Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

RECENT ADVANCES IN THE MODELING, SCHEDULING AND CONTROL OF FLEXIBLE AUTOMATION

Wayne J. Davis, Duane Setterdahl, Joseph Macro,
Victor Izokaitis and Bradley Bauman

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

ABSTRACT

This paper initially discusses the state-of-the-art and the current limitations in the modeling, scheduling and control of flexible automation. To model flexible automation, it is argued that the simulation tools must provide enhanced capabilities to consider both controller interactions and the flow of resources that support production. It is also demonstrated that scheduling and control must be considered concurrently in real-time to effectively manage flexible manufacturing systems (FMSs). The complexity of the modern FMS further requires that the integrated scheduling and control function must be distributed among several coordinators with the system.

The second part of the paper presents several research developments pertaining to the modeling, scheduling and control of flexible automation including: a Recursive, Object-Oriented Control Hierarchy for the integrated distribution of scheduling and control; a Hierarchical Object-Oriented Programmable Logic Simulator for the detailed modeling of FMSs; and a Hierarchical System Coordinator for implementing real-time scheduling and control. A physical emulator for an FMS which is being constructed is discussed as the example application in this presentation. Finally, the future research plan and the educational program based upon the developments is outlined.

1 INTRODUCTION

The adoption of flexible manufacturing techniques has become a major element of the modern manufacturing strategy. The need for flexibility arises from several sources including the desire to reduce inventories (particularly work-in-progress), the need to customize the product to meet individual customer requirements, and the decreased lifetime for a product design resulting from increased international competition and the emergence of new engineering technologies. Accompanying the adoption of flexible manufacturing techniques, there has been an increased reliance upon automation. Today, even the simplest flexible manufacturing systems will likely include numerous controllers whose coordinated interactions are essential to the production within the FMS.

Although there has been a major modification in the way we manufacture products, the accompanying manner in which we model, schedule and control FMSs has experienced little change. It is clear that the present analysis techniques are now suffering from the inertia of the past. Specifically, the analysis techniques still focus on the job-related entity flow with little or no consideration of the flow of resources that support the manufacturing or the controller interactions that orchestrate the production within a FMS.

In this paper, we first review the state-of-the-art as it pertains to the modeling, scheduling and control of flexible automation. Given the broad area being addressed, it is impossible that this paper serve as a literature review for all the areas. Our discussion will focus heavily upon our experiences in addressing each of these areas. From this discussion, we hope to demonstrate that a major revision in the manner that we view each of these areas is required. We also hope to demonstrate the need that modeling, scheduling and control must be considered as integrated areas and in real-time.

After this need is demonstrated, we will relate our research to address these needs. We will first define a new control architecture for modeling these systems, and then discuss a new approach to modeling FMSs based upon this control architecture. Next we will introduce the concept to the Hierarchical System Coordinator (HSC) which is responsible for the integrated/distributed scheduling and control of FMSs. In discussing the HSC, we will also relate our findings in the area of real-time, discrete-event simulation.

2 THE STATE-OF-THE-ART

2.1 Modeling and Simulation

Due to the discrete-event nature of the FMS, simulation remains the primary tool for the modeling of FMSs primarily. The current simulation tools continue to focus upon modeling the job-related entity flow through an enhanced stochastic queuing network. Historically, discrete-event simulation tools were first developed to assist the modeler in defining, scheduling and synchronizing the occurrence of the events associated with the dynamics of a discrete-event system. The first queuing network simulation packages were then introduced as a vehicle for modeling the assembly lines where sequential processing steps were assumed to occur. At this point, the preoccupation with the job entity flow began while the basic module for modeling a manufacturing system emerged, as depicted in Figure 1. This module consisted of an input queue, a process requiring the allocation of one or more supporting resources, and the output queue. For many other applications (particularly assembly lines), the output queue became the input queue for the next process. In the modeling of this basic element, four primary events were defined:

A_{jn} the arrival of Job_j at process n,
S_{jn} the start of processing for Job_j at process n,
F_{jn} the finish of processing for Job_j at process n, and
P_{jn} the removal of Job_j from process n.

Although today's simulation tools are certainly more sophisticated than their predecessors, the basic building block for modeling the modern manufacturing systems remains nearly unchanged. Perhaps the greatest advancement in the modern simulation packages is the provision of an enhanced capability for modeling the material handling systems which move the job entity among the processes included within the modeled system. Through this enhancement, the current simulation packages are providing an increased capability to specify the arrival event at the successor process n' for Job_j ($A_{jn'}$) given the occurrence of P_{jn} at the current process n.

Our concern with the current simulation tools emerged from attempting to model real-world FMSs to answer

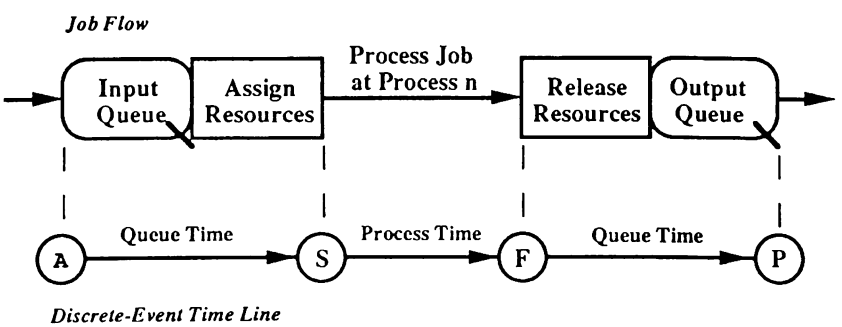

Figure 1. The Basic Job-Flow Module Employed in Current Simulations

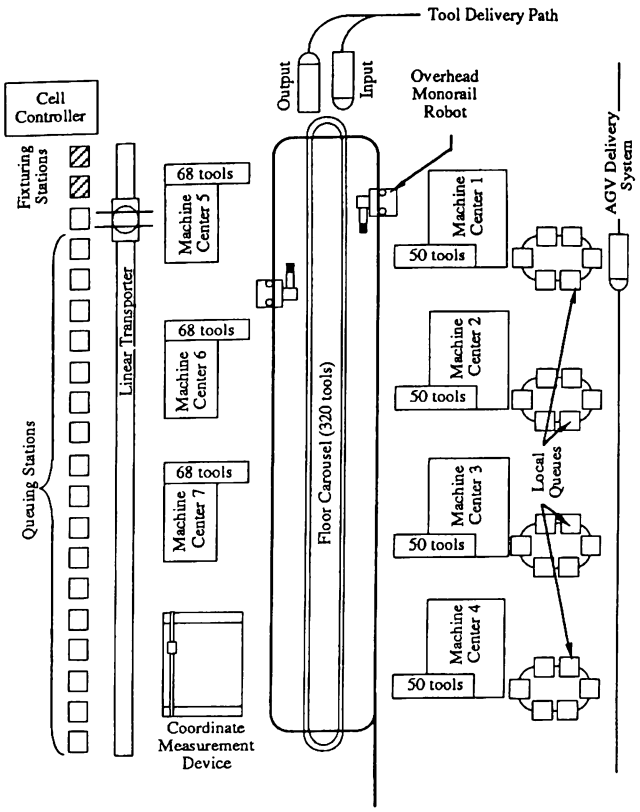

Figure 2. Layout for the FMS Operated by US Army Rock Island Arsenal

questions that are often ignored in most simulation analyses. Consider the FMS depicted in Figure 2 which is operated by the US Army Rock Island Arsenal (RIA). In modeling this system, we desired to assess the consequences of the proposed tool handling system upon the production throughput. Using SIMAN in this case, several problems emerged. First, there were now three primary entity flows to be considered: the flow of job entities, the flow of tool entities, and the flow of fixture entities. All three flows had to be modeled in detail. Second, there was a need to synchronize flow of different types of entities.

Since we were also modeling tool flow, it became essential that we consider the detailed processing plans for each part type to specify which fixture would be required for a given fixturing step; which machines could process the fixtured part; and which tools, in what order and for what duration were required to complete the processing of a given fixturing step at the assigned machine. The proposed tool management system would allow tooling to be shared by machines. Furthermore, the tooling requirements were extensive such that a complete ensemble of tools could not be stored at each machine. In some cases, the

complete tooling for a single fixturing step could not be stored on the machine. Rather the tooling was distributed among the machines, on the off-line floor storage carousel, or at an off-site tool crib. To consider the details of the processing plans, it was necessary to introduce several new events. The modeled events are illustrated along the axis in Figure 3. The Arrival event (A) is again modeled, but here we consider the arrival of both jobs and tools at a machining center. After their arrival, each entity type will flow to its respective queue to await the appropriate processing steps. The Start Job event (S_J) represents the removal of the Job from the input queue and placing it in the work area. Note this action requires the intervention of a dedicated material handling system at the machine center. The Start Task event (S_T) represents the initiation of the processing for the current fixturing step. The processing task requires the implementation of several processing instructions, each delineated by a Start Instruction (S_I) and a Finish Instruction (F_I) event. Each processing instruction requires a specific tool to be placed in the machine spindle and removed from the spindle by yet another dedicated material handling system at the machine center. The Finish Task (F_T) event represents the removal of the job from the work area and its placement in the output queue. The Pickup event (P) is again defined for both job and tool entities when they exit the machine center.

The discrete events described here only address the dynamics at a machine center. There are similar events at the cell level which govern the job in its movement between fixturing operations and its processing at assigned machining centers. Similarly there are also cell-level events governing the movement of tools among the various machining centers and storage devices in the cell. In both cases, it is assumed that there are dedicated material handling systems to perform job/fixture and tool transfers at the cell-level. Figure 3 also highlights another very important element in the management of job and tool flows within a machine center. Each event depicted along the axis represents a point where the direct control of either the job or tool entity is transferred to another controller. The flow of job control is delineated in the Primal Job Flow Plane while the flow of tool control is highlighted in the Dual Resource Flow Plane. These two planes highlight the synchronization that must exist between the controllers such that the associated events along their intersection axis can occur.

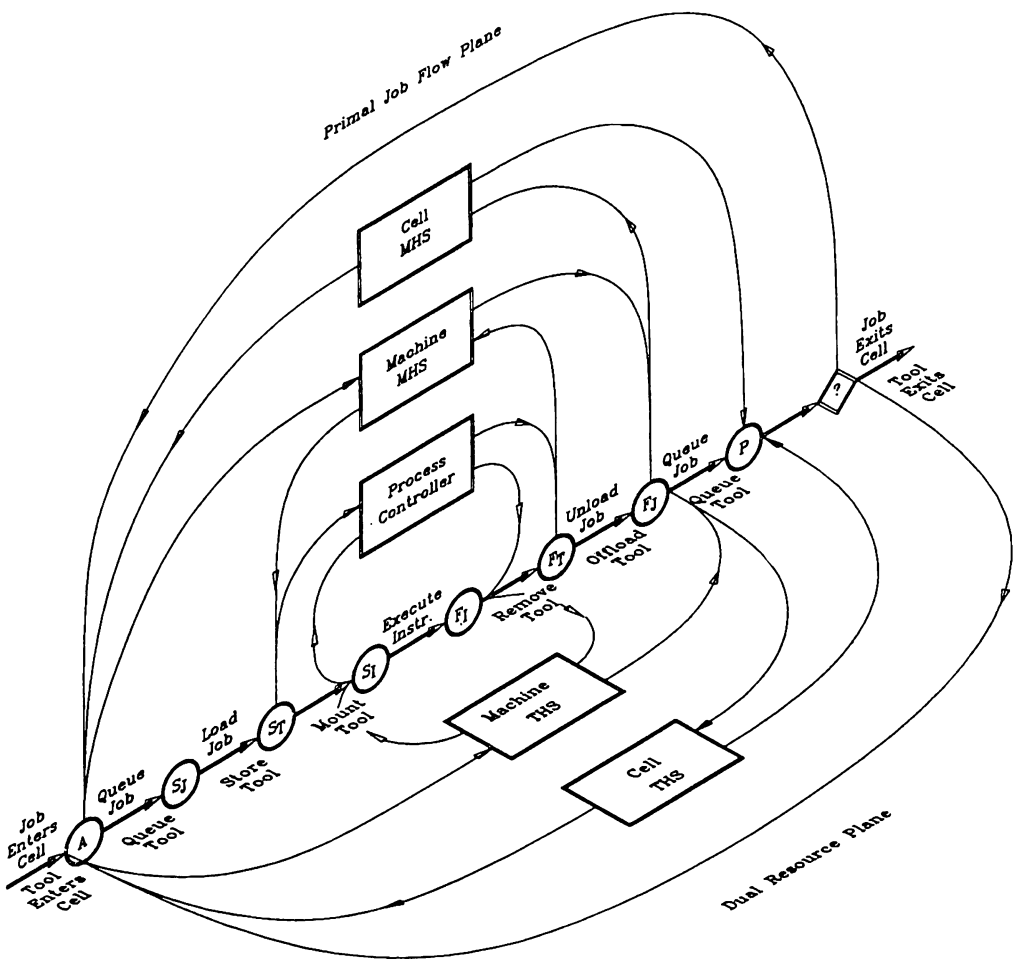

Figure 3. Schematic for the Consideration of All Entity Flows and Controller Interactions

As stated above, nearly every FMS consists of numerous individual controllers whose interaction coordinates the dynamics of the FMS. For the RIA FMS illustrated in Figure 2, there are over sixty individual controllers. To develop a detailed simulation model for the RIA FMS, each controller's logic, as well as its interaction with the other controllers, had to be specified. The conventional simulation tool failed miserably at this task. The current simulation tools are designed to model the flow of entities through a network, not the interactions among controllers. To model entity flow, they presume the logic which will govern the

entity flow will be placed locally at the nodes that define the arcs within the network. In actuality, this flow is being governed by a controller that can view the state information for the subsystem it controls which may include the flow of other entities and other entity types through the same or yet another subsystem. The current simulation tools possess only a limited capability to coordinate the flow of one entity with another.

The detailed simulation model for the RIA FMS was completed using SIMAN and numerous FORTRAN patches. It took over two man-years to develop the simulation code. The final working model was impossible to validate. We believe it is correct. However, with all the FORTRAN patches and nonstandard coding, it is impossible to prove that fact.

Is it really necessary to model an FMS with such detail. It is observed that few reported simulation models consider the detailed operational constraints associated with the flow of supporting resources (i.e. tooling, part kits, and fixtures) and controller interactions. There is a fundamental principle in optimization. Whenever a constraint is added to a decision, the best possible outcome is that the ability to optimize the performance criteria will not be diminished. In most cases, the addition of constraints lowers the achievable performance. The same fact holds true in simulation. When essential operational constraints are ignored in the simulation model, the predicted performance for the system is typically overestimated. In our dealing with the manufacturing sector, we have yet to find an operating FMS that has achieved its simulated performance projections. For the RIA FMS, the proposed vendor of the tool delivery system provided the US Army RIA with a simulated study that projected average process utilizations would exceed 70%. However, after we developed the detailed simulation model, we demonstrated that the average process utilization would be less than 40%. A 70% process utilization could only be achieved by shutting down three of the seven milling machine centers (see Hedlund, Davis and Webster (1990) and Dullum and Davis (1992)).

Before we leave this section, we should note that Figure 3 is yet a simplification of the actual situation. There are many more planes that intersect along the event axis. Currently, we are investigating another FMS where the manufacturer estimates it is losing between 15 and 25% productivity due to information flow alone. Modeling information flow within an FMS using existing simulation tools is again virtually impossible; modeling information flow relies upon our ability to model controller interaction. One final argument for the need to model controller interaction is evidenced by our current inability to design and test the controllers' logic and their interaction during the FMS design process. Too often, the first test of coordination among the controller occurs when the FMS is placed in operation. Nearly every existing FMS that we have modeled possesses one or more modes of operation that can result in system deadlock. In our detailed modeling, we must isolate these modes of deadlock and include them in the model. These modes of deadlock must be treated as we normally would treat the failure of a process, asserting the frequency of occurrence and the time that will be required to return the system to an operational mode.

Thus far, we have focused upon the limitations associated with the current modeling of a FMS. There is also problems arising from the way we perform the statistical analyses for FMSs. Nearly all published statistical analyses associated with simulation are directed toward predicting the steady-state performance of the system. Even terminating simulation studies are geared toward the prediction of the average time-varying performance over a repeating operational cycle. For FMSs, there is no steady state. If these systems are truly being operated in a flexible manner, then the product mix that they will address is constantly changing with time. Therefore, the performance characteristics must change with time. These systems are also significantly effected by their initial state. Recently, we completed yet another detailed simulation study for an FMS dedicated to the production of machined parts. Unlike most FMSs, this FMS was required to produce the same group of parts on a daily cycle. The desire was to define an optimal schedule for the production of the parts which would minimize the makespan for producing the daily part requirements such that the unused production time could be assigned to other production. In this model, *every phenomena was assumed to be deterministic including processing times.* No accounting was made for breakdowns or system deadlocks which certainly existed. In simulating a forty-day production scenario, we found that the makespan required to complete a predetermined daily production schedule varied from 75% to over 200% of the available daily production time. The only source of variability in this model was the state of the tooling residing at each machine at the beginning of each day, which in turn, determined the number of tool replacements that would be required during the day's production. This study clearly demonstrates the need to model FMSs in greater detail. Had our study ignored tooling constraints, the observed fluctuations in makespan would never have been documented.

To analyze FMSs, we need enhanced real-time simulation and statistical analysis capability. Unfortunately only a few papers have been published pertaining to either topic (see Davis, Wang and Hsieh (1991), Tirpak, Deligiannis and Davis (1992), Harmanosky (1990) and McConnell and Medeiros (1992)). Certainly much more research is needed to investigate the manner in which we conduct simulation studies for FMSs.

2.2 Scheduling

Like the current approaches to simulation, the current scheduling approaches are also too closely linked to job entity considerations. Mathematical approaches to scheduling are almost exclusively linked to job-related concerns and typically consider only processing durations, capacity constraints, and precedence relationships. Mathematical programming formulations ignore most material handling constraints and any constraints arising from the flow and utilization of supporting resources and the controller interactions. Given the numerous constraints that are omitted, the feasibility of the schedules derived from the formulated mathematical programming problem cannot be demonstrated. Therefore, any optimality characteristics of the derived solution to the mathematical programming formulation (if it can be generated) are of little merit.

Expert-system based schedulers generate a set of scheduling rules conditioned upon the potential states for the system response. Usually the set of scheduling rules is defined with the input of the individual (expert) currently responsible for scheduling the system. This approach has several weaknesses. First, given the complexity of the considered systems, it is impossible to quantify every possible state that the system can assume. Hence, a comprehensive rule set cannot be achieved. Second, the quality of the derived schedules depends significantly upon the quality of the expert. Furthermore, since the introduction of FMSs is a recent trend, the pool of experts is being reduced.

Current scheduling research is now investigating the use of new techniques, including genetic algorithms and neural nets. Most reported attempts at genetic scheduling have focused upon a "traveling salesman" type formulation. Again they have ignored numerous constraints involved in the operation of an FMS. Neural networks are not a decision-making algorithm, but rather a mechanism through which complex functional relationships can be described and developed. They require training, and the quality of the schedule that they produce is inherently dependent upon the way they are trained.

Perhaps the single conclusion that can be drawn is that much research remains in the development of robust schedulers for flexible automation. There is another observation that merits discussion before we conclude this section. With the increased complexity of emerging manufacturing systems, it is no longer feasible for any single scheduler or controller to coordinate an entire factory. Often there are shared resources within a given subsystem for which the local scheduler/controller must resolve contention.

Future scheduling research must recognize the fact that scheduling will be a distributed function. Furthermore, scheduling will be conducted in real-time and be dependent upon the current state of the subsystem where the scheduling is occurring. The outcome of this scheduling activity must define the control law which can be immediately executed to implement the defined schedule. This control law when implemented will change the state of the system (perhaps in an unpredictable fashion), and therefore, can modify the scheduling problem. Hence, the current scheduling problem to be addressed by a given subsystem must also be updated continuously in real-time. This necessarily implies that scheduling is an on-going activity which must be addressed in real-time. Since problem assessment, scheduling and control are all real-time functions, it necessarily implies that they must be implemented as concurrent activities within a given subsystem. Furthermore, since it is presumed that the scheduling and control functions must be distributed across several subsystems, the activities of one subsystem must be coordinated with the other subsystems.

Finally, production scheduling is but one function in a computer-integrated manufacturing (CIM) environment. There is no way that production scheduling can ever solve the entire CIM function. Instead, the distributed scheduling and control function must interact with the other CIM functions including computer-aided engineering and design, production planning and control, material and capacity requirements planning and process control. In this light, it now is impossible for the researcher and educator in production scheduling to isolate his/her attention to the formulation and solution of the production scheduling problems. The researcher must adopt a much broader perspective.

The issue remains as how simulation relates to scheduling. Simulation is neither a scheduling nor a controlling algorithm. *Simulation is crucial to predicting the behavior of a system operating under a given schedule and associated control strategy.* Given that the scheduling and control functions must be distributed in an FMS, the simulation model must account for this distribution in its modeling. Furthermore, the distributed scheduling and control functions will rely heavily upon a detailed simulation capability to assess the performance of a proposed schedule or control strategy in real-time. To address this role, a totally new perspective in the way we develop simulation models and employ these models in analyzing FMSs is required.

2.3 Control

Although much is now understood pertaining to the control of continuous-state systems, the control of discrete-event systems is in its infancy. There are currently no off-the-shelf algorithms for the control of a large class of discrete-event systems. There is still no uniform consensus on what

functionality should be included. Thus, the technology for a discrete-event system controller awaits development. Furthermore, the conceptual/theoretical basis for distributing the control of discrete-event systems does not exist. Throughout their histories, the literature pertaining to decision making and control have remained nearly distinct; there is little theoretical foundations for their integrated consideration. We now know that this integration must occur across a distributed environment.

3 AN INTEGRATED RESEARCH APPROACH

Over the past several years, in the Manufacturing Systems Laboratory (MSL) at the University of Illinois, several new conceptual frameworks for the modeling, scheduling and control of FMSs have been realized through both basic research and the detailed analysis of operational FMSs. These developments can now be summarized into three fundamental concepts and are discussed below.

3.1 Recursive Object-Oriented Control Hierarchy (ROOCH)

The ROOCH was developed in collaboration with the Government Electronics Group at Motorola, Inc. and is published in Tirpak, Daniel, LaLonde and Davis (1992). The ROOCH introduces a basic building block for modeling a FMS, termed the scheduled object pictured in Figure 4. The scheduled object represents the most fundamental hierarchical element where scheduling and control are implemented. It is assumed that each scheduled object contains one or more primary manufacturing resources or processes P_n (n=1,...,N) which are to be allocated to the

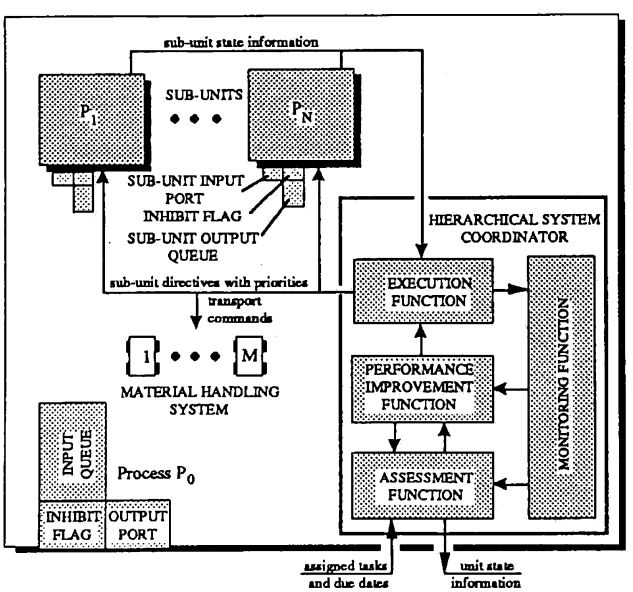

Figure 4. Schematic of the Scheduled Object: The Basic Module for Scheduling and Control

production of jobs residing within the scheduled object. Both jobs and supporting resources (tools, part kits and processing information) enter the scheduled object through its input port and will eventually exit through the exit port which belongs to the scheduled object's supervisor. Jobs and supporting resources are assumed to be under the control of the scheduled object from the moment they enter the input port until they exit through the exit port. Thus, jobs or supporting resources residing in the output queue for the scheduled object are controlled by the supervisor to the scheduled object.

When the scheduled object allocates a primary resource P_n (process) to a given job or supporting resource, the physical control of that entity is passed to the primary resource to which it has been assigned. Note in Figure 4 that the Input Port and the Output Queue of each subordinate primary process belong to the scheduled object whereas the Input Queue and the Output Port for each process belongs to that subordinate process. Therefore, a consistent chain of command for the control of a given entity has been defined as it flows among the various processes within the scheduled object until it eventually departs from the scheduled object through its output port. To regulate the flows into the included input and output queues, inhibit flags have been specified. In general, these flags are controlled by the object to whom the recipient queue belongs. Thus, the Input Inhibit Flag is controlled by the subordinate while the Output Inhibit Flag is controlled by the supervisor.

As stated above, the scheduled object is the most fundamental hierarchical element where scheduling and control occurs. To this end, each scheduled object contains a Hierarchical System Coordinator (HSC) to perform the concurrent functions of scheduling the allocation of the primary processing resources and controlling the flow of the entities to implement the developed schedule. We will discuss the functions for the HSC in greater detail below. To control the flow of entities, it is further assumed that each scheduled object contains one or more material handling systems (MHSs) that are capable of moving entities among the various queues contained within the scheduled object. It is absolutely essential that these MHSs exist since the scheduled object must be able to effect the entity movement if it is to govern the evolution of entity flow as defined in its schedule.

The recursive nature of the ROOCH arises from the fact that any subordinate processing resource can also be a scheduled object. In this manner, we employ the recursive approach to construct the ROOCH with the essential number of hierarchical levels needed to define the FMS that it models. As an example, we depict the ROOCH for the first-stage, physical emulator of an FMS that is currently under construction in the MSL in Figure 5. (We have selected this example rather than an actual system because the emulator is a critical element of the MSL research and education

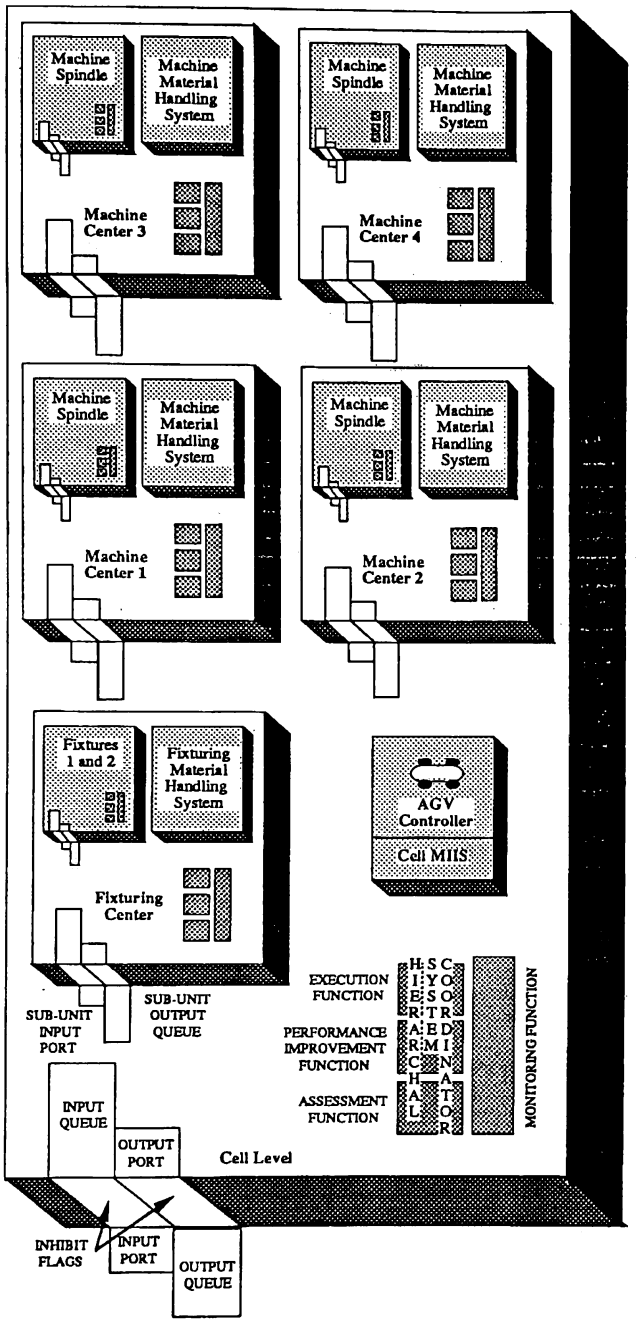

Figure 5. ROOCH for the First-Stage Emulator in the MSL

program and will be discussed throughout the remainder of the paper.) As depicted, the emulated FMS will have four machining (processing) centers, denoted as Machine Center 1 through 4. Each machine center is also a scheduled object containing one primary subordinate processing resource the Machine Spindle. Each Machining Center also will have its own dedicated MHS to move and store the queued jobs, controlled by a dedicated Programmable Logic Controller (PLC). The Machine Center controller is a laptop computer connected to its MHS PLC via a dedicated RS 232 link. In our research and teaching, we will not explicitly model the manufacturing processes in the emulated FMS. To this end, there is no spindle controller included in the emulator. Rather, we will construct a physical display on the Machine Center controller (laptop computer) which closely mimics the typical display that would be provided on actual processing equipment. The Machine Center controller will be tied to a real-time clock and the status of the Machine Center display will be updated as the state of the center changes. We chose to emulate the processing to permit extended experiments to be conducted without the consumption of materials.

Within the emulated FMS (see Figure 5), another subordinate process is the Fixturing Center which also represents a scheduled object. The structure and emulation of the Fixturing Center is very similar to one of the Machining Centers. The Fixturing Center will have its own dedicated MHS consisting of a primary carousel capable of holding sixteen jobs and two smaller carousels for loading and unloading jobs from the AGVs. The movement of these carousels will be controlled by a dedicated PLC. The Fixturing Center will have two fixturing positions which correspond to the spindle at the Machining Center. We will not model explicitly the fixturing process. Rather a dedicated Fixturing Center controller will provide the real-time status information for each fixturer. This same controller will also issue control messages to the dedicated PLC for the carousels via a RS 232 link.

The final subordinate process is the Cell MHS which will be emulated with a HO-scale electric train. The schematic for the HO-scale layout is provided in Figure6. In this layout, there are over forty track segments, indicated by the italic numbers. Both the black circles and squares indicate nodes in the network. The black circles also represent a special class of nodes where switches exist to direct the flow of the AGVs. A detailed Petri net has been developed to govern the transitions between the nodes such that no more than one AGV ever occupies a single track segment at a time. This Petri network both prevents collisions and deadlock by determining which segment(s) of track must be electrified and the proper switch settings (if required) to permit an AGV to move from one node to an adjacent node. A dedicated PLC will be able to individually power each track segment, control the settings of each switch and determine the location of up to seven distinct AGVS operating simultaneously on the track.

The PLC receives directives from the Material Handling controller to move a given AGV from one node to another. The dedicated PLC returns the location of each AGV as it moves past a sensor to the Material Handling Controller. Therefore, the Material Handling Controller knows when a given AGV has reached its desired position, i.e. an issued control command has been implemented. The Material Handling Controller also is responsible for de-

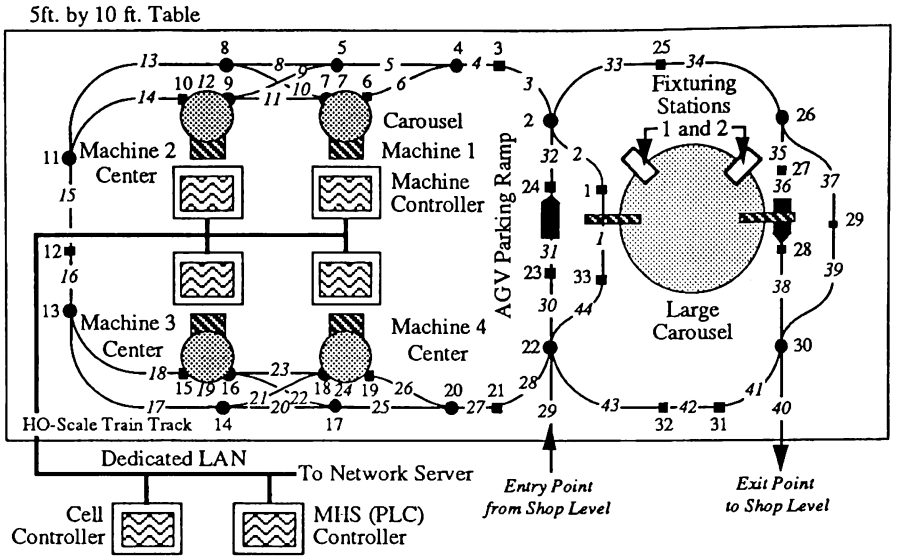

Figure 6. Schematic for the First-Stage Emulator

termining which of the pending material handling transfers will be processed and which AGV will be assigned to complete the requested transfer. This, in turn, defines to which node a given AGV will be moved and results in the appropriate command being issued by the Material Handling Controller to the dedicated PLC via a RS 232 link.

In Figure 6, segments 29 and 40 are special. Segment 29 represents the input port into the cell and the input queue. An AGV is placed on this track segment by the supervisory scheduled object's MHS. Once the AGV arrives on the track segment, the control of the AGV and the job it contains is given to the cell controller for the scheduled object who subsequently tells the cell MHS to move the job to node 28. All incoming jobs to the cell are unloaded at node 28 where one of the smaller carousels is situated. Similarly, all jobs leaving the cell are also loaded upon the departing AGV at node 28. The departing AGV then moves to track segment 40 and stops. Track segment 40 represents the output port and queue. After the departing AGV stops on track segment 40, control of the AGV and the departing job is returned to the supervisory scheduled object for the cell who determines its disposition.

The cell controller is implemented upon yet another computer. It is connected to each Machine Center Controllers, the Fixturing Center Controller and the MHS Controller (and a network server) via an ether network. Various commands and feedback information flow across this network. We will not detail these control messages in this section. The role of the Cell Controller is to orchestrate the flow of the entities of all types (jobs as well as supporting resources) among the subordinate processes. In this manner it is expected to implement the scheduling law established by the cell-level Hierarchical System Coordinator. It must be noted that the Cell Controller is not currently configured as a HSC; it simply does not have the computational power to fulfill this need. An advanced concurrent computing processor will be connected to the network of primary controllers within the cell. The network of controllers (except the PLCs) will operate under UNIX and interface with the concurrent processor representing the HSC. The HSC will provide the priority schemes to be employed by each scheduled object within the cell to allocate its subordinate processes such that the schedule derived by the HSC can be implemented. We chose to develop an interface between the HSC and the cell controller because nearly all existing FMSs will have their own proprietary controllers operated under the software provided by the vendor. To minimize the disruption of the FMS when an advanced HSC is to be implemented, we expect that the HSC must speak to the proprietary controllers within the existing cell using commands that the controllers can interpret.

Before concluding our discussion of the ROOCH, we must note that the ROOCH has already been applied to a broad class of manufacturing environments. At the US Army Rock Island Arsenal, we have described complex FMSs for production of discrete-machined parts. At a major electronics manufacturer, we have described prototypical electronic manufacturing processes using flip-chip, chip-on-board circuitry. For a speaker manufacturer, we have described high volume assembly lines for the production of automobile speakers. The list of attempted applications for the ROOCH is constantly being challenged by new applications when the opportunity arises. To date, the ROOCH has been sufficiently robust to address each application. Furthermore, in defining the ROOCH for a given application, considerable insight is gained in both the flow of all entity types as well as essential controller interactions required to coordinate the flows. It is these controllers interactions which lead us to the second conceptual breakthrough, and these experiences have also been transferred into our design of the physical emulator to maximize its research and educational potential.

3.2 The Hierarchical Object-Oriented Programmable Logic Simulator (HOOPLS)

To address the flow of supporting resources as well as the controller interactions depicted in Figure 3, a new simulation tool, termed the Hierarchical Object-Oriented Programmable Logic Simulator (HOOPLS), has been formulated. A HOOPLS-based model for a FMS will consist of four primary frames. The first frame is the model frame

which contains the specifications for the ROOCH associated with the modeled FMS. The second frame is the control frame which provides for the extensive definition of the control messages that will be issued or received by each controller contained within the ROOCH. It will also define the state transition mechanisms which will occur upon the receipt of a control message and the subsequent messages that will be issued. Specification of these control messages and state transition mechanisms is a challenging, but essential task. As stated above, it is these controller interactions which govern the physical movement of the various entity types through the modeled FMS. Our research with the ROOCH has demonstrated, however, that there is only a limited number of primitives that need to be recognized by each control node in the ROOCH.

Given that HOOPLS explicitly models the interaction among the controllers within the ROOCH and considers the flow of all entities to be a consequence of these interactions, HOOPLS has completely abandoned the use of an event calendar. Instead HOOPLS employs a control message calendar which stores the control messages that will be passed among the controllers chronologically-ordered based upon its delivery time. Each control message will designate the controller that issued the message, the recipient controller for the message, the actual message, and the scheduled time for its delivery. The simulation executive object which manages the control message calendar will be responsible for delivering the control messages to the recipient controller at the appropriate simulated time. We note that this simulation executive object physically mimics to the network which links the controllers in an actual FMS.

The need to consider controller interactions within the simulation is further accentuated by our experience in the development of the HOOPLS-based model for the physical emulator described above. When we modeled this emulator using the conventional simulation approach with current simulation tools, approximately 20 events were depicted for each job moving through the emulator. Using the HOOPLS-based model with its detailed control frame, over 160 controller interactions were modeled for each job flowing through the emulator. As presently configured, the current emulator does not address the flow of supporting resources (the physical emulation of a tool handling system has been designed and will be constructed shortly). If these flows were considered, the number of controller interactions is expected to exceed one thousand per job. Consider a conventional control system and reduce its sampling rate by one or two orders of magnitude. Invariably, the ability to control the system will be compromised. If this fact holds for the simpler continuous state system, we must expect that it will also hold for the more complex discrete-event systems.

The exercise for developing the detailed control frame for the emulator also verified the controller interactions and the absence of deadlock *before* the emulator was constructed. In fact, the control objects used in the simulation model will be ported directly to the individual computers within the emulator to provide the logic for each controller. For most existing FMSs, it is nearly impossible for the owner/operator to provide a complete specification for the set of controller interactions. Such a set was never generated even during the planning process. Rather the FMS was developed by bringing various vendors' equipment together and then developing the interfaces. Often the first test of the interoperability for the resulting FMS occurs when the FMS is brought into operation. It is no wonder why most existing FMSs contain one or more potential modes for system deadlock. We believe that HOOPLS will alleviate this situation by providing the capability to specify the detailed interactions for the contained controllers and demonstrate their interoperability characteristics before the FMS is constructed. These specifications can then be provided to vendors as absolute design constraints whose satisfaction must be demonstrated.

The third simulation frame in the HOOPLS-based model is the processing plan frame. The processing plan details not only which manufacturing processes will be required and in which order, but also details which supporting resources will be required to complete a processing task. HOOPLS recognizes an important fact in its inclusion of a separate processing plan frame. With respect to the FMS, it is not concerned with which parts are to be produced so long as the essential processing resources exist within the FMS and the essential supporting resources can be delivered to and employed by the processing resources. Similarly, the processing plans are not concerned with the organization of the FMS; they only specify what resources are needed and what processing instructions will be implemented. Furthermore, flexibility is not only achieved by allowing an FMS to produce a wide variety of parts, but also allowing a wide variety of processing options for the FMS to consider in producing the part.

The fourth simulation frame is the experimental frame which specifies the experimental parameters governing the simulation study. The experimental frame is now an accepted element of many commercial simulation tools and will not be discussed in detail here.

The complete specifications for the HOOPLS language are currently under development. Before finalizing these specifications, we hope to model a wide variety of flexible manufacturing environments to expand the scope of applicability for the proposed language. It should also be noted that ROOCH-based simulation models are also being developed for other complex discrete-event systems. One application under development is an advanced simulator for an intelligent vehicular highway system. Finally, HOOPLS is being developed explicitly to support the real-time, discrete-event simulation, a technology which is

critical toward the implementation of the integrated real-time scheduling and control algorithm within each scheduled object.

3.3 The Hierarchical System Coordinator (HSC)

To implement the integrated, real-time scheduling and control within a scheduled object, the concept of the HSC (see Figure 7) is introduced and discussed in Davis, Jones and Saleh (1992) and Davis (1992). The planning horizons to be considered by the HSC are very short, typically a day or less, and we do not expect steady-state operation. That is, the HSC is constantly addressing a transient response. This situation is further exacerbated in the flexible manufacturing environment where the products to be produced are often modified.

Given the transient nature of the discrete-event system response, we expect that the scheduling decision to be addressed will be constantly modified and dependent upon the current state of the system. Since the scheduling decision is constantly changing, the need to optimize is an on-going process. It is also noted that these systems are typically stochastic in nature and that multiple performance criteria are to be considered. Furthermore, the potential for compromise among these criteria is again dependent upon the state of the system.

Making a decision in the real-time environment does not end with the specification of a schedule. In the real-time environment, we must also specify the control law through which the schedule can be realized. Furthermore, a control law must always be available to specify the next control action. Postponing the issuing of a control action to a subordinate is an implicit control action in itself. In the FMS, this postponement can result in a process becoming idle until it is directed to do its next task. Note, however, with each control action there is a potential to change system state, and, hence, the current scheduling decision. In this regard, decision making and control become intrinsically linked and must be addressed concurrently.

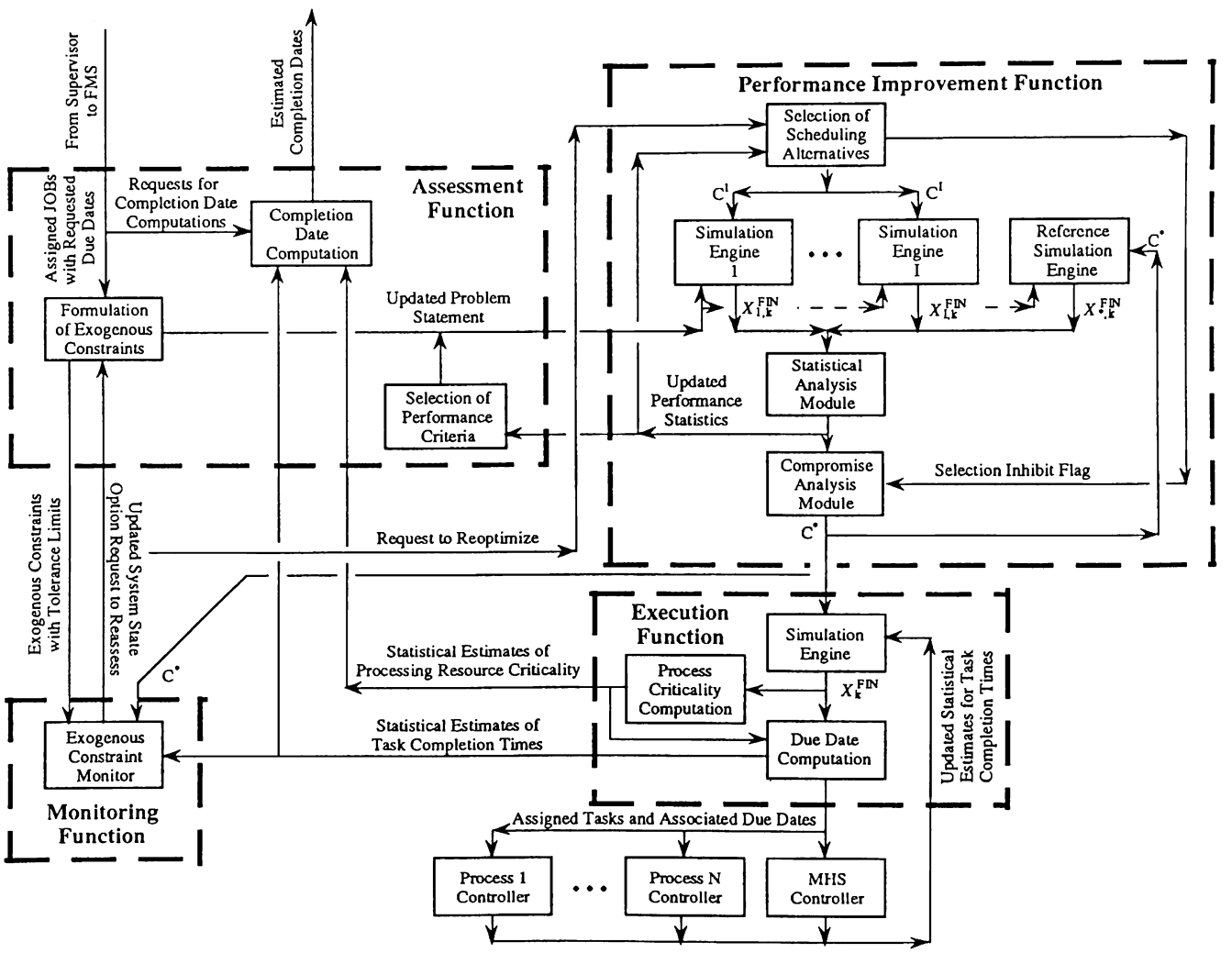

Figure 7. Schematic for the Concurrent Computing Processes Comprising the HSC

The latest configuration for the HSC is given in Figure 7. The HSC is designed to be implemented on a parallel (concurrent) processor as a series of asynchronous computing processes. That is, the communication among the computing processes is unidirectional only; no hand-shaking is required. These computing processes can be grouped into four basic functions. The Assessment Function (AF) is responsible for continuously updating the current decision to be addressed by HSC. It receives as input the updated state of the controlled subsystems, i.e. the scheduled objects. From the supervisor to the scheduled object, it also receives assigned tasks and negotiates due dates using the Completion Date Computation module. Note that the Completion Date Computation is an on-going process in real-time, and these projected dates provide the primary feedback information to the supervisor. The imposed due dates are used to develop the current set of exogenous constraints for the decision making. These constraints and the tolerance limits to which they must be satisfied (the controlled subsystem is stochastic) are submitted to the Performance Improvement and the Monitoring Functions. The AF also receives statistical information from the Performance Improvement Function pertaining to the success that it has experienced in achieving various performance goals or criteria. Based upon this information, the Selection of Performance Criteria module determines the tradeoffs that should be considered among the criteria in the stochastic compromise analysis element of the multi-criteria scheduling decision. This information is again passed to the Performance Improvement Function.

The Performance Improvement Function (PIF) is always seeking a better scheduling control law C* for implementation. To accomplish this task, several (I) alternative control laws are compared to the current C* using real-time, discrete-event simulation. This comparison requires that real-time statistical analysis and compromise analysis be performed as described in Davis, Wang and Hsieh (1991) and Tirpak, Deligiannis and Davis (1992). One important facet of this analysis is the determination of whether the system is currently in a relative steady state or in a highly transient regime. It is important to note that statistic analyses performed in a transient regime are likely to have greater uncertainty which makes comparison of scheduling alternatives more difficult. Furthermore, modifying the implemented control law during a transient regime may introduce additional uncertainty to the system. Antonacci (1992) has experimented with adaptive statistical analysis procedures to first assess the degree of transient response and second maximize the confidence in the estimates that can be achieved in all modes of system response.

The Selection of Scheduling Alternatives is defined to support all for generating production schedules. That is, we are not advocating any specific approach toward the generation of a schedule. For example, if one desires to use mathematical programing, then one might consider several distinct mathematical programs where the basic constraints for the decision are the same, but the considered objective functions differ. If one desires to develop an expert-based scheduling approach then several potential experts may be employed. It is also possible that the mode for the generation of scheduling alternatives could differ for each alternative. For example, some alternatives could be generated through mathematical programming while others are generated by expert systems. After the alternatives are generated, each alternative is subjected to real-time, discrete-event simulation which projects the near-term performance of the schedule based upon the current state of the system. Note that the current schedule defined via the control law C* is always simulated and considered in this comparison.

The outputted simulation trials are then passed to the Statistical Analysis Module which performs the real-time statistical analysis. The number of considered simulated trials for each scheduling alternative depends upon whether the system is currently operating at a nearly-steady state condition or undergoing a significant transient behavior. Specifically, the mean and variance for each considered performance criterion and the pairwise correlation among the considered performance criteria are computed under each scheduling alternative in real-time as discussed in Davis, Wang and Hsieh (1991). The statistical results are graphically depicted using individual computer screens as shown in Tirpak el al. (1992).

The results of the statistical analysis module, as well as the simulation trials upon which they are based, are then passed to the Stochastic Compromise Analysis Module. While the Statistical Analysis Module limits its consideration to the computation of statistics within a given scheduling alternative, the Stochastic Compromise Analysis Module performs its comparative statistical analysis across the scheduling alternatives. Specifically, the Statistical Compromise Analysis modules compares the statistics for each scheduling alternative against the current statistics for the control law C* which is being implemented. Whenever an alternative control law is demonstrated to outperform the current control law C* that is being implemented, then the current C* is replaced by the new alternative.

When a new control law C* is selected, it is then passed to the Execution Function for implementation. The EF again performs yet another detailed, real-time simulation using the detailed task completion times received from the AF of its subordinate scheduled resources. Based upon this information, it generates statistical estimates for the completion time for the tasks it will assign to its subordinates resources or processes in the future. It also uses these detailed, real-time simulations to project the future criticality for each scheduled resource as discussed above. These criticality data are passed to the AF and the PIF to be used within their analyses.

The Monitoring Function (MF) is involved in maintaining the feasibility of the scheduled response when disruptions occur. It also coordinates the concurrent efforts of the AF, the PIF and the EF. These processes are rather complicated, but they are described in detail in Davis (1992). The HSC has not yet been fully implemented. However, major elements of the PIF have been tested, particularly the comparison of the scheduling alternatives using real-time, discrete-event simulation. The proposed HSC will be fully implemented for the scheduling and control of the emulator as a first step toward the transferring of this technology to a real FMS. Working with an industrial collaborator, we hope to begin implementation of the HSC on an actual FMS within a three-year horizon.

4 THE FUTURE RESEARCH AND EDUCATION AGENDA

The FMS emulator discussed in this paper is a critical component of our research and education program in the MSL. The construction of the emulator represents a major advancement in the simulation capability. Whereas current simulation tools employ an animation of the entity flow to illustrate the operation of the model, the HOOPLS-based control objects have been ported to the various controllers and actually coordinate the physical operation of the emulator. That is, we have demonstrated that a detailed simulation can actually control the system that it models. We are using this fact to define an ensemble of new courses in the manufacturing engineering program at the University of Illinois. The first course will teach engineering undergraduates how to program a FMS. Specifically, the students will be introduced to the ROOCH and will be expected to generate their own computer code for every controller contained within the emulator. The second course will be an inductory simulation course addressing flexible manufacturing systems, where the students will be introduced to conventional simulation tools. They will be expected to collect data from the operating emulator for inclusion in their models. They will also be expected to fully validate their simulation model against the operation of the emulator. In short, the emulator will provide the closed-loop, real-world experience that is often lacking in present simulation courses. The third course is an advanced simulation course which will require the students to generate their own HOOPLS-based simulation model for the emulator in C++. The students will also be required to apply the ROOCH architecture to several other FMSs. Finally, the students will be introduced to technology of real-time, discrete-event simulation and the associated real-time statistical and compromise analyses.

There are four primary research thrusts being addressed in the MSL. The first is the development of a complete set of specifications for an eventual commercial implementation of HOOPLS. To this end, several FMSs will be modeled to gain the essential experience needed to provide a robust simulation tool which is not only capable of analyzing an existing FMS, but can also serve as a tool for the computer-aided design of FMSs. The second research area is the implementation of the HSC. This effort has been delayed by both the availability of a concurrent computing process for the implementation of the HSC and the availability of a test site. Several manufacturing entities have already offered their FMSs for implementation of the HSC. However, the HSC will require a major effort to implement with several advances in technologies including real-time simulation, statistical and compromise analyses. Therefore, it is felt that the emulator will provide the best site for first implementation. In addition to the direct implementation of the HSC, there have been new adaptive search algorithms developed for the generation of scheduling alternatives, and the efficacy of these algorithms will be tested within the implementation.

The third research is the development of improved algorithms for the real-time, decentralized scheduling and control of discrete-event systems. Already considerable development has occurred and is reported in Davis and Jones (1992). Like the ROOCH, it is nearly impossible to theoretically assert the correctness of a proposed algorithm. Rather we must view these algorithms as they expand the existing theories in decentralized decision making and decentralized control. This research must also develop the interfaces between the basic principles of decision making and control which are usually viewed as distinct topics. To test these theories, the plans for the second- and third-stage emulators have already been formulated to provide a multi-level flexible manufacturing emulator consisting of two emulated cells and a shop-level emulator that coordinates their production.

The fourth research area will advance the formulation of the other CIM functions and their interfaces. Already the HSC has been applied to develop new production planning and control hierarchies (see Davis and Thompson (1991, 1993), Thompson, Jewell and Davis (1992), Thompson, Watanabe and Davis (1993)). Currently we are investigating the material requirements and capacity requirements planning functions. We are also using the HSC to develop even more advanced process controllers. In this latter development, we are exploring the interfaces between the control of discrete-event and continuous-state systems.

REFERENCES

Antonacci, L. A. 1992. Experimentation in Stochastic Compromise Analysis for Real-Time Multi-Criteria Decision Making. An unpublished Masters Thesis, W. J. Davis (advisor), Department of General Engineering, University of Illinois, Urbana, IL.

Davis, W. J. 1992. A Concurrent Computing Algorithm for Real-time Decision Making. *Proc. of the ORSA Computer Science and Operations Research: New Developments in their Interfaces Conference*, eds. O. Balci, R. Sharda and S. Zenios, 247-266, Pergamon Press, London.

Davis, W. J. and A. T. Jones. 1992. The Application of a Generic Controller in a Multi-Level Production Scheduling and Control Hierarchy. National Institute of Standards and Technology Report NISTIR 4835, Gaithersburg, MD.

Davis, W. J., A. T. Jones and A. Saleh. 1992. A Generic Architecture for Intelligent Control Systems, *Computer Integrated Manufacturing Systems*, 5(2), 105-113.

Davis, W. J. and S. D. Thompson. 1991. Decision-making and Control Schema for Production Planning in CIM Systems, *Proc. of the Joint US/German Conf. on New Directions in Operations Research*, eds. G. Fendel, Th. Gulledge and A. Jones, 101-125, Springer-Verlag, Berlin.

Davis, W. J. and S. D. Thompson. 1993. A Production Planning and Control Hierarchy using a Generic Controller. To appear in *IIE Transactions*.

Davis, W.J., H. Wang and C. Hsieh. 1991. Experimental Studies in Real-Time, Monte Carlo Simulation, *IEEE Systems, Man and Cybernetics*, 21(4), 802-814.

Dullum, L. M. and W. J. Davis. 1992. Expanded Simulation Studies to Evaluate Tool Delivery Systems in a FMC. *Proc. of the 1992 Winter Simulation Conference*, eds. J.J. Swain, D. Goldsman, R. C. Crain and J. R. Wilson, 978-986, The Society for Computer Simulation, San Diego, CA.

Flanders, S. W. 1991. Optimal Sequencing of Part Production in a Flexible Manufacturing System Using Genetic Algorithms. An unpublished Masters Thesis, W. J. Davis (advisor), Department of General Engineering, University of Illinois, Urbana, IL.

Harmanosky, C. M. 1990. Implementation Issues Using Simulation for Real-time Scheduling, Control, and Monitoring. *Proc. of the 1990 Winter Simulation Conference*, eds. O. Balci, R. P. Sadowski, and R. E. Nance, pp. 595-598, The Society for Computer Simulation, San Diego, CA.

Hedlund, E., W. Davis and P. Webster. 1990. Using Computer Simulation to Compare Tool Delivery Systems in an FMC. *Proc. of 1990 Winter Simulation Conf.*, eds. O. Balci, R. P. Sadowski, and R. E. Nance, 641-645, The Society for Computer Simulation, San Diego, CA.

McConnell, P. G. and D. J. Medeiros. 1992. Real-Time Simulation for Decision Support in Continuous Flow Manufacturing Systems. *Proc. of the 1992 Winter Simulation Conference*, eds. J.J. Swain, D. Goldsman, R. C. Crain and J. R. Wilson, 936-944, The Society for Computer Simulation, San Diego, CA.

Thompson, S. D., J. A. Jewell and W.J. Davis. 1992. Issues in Specifying Planning Horizons for Production Planning in CIM Environments. *Proc. of the Joint US/German Conf. on New Directions in Operations Research*, eds. G. Fendel, Th. Gulledge and A. Jones, 270-290, Springer-Verlag, Berlin, Germany.

Thompson, S. D., D. T. Watanabe and W. J. Davis. 1993. A Comparative Study of Aggregate Production Planning Strategies under Conditions of Uncertainty and Cyclic Product Demands. To appear in *International Journal of Production Research*.

Tirpak, T. M., S. M. Daniel, J. D. LaLonde and W. J. Davis. 1992. A Fractal Architecture for Modeling and Controlling Flexible Manufacturing Systems. *IEEE Trans. on Systems, Man and Cybernetics*, 22(5), 564-567.

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

AUTHORS BIOGRAPHIES

WAYNE J. DAVIS is a Professor of General Engineering at the University of Illinois. His research interests include computer-integrated manufacturing; production planning and scheduling; and the integrated, real-time simulation, scheduling and control for discrete-event systems. He collaborates with the Automated Manufacturing Research Facility at the National Institute of Standards and Technology at Gaithersburg, MD and the Electronics Manufacturing Productivity Facility at Indianapolis. Dr. Davis founded the Manufacturing Systems Laboratory which has considered flexible manufacturing systems for industrial manufacturers including Motorola, Inc. and Caterpillar, Inc. and defense manufacturers including the US Army Rock Island Arsenal and the Naval Air Warfare Center/Indianapolis.

DUANE SETTERDAHL, JOSEPH MACRO, VICTOR IZOKAITIS and BRAD BAUMAN are all graduate student researchers in the Manufacturing Systems Laboratory at the University of Illinois. Joseph Macro is a recipient of a US Departement of Energy Predoctoral Felloship in Integrated Manufacturing while Victor Izokaitis is a recipient of the Motorola Scientific Advisory Board Fellowship. Both are currently pursuing their doctorate degrees in Industrial Engineering. Duane Setterdahl and Bradley Bauman are currenlty pursuing their masters degree in General Engineering.