
<td>Robot_loc</td>
<td>Target location for the robot</td>
</tr>
<tr>
<td>106</td>
<td>NC_Program</td>
<td>The NC file that has to be executed</td>
</tr>
</tbody>
</table>

3.4.1. Event Blocks

Whenever an entity in the simulation model reaches one of these blocks, the function “cevent” in the interface code is executed. There are six types of events implemented and they are described briefly in the following section. In Arena 3.01, an event block can be implemented in either Visual C++ or Visual Basic Application (VBA) embedded in Arena software. In our model, events 1 to 5 are implemented in visual C++ 5.0 and event 6 is implemented in VBA.

3.4.1.1. Event 1

This event is triggered by the entity that is periodically created once per minute by the first submodel of the simulation. It reads the orders from the master production database. The code sends an SQL command to the database to retrieve the order identification number, process plan and volume for each order stored in the database.

3.4.1.2. Event 2

This event is executed when an entity representing a new order enters the system for the first time. Process plans can contain alternative sequences of steps for manufacturing a product.
They are represented as AND-OR graphs and are stored in the database. Whenever the simulation needs to access a process plan for a new product, the associated table is read from the database, and a graph is internally built to keep this AND-OR graph in memory, containing all the branches (options) in the process plan.

Whenever a step in the sequence that makes up the process plan is completed, this process plan (internally stored as an AND-OR graph) is accessed to determine the next operation to be performed. Given a current step in the process plan, several options may exist. In the current implementation, the first option is chosen. However, a planner/scheduler module could be added to decide which alternative should be executed next. Finally, this event updates the process plan pointer to point to the next operation.

3.4.1.3. Event 3

This event determines the next step in the process plan. It is triggered when an operation is completed.

3.4.1.4. Event 4

This event will delete an order from the database. When all the steps in a process plan have been completed, this event is used to delete that order from the database.

3.4.1.5. Event 5

This event activates a look-ahead simulation. Once a look-ahead simulation is activated, the look-ahead manager written in Visual Basic Application (VBA) in the fast-mode simulator will automatically run a number of alternative models and write each resulting throughput to the specific file. This event will be described in greater detail in Section 5, in conjunction with communication between two simulations.

3.4.1.6. Event 6

This event compares the performances of the alternatives, chooses the best operating policy, and then applies it to the real-time simulation. This event is implemented using VBA embedded in Arena 3.01. This event will also be described in greater detail in Section 5.
3.4.2. Real-time blocks

The simulation model uses the following blocks to send and receive messages to and from the lower level of the control hierarchy (the Big Executor). The real-time blocks used in the on-line simulation model were the message, delay, and transport blocks. Smith et al. (1994) described Arena/SIMAN language modifications to facilitate implementation of the physical control equipment. In this section, we will focus how real-time blocks work in real-time mode and Section 5 will discuss how each real-time block works differently in real-time mode and fast time mode.

3.4.2.1. Messages block

All the messages that the simulation will send must be listed in the MESSAGE block, including the parameters that will be sent in those messages. These messages correspond to the ones that the Big Executor accepts, including the name of the command.

3.4.2.2. Delay block

When an entity arrives at a delay block, the message associated with the delay is sent to the Big Executor and to the corresponding equipment level controller. The entity waits in the delay block for a response from the BigE, which is waiting for a response from the equipment level controller that received the message. Once the message is received, the entity continues on to the next block.

3.4.2.3. Transport block

When an entity arrives into a transport block, the message associated with the transport is sent to the BigE and to the CARTRAC controller. The entity moves to the station specified in the block and waits there for a response from the BigE, stating that the cart has arrived at the station. Once the message is received, the entity that was waiting in the transport block continues.
4.0. MULTI-PASS SIMULATION-BASED CONTROL

A stand-alone real-time simulation for shop floor control was described in Section 3. In this section, an architecture of the multi-pass simulation is discussed. Since real-time simulation has already been described in the previous section, this section will focus on inter-simulation interactions and significant parameters for the multi-pass simulation along with their effects.

4.1. Overview of the Multi-pass Simulation

The architecture of the multi-pass simulation-based, real-time scheduling can be seen in Figure 5. Figure 5 describes in detail the “scheduler” block in Figure 3. It should be noticed that two identical simulation models exist on two different computers. One is used primarily as a real-time task generator to the shop floor while the other is used for previewing the system. At each decision point in real-time simulation, a subroutine ("event 5" in Section 3.4.1.5 in our implementation) in real-time simulation activates a fast mode simulation by sending a message to see what control policy impacts the current system favorably. Once the look-ahead manager receives a request, it opens fast-mode Arena and runs the number of alternative models sequentially. Several simulations are run, and the performances of each alternative are recorded in the specific file. When this procedure is done in the fast mode simulation, this control policy is then fed to the real-time simulation for execution of the best control strategy on the system. The execution between the real-time Arena model and the shop floor is identical to that developed by the RapidCIM project (in Section 3). Consequently, the major difference between single-pass simulation-based control and multi-pass simulation-based control is that the fast mode simulation is added to the multi-pass simulation and makes real-time scheduling possible.
4.2. Important Parameters in the Multi-pass Simulation

This section discusses in detail the modeling parameters of a multi-pass simulation. Some of these concerns have also been discussed in Wu and Wysk (1989), Rogers and Gordon (1993), and Dewan (1996). Parameters described in this section can be significant and even critical to the overall performance of the proposed multi-pass simulation mechanism. However, the only way to determine reasonably appropriate values of each parameter is empirically, i.e. through trial and error. Since the main objective of this research is actual implementation of the multi-pass simulation, investigation of the effect of each parameter is outside the scope of this research. Nonetheless, potential effects of each parameter will be discussed.
4.2.1. Does Real-time Simulation need to be inactive while looking ahead?

Does real-time simulation need to be inactive and wait while the fast mode simulation looks ahead? Depending on the decision in this question, the performance of the multi-pass simulation and actual implementation may be different. If the real-time simulation runs while the multi-pass model looks ahead, the multi-pass simulation is guaranteed to perform at least as well as the single-pass simulation. However, the actual performance and the predicted performance from the fast mode simulation could be different (new orders arrive, machines fail, etc). On the other hand, if the real-time simulation is inactive during the look-ahead, there exists a slight chance that the multi-pass simulation will perform worse than single-pass simulation. The actual result should be the same as the predicted performance from the fast mode simulation. The importance of this issue depends on the parameters of the multi-pass simulations, which are discussed throughout this section.

In our current implementation, the real-time simulation continues to run during the look-ahead because Arena does not support any command to halt the simulation.

4.2.2. Rescheduling Point

Real-time scheduling should be advantageous because it allows rescheduling if the future performance of a different dispatching rule is better than the current rule. Unfortunately, we do not know what the critical reasons for attempting to reschedule would be. Rescheduling can be done periodically or it can be event-based. Furthermore, the periodic interval or the number of events to trigger rescheduling can significantly affect the multi-pass implementation performance.

In general a reschedule can be triggered by a predetermined event such as a machine breakdown, the arrival of \( x \) number of orders, or the completion of \( x \) number of orders. It can also be triggered by predetermined intervals of time. If the rescheduling point is not appropriately chosen, the following undesirable conditions may be encountered. Some of these effects are related to the simulation window, which will be discussed next.

(1) If the rescheduling point is too frequent, performance will be degraded (if the real-time simulation is inactive while looking ahead).

(2) If the rescheduling point is too infrequent, the scheduling mechanism will not be sensitive enough to switch between different dispatching rules at the right time;
Therefore, a design of experiment should be performed to investigate these factors. In this research, a rescheduling point is arbitrarily determined and a reschedule is triggered every 100 parts. As discussed earlier, this factor could significantly affect the overall performance of the proposed mechanism. Since the focus of this paper is on implementation issues rather than optimizing system performance for our application, we will postpone investigation of performance for future work.

4.2.3. Simulation Window (Fast Simulation Length)

The simulation window determines how far the fast mode simulation should look ahead. Intuitively, the value $dt$ can be critical to the overall performance of the multi-pass simulation mechanism. Wu and Wysk (1989) described the following undesirable conditions if $dt$ is not appropriately designed:

1. If $dt$ is too short, no statistics can be collected to give a reasonable performance measure.
2. If $dt$ is too long,
   a. the scheduling mechanism will not be sensitive to switching between different dispatching rules at the right time;
   b. the scheduling mechanism will only provide average and aggregate performance measures in each period, which may lose the advantage of the multi-pass simulation;
   c. the mechanism will tend to view the system in more of a static sense, which is not desirable.

Therefore, an experimental design needs to be performed to investigate this factor to significant detail. In this research, a simulation length is chosen arbitrarily as 8 hours. As discussed earlier, this factor could affect the overall performance of the proposed mechanism.

4.2.4. Performance Measure

The main objective of companies is simply to maximize the net profit. However, estimating the net profit is very complicated. In order to improve profit, most companies use more than one performance measure. A list of primary performance measures of flexible manufacturing systems (FMS) include throughput, maximum completion time (makespan), mean flow time, maximum flow time, number of tardy jobs, mean tardiness, maximum lateness, etc. It
should be stated that the maximum throughput itself cannot guarantee the optimum profit. When costs of resources are involved and/or part mix is different, the relationship between the throughput and net profit is not obvious.

In this research, due date and cost are not taken into account, and performance is limited to the number of parts of a specific part type produced. With due date and/or cost involved in the system, the problem will be more realistic and interesting. Selecting the candidate alternatives to be used in the fast time simulation depends on the performance measure of the system.

4.2.5. Candidate Alternatives

In a real world manufacturing system, we would have a wide range of alternatives for achieving a better operating policy. However, evaluating alternatives takes time. Given the restriction that we can not run more than one model simultaneously in the Arena software, we must run models sequentially. This limit results from the software "key", which is required for real-time operation. The real-time variable gets set at system initiation and can not be easily changed. In our case, looking-ahead time will increase exponentially with the number of variables to be considered, because we must run a model for each alternative.

In this research, the only variable studied is the initial dispatching rule to the system. Four rules are chosen based on literature by Wu and Wysk (1988, 1989), Kim and Kim (1994), and Baker (1995), and they are First In First Out (FIFO), Number In Next Queue (NINQ), Shortest Job processing Time (SJT), and random (RAN). Other factors also could be concerned as variables such as number of carts in CARTRAC system, number of parts per cart, system load, etc. However, Rivera (1998) conducted an off-line experiment to determine the optimum values of these factors. He concluded that the optimum configuration was 3 carts, 3 parts per cart, and a fully loaded system. This analysis is however system specific, so it would need to be performed for each different system configuration.

5.0. IMPLEMENTING MULTI-PASS SIMULATION

This section will describe actual implementation of the multi-pass simulation. Since designing architecture for the multi-pass simulation was partially influenced by the capabilities of the Arena software, the constraints of Arena are described first.
5.1. Constraints of the Arena Software

- Two modes (real-time mode and fast mode) can not run simultaneously on one computer. Therefore, in order to implement a multi-pass simulation, we need to run real-time Arena on one computer, and run fast mode Arena on another computer.

- Multiple models, even under the same mode, cannot run simultaneously on the computer. Therefore, we cannot run multiple alternatives simultaneously. We must run them sequentially instead.

- Arena does not support a command to halt the simulation. If such a command existed, it would be possible to implement the case of real-time inactive during look-ahead.

- Arena does not support a "restore" command, a run control command. In SIMAN (pre-version of Arena), the "save" and "restore" commands let you save and recall snapshots of the current state of the simulation model. Using these commands, the fast mode simulation could be automatically generated from the real-time simulation. However, because of this constraint in Arena, the architecture proposed by Drake et al. (1995) had to be modified. This is discussed in the following section.

5.2. Real-time Simulation vs. Fast mode Simulation

The only blocks that act differently in real-time mode and fast mode are the real-time blocks described in Section 3. In real-time mode, these blocks actually send messages to the execution module. In fast mode, messages are ignored, and entities only emulate normal blocks such as "delay" and "transport". In spite of this dual mode feature in Arena, the following three things in the real-time model are modified for the fast mode model:

1. The "Event" blocks interacting with the database in the real-time mode had to be removed. With these blocks in the simulation, the model worked in fast mode; however it took 62 seconds to look ahead 10 minutes. The model was extremely slow because it read a process plan in the database for every operation.

2. The "sequences" element had to be created to compensate for the "event" blocks. A sample process plan can be seen in Figure 6. Basically, the process plan specifies the part routing in the real-time mode and provides the delay time for the fast-mode. Therefore, the "sequences" element was required for part routing information and with delay time for each operation.
The "sequence" associated with the process plan can be seen in Figure 6. After replacing the "Event" blocks with the "sequences" element, it took about 16 seconds to evaluate four alternative models to look ahead 10 minutes; whereas, it took 22 seconds to look ahead 8 hours. The look ahead time is a function of the simulation window and the number of alternatives. For each alternative, a constant time (about 3.5 seconds) was taken to open and close the model, to read the input data from the file, and to write the output results to the files. In addition, a variable time $t$ ($t < 2$ seconds if the simulation window is shorter than 8 hours), varying on the simulation window, was taken to run the simulation. The effect of the simulation window to the look ahead time increased very gradually (6 seconds difference between 10 minutes and 8 hours). Therefore, the simulation window should not be a constraint for the look ahead time. The number of alternatives is not a constraint for the looking-ahead time, as long as it is unreasonably high. Since the result was system specific, it cannot be generalized. In general, the looking-ahead time becomes more significant under the following conditions: 1) the processing times in a system are shorter, 2) the rescheduling point is more frequent, and 3) the system becomes more dynamic.

(3) When the fast mode simulation was activated, the current system status as well as order information was required for true system fidelity. If Arena supported the "save" and "restore" command, this could be done easily. In this study, only order information is input to the fast simulation. Although system status can be significant depending on the system, the implementation cost is quite expensive.

---

Figure 6: A process plan and the associated process plan in the fast mode simulation
5.3. Interaction of Multi-pass Simulation

Figure 7 describes the sequence of events in each computer performing a simulation. In this section, implementation of each event is described. As mentioned earlier, the real-time simulation looks ahead every 100 parts. This is implemented within the model using a variable and "scan" block. A counter is initialized to 100 and the number of parts is deducted until the counter reaches zero. When it becomes zero, the order information is recorded as "order.dat", and the simulation requests the look-ahead manager residing in computer #2, and the variable is reset to 100. Writing down the order information and activating the look-ahead manager is implemented in "event 5" which was described in Section 3. The communication between the real-time simulation and the look-ahead manager is implemented using a Remote Procedure Call (RPC). The look-ahead manager acts as a server under the client/server environment constructed by the RPC. RPC will be described more fully in the next section.

Once the look-ahead manager receives the request, it executes the "fastmode.bat" file. This batch-file provides the experiment file (.doe) for the look-ahead simulation. This command automatically opens the Arena software and the specified model. Because noticeable time is required to open Arena, it is better to leave Arena open. Once the specified model is open, it is automatically run for a specified time. This automatic run is implemented in a Visual Basic Application (VBA) embedded in Arena 3.01. The VBA code for this action is as follows:

```vba
Private Sub ModelLogic_DocumentOpen()
    ' Set the global SIMAN variable
    Set g_Model = ThisDocument.Model
    Set g_SIMAN = g_Model.SIMAN
    Set g_APP = g_Model.Application
    g_Model.QuietMode = True
    g_Model.Go
End Sub
```

Once the first candidate model has finished running, it writes the performance measure (throughput in this study) to "out.dat" and opens the next candidate. When all the candidates have been run, the real-time simulation accesses the "output" file (in the network drive), compares the performances, selects the best operating policy, and changes the control policy. As mentioned earlier, Arena does not support any command to halt the simulation. Therefore, the real-time simulation keeps running during the look-ahead. In this research, the real-time simulation reads the "output.dat" file and updates the control policy one minute after the look-ahead was invoked. If Arena supported a halting command, a different communication could have been implemented using the same method.
5.4. Communication between two Simulations

The Remote Procedure Call (RPC) technology supported by Microsoft Corporation was used for inter-simulation communication. The whole procedure for using RPC is summarized in Figure 8. As shown in Figure 8, a short IDL (Interface Definition Language) file automatically generated a large portion of the communication programs. As shown in the IDL file, the names of the functions that we want to call remotely must be formally declared. Here, a remote call means access to functions residing in a remote computer. The automatically generated files provide the framework for a client/server environment. The client portion of the generated programs is combined with "userc.cpp" to produce a DLL file. This DLL file will interface with external applications in the real-time simulation. As shown in the figure, the stand-alone server listens for the client. This server program acts as a look-ahead manager.
6.0. ON-LINE EXPERIMENT

6.1. Running Multi-pass Simulation

Since we need to run two different modes of Arena simulations at the same time, it is worth clarifying physical and environmental settings for each mode. Both sets of settings will be described simultaneously. For running Arena in real-time simulation mode, the following are required:
• Arena 3.01 software, real-time template, and sentinel drive
• Arena key: this key is also required for fast mode because there are limits on the number of entities, variables, etc that we can use without this key
• Declaration and setting two environment variables (In “Autoexec.bat” file)
  The existence of two environment variables specifies Arena's major operation modes (Real-time mode and Fast mode). They are “SM_Realtime” and “SM_Execute”. If “SM_Realtime” exists and is set to 1, the simulation will run in real-time as measured by the computer’s internal clock. Otherwise, the simulation runs in fast time just like traditional simulations. If “SM_Execute” exists and is set to 1, the simulation will execute tasks by writing them to the task initiation queue and waiting for the response from the task completion queue. Otherwise, all the tasks are emulated as features built in Arena. We can set and reset these variables in the “Autoexec.bat” file. The format for setting these variables are “SET SM_Realtime = 1” and “SET SM_Execute = 1”. In resetting these variables, “#” needs to be added to the front of each statement. Under the Windows NT 4.0 environment, it is necessary to log out and log in again to make effective the newly set environment variables.
• Generate a Dynamic Link Library (DLL) file and link it
  There is a standard file "userc.cpp" provided by Systems Modeling Corporation. This file initially contains a skeleton for real-time communication with the control module. User functions can be added. They can be called by "event" blocks in the model.

6.2. Validation of the fast mode simulation

A single experiment was conducted to validate how accurately the fast mode simulation can predict performance of the real-time simulation. In general, in a fully automated manufacturing environment, all the processing times are virtually deterministic. Therefore, the results between the fast mode simulation and the real-time simulation are expected to agree, as long as the data collected are accurate. For our experiment, the performance measure used was makespan to produce 4 part types with 7 parts of each type. The number of part types and parts used in the experiment were arbitrarily chosen because the objective of this experiment was to validate the concept rather than optimize a system. In order to reduce the experiment time, the equipment controllers (Figure 3), emulating the physical equipment, were delayed by one tenth of the expected processing times. The makespan was 449 in the fast mode simulation, as opposed to
a makespan of 531 in the real-time simulation. This difference can be contributed to the reasons listed below:

(1) In simulation, the tasks are modeled in aggregate fashion. While the simulation logic tracks what resources are involved in the implementation of the task, it contains no details regarding how the tasks are actually implemented, which is actually the role of the BigE. For example, the fast mode simulation uses aggregate processing times (NC processing time to run a part program), whereas the BigE (real-time simulation and controller) includes a set of associated lower level tasks, such as downloading the part program, opening the chuck, closing the chuck, communicating with related controllers and then processing the part program. It is difficult to collect accurate data for each lower task. Furthermore, the time for processing an object in the simulation is typically optimistic.

(2) As shown in Figure 2, the equipment level controllers are located in four different computers. Even though the transmit time for each individual message is small, there may be noticeable time delays due to network congestion. As the system gets larger, the impact of this factor may be more serious.

(3) Time taken for database interaction is not considered in the fast mode simulation. In the real-time simulation, each entity (part to be produced) reads its process plan for every operation from the database. The significance of this factor was discussed in Section 5.2.

(4) Time scaling may have influenced the performance of the real-time simulation.

The above factors could also be modeled, but the associated cost and simulation overhead is expensive. If the accuracy of the fast mode simulation is within a reasonable bound, the proposed method will be valid. This is because decisions are made based on the results among the alternatives in the relative sense. However, it is difficult to determine an accurate value of the actual activity times.

6.3. Comparison between the performances

An on-line experiment was conducted to prove the superiority of the multi-pass simulation. Because of the time required to validate and verify "Real-time control" of the Penn State System (the system has to be run to produce actual parts), a single physical experiment was conducted. First, the system was run with a single-pass real-time simulation. Second, the system was run with the proposed multi-pass simulation system. Each part was run for a period of 8
hours. Detailed part information is not described because the objective of this section was simply to show that the proposed controller works better than the single-pass simulation system.

The single-pass simulation produced 13 parts, while the multi-pass simulation produced 31 parts. The multi-pass simulation produced 238% of the throughput of the single-pass simulation, still using the same available resources. This improvement was the result of both the pure efficiency of the multi-pass simulation system and the best operating logic determined from the off-line experiment.

Important parameters in the multi-pass simulation were described in Section 4.2. To investigate each factor in significant detail, experimental designs are required. However, in this research, the simulation window was arbitrarily chosen as 8 hours, and four alternative operating policies were used. For different system instances, there may be even greater improvements in throughput performance.

7.0. CONCLUSIONS

This paper documented the implementation of a multi-pass simulation-based, real-time scheduling and shop floor control system in the Computer Integrated Manufacturing (CIM) lab at The Pennsylvania State University.

The multi-pass simulation is composed of two simulations -- a real time simulation and a preview simulation in fast mode. The real-time simulation acts as a real-time shop floor controller whereas the fast mode simulation acts as a real-time scheduler.

The proposed multi-pass simulation system was implemented using Arena 3.01. Implementation scheme issues were investigated in great detail. A Visual Basic Application (VBA) embedded in Arena 3.01 played an important role in making this implementation possible. If Arena supported other commands such as "save", "restore" and "halt a simulation", a more realistic multi-pass simulation system would be possible to implement.

The current research focused on linear process plans, and no alternative routings were considered. Further investigation of the implementation of alternate routings in a job-type FMS as presented in Tunali (1997) is left for future research.
REFERENCES


