Proceedings of the 1994 Winter Simulation Conference
ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila

DISCRETE EVENT SIMULATION FOR SHOP FLOOR CONTROL

Jeffrey S. Smith
Richard A. Wysk

Industrial Engineering
Department
Texas A&M University
College Station, TX 77843

David T. Sturrock

Systems Modeling
Corporation
504 Beaver Ave.
Sewickley, PA 15143

Sanjay E. Ramaswamy
Glen D. Smith
Sanjay B. Joshi

Industrial and Management Systems
Engineering Department
Penn State University
University Park, PA 16892

ABSTRACT

This paper describes an application of discrete event simulation for shop floor control for a flexible manufacturing system. In this application, the simulation is used not only as an analysis and evaluation tool, but also as a "task generator" for the specification of shop floor control tasks. Using this approach, the effort applied to the development of the simulation is not duplicated in the development of the control system. Instead, the same control logic is used for the control system as was used for the simulation. Additionally, since the simulation implements the control, it provides very high fidelity performance predictions. Implementation experience in two flexible manufacturing laboratories is described. These implementations use a special feature of the Arena/SIMAN simulation language which allows Arena/SIMAN to interact directly with the shop floor control system through an interprocess communication mechanism. This feature is described in detail in this paper.

1. INTRODUCTION

Traditionally, discrete event simulation has been used as an analysis and planning tool prior to the actual implementation of manufacturing systems. This is especially true in the case of flexible manufacturing systems (FMS) which are difficult to model using purely analytical techniques. In general, the first step in the development of a new FMS is to identify a potential system design. Once a design has been identified, a simulation is developed to analyze this design according to a set of specified performance metrics. The system design is then "tweaked" in order to improve the expected performance, and, in some cases, additional designs are developed and evaluated. The role of the simulation in this activity is to predict the system performance for a given design. Once a promising design has been finalized, the simulation is set aside and the control system is created.

In this development cycle the same system control logic is essentially developed twice: once for use in the simulation, and then again for the control system. Similarly, when system modifications are required, the simulation is typically used to test potential modifications first. Once the impact of the modifications has been verified, the control system is correspondingly modified. This duplication of effort represents a significant cost in the development and maintenance of a flexible manufacturing system. A more attractive alternative to this cycle is to use the same logic for the simulation and the control, thereby reducing or eliminating this duplication of effort.

Under this paradigm, once the system design has been finalized, the simulation that was used for evaluation is then used as the basis for the control system. A novel approach towards this goal has been developed as part of the RapidCIM project. RapidCIM is a three year joint project between Texas A&M and Penn State Universities and Systems Modeling Corporation (Wysk *et al.*, 1992). The overall objective of the project is to reduce the time required to develop fully functional shop floor control systems for flexible, discrete parts manufacturing systems. This paper describes the simulation-based control system that has been developed for the Texas A&M Computer Aided Manufacturing (TAMCAM) lab and Penn State CIM lab in conjunction with this project.

2. RAPIDCIM APPROACH

In an FMS, the shop floor control system (SFCS) is responsible for processing parts through the system. This includes selecting part processing routes and material handling operations, scheduling operations based on the current system state and current production requirements and then implementing these plans in the physical system.

In the proposed approach, the shop floor controller functions are partitioned so as to separate the *execution* functions from the *decision making* (planning and scheduling) functions. According to this partitioning, execution is responsible for interacting with the shop floor equipment in order to implement the physical tasks, and decision making is responsible for specifying which tasks will be executed and in which sequence in order to meet the production requirements. The rationale for this partitioning is that the execution functions depend only on the physical equipment configuration, while the decision making functions also depend on the specific part types being produced and the production mix. Explicit separation of these functions allows modular development of the execution and decision making functions.

The execution functions have been modeled using a new formalism called a message-based part state graph (MPSG). An MPSG for a controller describes the *processing protocol* for that controller. In a distributed control system, control is exercised by passing messages and signals between controllers and by performing specific controller tasks. The processing protocol describes the controller in terms of the messages it receives and sends and the tasks that it performs in response to messages. Given the MPSG and the device-specific interaction routines for a piece of equipment, the execution portions of the shop floor controller can be automatically generated using previously developed tools (Smith and Joshi 1993).

Figure 1 illustrates the structure of the RapidCIM shop floor controller. In this structure, an Arena/SIMAN (Drevna and Kasales, 1994) simulation serves as the decision maker (or *Task Generator*) and MPSG-based controllers perform the execution functions. The Task Generator and execution module communicate through the *task initiation queue* (TIQ) and the *task completion queue* (TCQ). The Arena/SIMAN simulation uses the TIQ to instruct the execution module to perform a specific task. The MPSG-based execution module interprets the request as an incoming message and performs the requested task. Upon completion of the task, the execu'ion module uses the TCQ to inform the decision maker that a specified task has been completed. These queues facilitate the explicit separation of the decision maker from the execution module. The execution module simply executes tasks it reads from the TIQ without regard for how these tasks are selected. Similarly, once a task sequence has been determined, the decision maker simply writes tasks to the TIQ. When the decision maker receives the task completion message from the TCQ, it knows that the task has been completed successfully, but has no knowledge of how the task was implemented.

The separation of the decision maker and the execution module makes the system truly "plug and play." Assuming the decision maker understands the physical constraints imposed on the task sequences, any decision maker can be "plugged in" to the execution module according to the current production requirements. This allows use of a decision maker most appropriate to the specific application such as an expert system, custom optimization algorithm, simulation, or other appropriate tool.

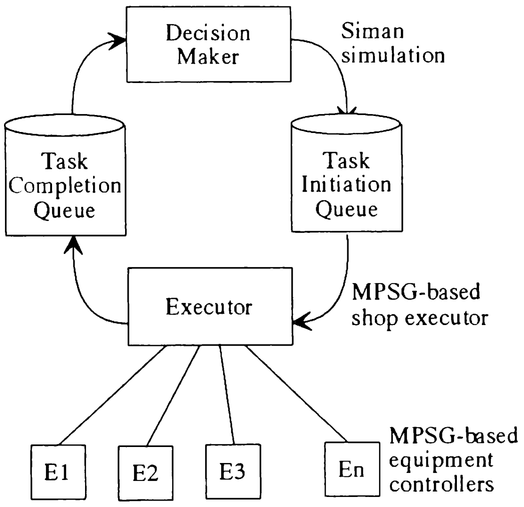

Figure 1. RapidCIM shop floor controller structure.

The separation between the decision maker and the execution module, also makes it possible to use the decision maker in an off-line "play and plug" mode. Off-line, it can be used to "play" with different production alternatives and/or refine the production heuristics to provide improved operation. Likewise, this play mode can be used to determine the best response to exceptions such as equipment failures. This new knowledge can then be "plugged" back into on-line operations.

It is frequently desired to emulate certain equipment. Emulation can be used to permit limited operation during installation, equipment failures, and while testing new equipment/approaches. More important, emulation allows the same control system to be used in both real-time control as well as the off-line mode discussed above.

Emulation can be accomplished in two ways. The first approach is to let the execution module simply respond with a task completion message after a fixed time interval. This approach does not require any extra logic in the decision maker, and, in fact, the decision maker is unaware that tasks are being emulated.

The second approach is to let the decision maker implement emulation logic in place of actually executing a task. For example, Arena/SIMAN can incorporate any logic necessary (from a simple delay to complex guided transporter interactions) to emulate equipment. This logic would be developed during initial system design/analysis and retained for later reuse during testing and the emulation discussed above. Using this form of emulation, the system can be emulated on a single computer without the use of the network.

3. ARENA/SIMAN LANGUAGE MODIFICATIONS

As part of the RapidCIM project, Arena/SIMAN has been enhanced with additional constructs to make it possible to implement the physical control of equipment. The following discussion illustrates one approach implemented for real-time control. It is likely that this approach will be improved as more is learned from our research. Typically, equipment processing functions are modeled as simple time delays in which the delay time closely approximates the expected processing time on the physical equipment (in simulation time). These delays can be modeled with the current SIMAN DELAY and TRANSPORT blocks in conjunction with the STATION and SEQUENCES elements.

However, these constructs are not sufficient for on-line physical control. Under physical control mode where the simulation is acting as the task generator, when an entity reaches a point in the simulation involving a processing step, the simulation must tell the execution module to perform a task and wait for it to be completed before the entity can continue through the simulation. Consequently, several modifications have been made to Arena/SIMAN language to facilitate this type of control. The modifications include the addition of the TASK element and modifications to the DELAY, ROUTE,. MOVE, and TRANSPORT blocks. These language constructs are described in the following paragraphs.

The existence of two environment variables specify the major modes that Arena/SIMAN operates within. If the environment variable SM_RealTime exists, the simulation will run in real-time as measured by the computer's internal clock (referred to as *wall time*). Real-time or control mode is mandatory when using Arena/SIMAN to control a physical system. By running the simulation in fast mode, it can be used in the more traditional role of a performance predictor. If the environment variable SM_Execute exists and is set, Arena/SIMAN will attempt to execute tasks by writing them to the task initiation queue and waiting for the repsonse from the task completion queue. Otherwise, all tasks are emulated using the standard Arena/SIMAN language constructs.

The TASKS element specifies physical tasks which are to be written to the task initiation queue (TIQ) to be performed by the execution module. The syntax for the TASKS element it as follows:

TASKS: Number, TaskID, ExecExpr, Format, Parameter, ... : repeats;

where *Number* is an optional task number, *TaskID* is the task identifier (a string), *ExecExper* is an expression which, if it evaluates to non-zero, specifies that the task should be executed. If *ExecExper* evaluates to zero, the task is emulated. *Format* is the format string (similar to C's printf format string) used for writing messages to the TIQ. *Parameters* represent parameters specified in the format string. The following is an example of the TASKS element.

TASKS: pick, LocalExecOn, "pick %1.0f TGID=%1.0f loc=%f mhe=%s",partNum, IDENT, CurrentLoc, STR(STATIONS,m);

If LocalExecOn evaluates true, the task associated with this element is written to the TIQ in response to a DELAY, ROUTE, MOVE or TRANSPORT block (described below). The format of the task written to the TIQ is as follows:

pick 5 TGID=17 loc=23.25000 mhe=puma

The execution module interprets this task as a request for the specified material handling entity to pick the specified part from the specified location.

A new field has also been added to the DELAY, MOVE, TRANSPORT, and ROUTE blocks to permit real-time execution of a task. The new syntax of these blocks is as follows:

DELAY: Time, Storage, TaskID;

ROUTE: Time, Destination, TaskID;

MOVE : Transporter, Destination, Velocity, TaskID;

TRANSPORT: Transporter, Destination, Velocity, GTR Dest, TaskID;

Where the parameter *TaskID* is a name or an expression which evaluates to a task identifier as defined by a TASK element.

When a simulation entity reaches one of these blocks, Arena/SIMAN first looks at the global execution mode (based on SM_Execute). If it is off, the task portion of the block is ignored. If global execution mode is on, the execution expression (*ExecExpr*) for the task is evaluated to determine whether or not the task should be executed..

If *ExecExpr* for the task is false, the equipment is being emulated and the task is ignored. If *ExecExpr* is true the task will be executed in an operation parallel to the logic of the block being processed. If the block finishes first, its operation will be suspended until the task returns. If the task finishes first, the block operation will be aborted and treated as though it had finished early.

To execute a task, a message is added to the task initiation queue. The content of the message will be determined by the format and parameters specified by the task element. The entity executing this task will take no further action until it is identified in a "task complete" message in the task completion queue. Although this entity will be suspended, Arena/SIMAN will continue to process other entities and execute any associated tasks.

The final modification required for use in the RapidCIM environment is a mechanism for implementing the task initiation and task completion queues. To facilitate these message queues, Arena/SIMAN provides the following stub functions.

srIPC_Initialize() - This function initializes the interprocess communications required for the queues.

srIPC_Shutdown() - This function closes the interprocess communications when the simulation terminates.

srIPC_WriteMessage(msg) - Writes the specified text message to the task initiation queue.

srIPC_ReadMessage(msg) - Reads the first message waiting in the task completion queue.

The user creates these functions for the specific operating system and inter process communication mechanism under which the system is being run. Within the laboratories described in the following section, these functions have been developed under OS/2 and DEC ULTRIX.

4. LABORATORY DESCRIPTIONS

4.1 TAMCAM LAB

The RapidCIM controller has been implemented at the Texas A&M Computer Aided Manufacturing (TAMCAM) lab. This lab includes a full-scale flexible manufacturing system (shown in Figure 2) consisting of three CNC machine tools, several industrial robots, a vertical automatic warehouse, and a "smart" conveyor. Table 1 describes the specific equipment in the TAMCAM lab. In this lab, the conveyor transports parts from station to station on specialized pallets. When a pallet arrives at a load/unload station, parts are automatically unloaded by an industrial robot. The robot then loads the part on one of the machine tools where the part is processed. While the part is being processed on one of the machine tools, the pallet can either wait at the load/unload station for the part to be completed, or it can be used to transport other parts within the system.

The TAMCAM lab uses a variety of computers and operating systems to implement the shop floor control system. An equipment level controller running on a DOS-based personal computer front-ends each individual piece of equipment. Each of these computers is connected via Ethernet to the centralized shop level controller running on an OS/2-based personal computer. The shop level controller consists of the Arena/SIMAN simulation model, the shop execution module, and the task initiation and completion queues. A "Design Workstation" is used to design and modify part produced in the lab and to create and store the numerical control (NC) programs required to process the parts. The Design Workstation is based on AutoCAD and runs on an IBM RS/6000 running AIX 3.2.

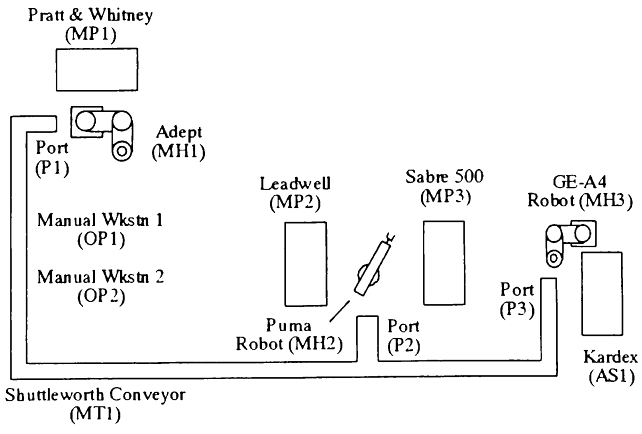

Figure 2. TAMCAM lab FMS.

Table 1. TAMCAM equipment.

Item	Description
1.	Cincinnati Milacron Sabre 500 – CNC Machining Center
2	Pratt & Whitney Drill-o-Mate* CNC Machining Center
3	Leadwell LTC CNC Turning Center
4	GE A4 SCARA Robot
5	Adept SCARA Robot System*
6	Kardex Vertical Carousel AS/RS
7	Material Handling System – Flexible Conveyor System*†
8	Puma 760 Robot
9	Computer Control System and Software

* Donated by DEC.
† Donated by Shuttleworth

The communications network software which connects these systems has been developed specifically for this project and is based on TCP/IP. This communications system provides tremendous flexibility in the operation of the system. Processes can be moved between computing platforms with only simple recompilation. For example, to move the shop execution module to the RS/6000 to free up resources for the simulation, the shop execution code can be recompiled under AIX without modification. The only other required modification is in the *network route map* files. These files are ASCII text files (roughly 10 lines long each) which the controller uses to route messages. These files can be modified for each controller without mandating code recompilation.

4.2 PENN STATE CIMLAB

This Lab hosts a flexible manufacturing shop consisting of three CNC machines, several robots, an automatic storage and retrieval system (AS/RS), and a Cartrac material transport system (Figure 3). Table 2 details the equipment currently operational in the lab. Parts are stored in the AS/RS, from where they can be retrieved by the IBM 7535 robot and placed on one of the carts of the Cartrac system for transport to the processing workstations. At each workstation another robot is provided for material transport within the workstation area. Completed parts are routed back to the AS/RS using the same transport equipment. The shop floor control system is implemented on DOS-based personal computers at the equipment level. Each of these computers are connected via ethernet to a centralized shop controller that can variably be run on either an OS/2 or UNIX platform. Two DEC Workstations and an OS/2 based personal computer are available for this purpose.

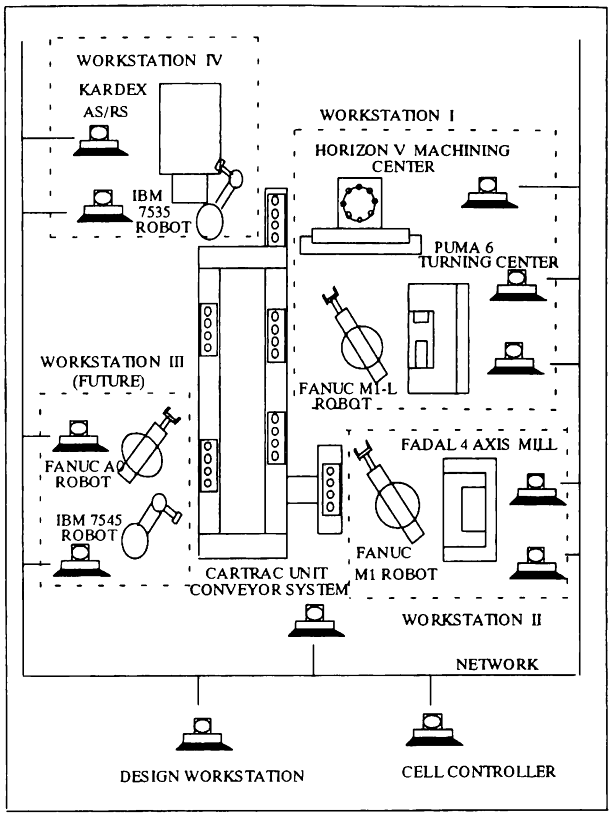

Figure 3. Penn State CIMLAB Layout

Table 2. Penn State CIMLAB equipment.

Item	Description
1	Daewoo PUMA 6 - CNC Lathe
2	Horizon V - CNC 2 1/2 Axis Milling Machine
3	FADAL VMC-20 4 Axis CNC Milling Machine
4	GMF M1-L Robot
5	GMF M1 Robot
6	IBM 7535 Robot
7	IBM 7545 Robot
8	GMF A0 Robot
9	Kardex AS/RS System
10	Cartrac Material Transport System

5. SIMULATION FOR CONTROL

There are three different modes of operation for this modified version of Arena/SIMAN. In emulation mode for the purpose of rule/capacity evaluation you would typically desire stochastic processing times and product mixes and hence require multiple replications. In emulation mode during control or while planning exception handling, you would typically desire deterministic data. In this case the data can be explicitly stated or read from a file or the shop floor database. Finally, in control mode, the actual parts being processed must be used and the data describing those parts and their processing must be available in a shop floor database.

Except for the physical control mode components, the structure of the Arena/SIMAN simulation developed for the TAMCAM lab is rather typical for manufacturing systems of this type. The processing machines and robots are modeled as equipment resources which must be seized prior to use. The conveyor and pallets are modeled as an AGV system with each pallet being an individual transporter and each conveyor stop being a station. Parts arrive at the Kardex storage system station and are queued to wait for a pallet to transport them to the first processing machine in the part route. In emulation mode, parts arrive according to an arrival distribution specified in the experiment frame. In control mode, part arrivals are explicitly defined at system startup and through a special "order task." This explicit definition is required since each part entity in the simulation represents a physical part being manufactured.

The "order task" is placed on the TIQ when the simulation is initialized. When the execution responds with a task completion message, a predefined "order file" which describes the newly ordered parts is read. After reading the order file, the simulation again places the order task on the TIQ. This mechanism allows the execution system to explicitly specify arrivals at any time during the operation of the simulation.

Part processing routes are specified in the experiment frame using the SEQUENCES element. In emulation mode, the part type is a stochastic variable whose value is determined upon arrival. In control mode, the part type is specified explicitly on start up. Once a pallet is available at the Kardex load/unload station, the part is loaded on the pallet using the Adept robot and is routed to the first processing station using a TRANSPORT block. Once the pallet arrives at the station, the part is picked up by either the Puma robot or the Adept robot (depending on the station) and is loaded on the specified machine tool for processing. Upon completion of processing, the part is unloaded with the robot and is placed on either next processing machine (if it is reachable by the robot) or back on a pallet at the load/unload station. Once on a pallet at the load/unload station, the part can be transported to another work station or back to the Kardex storage system.

Consider the portion of the TAMCAM simulation shown in Figure 4. These portions of the model and experiment frames represent loading, processing, and unloading parts on the Sabre machining center and the Leadwell lathe. In this workstation, the Puma robot is used to pick parts from pallets and load the parts on the machining center. According to the model segment, entities are initially transported to the Puma workstation where they are queued. Entities then attempt to seize the Puma robot and either the Sabre machining center or the Leadwell lathe (depending on the routing for the specific part type -this decision logic is not shown). Once both resources have been obtained, two tasks are executed. The first DELAY block represents the task *Pick* which instructs the robot to pick up the part from the specified station. The second DELAY block represents the *Put* task which instructs the robot to load the fixture of one of the machines. Upon completion of this task, the part has been loaded on the machining center, the robot has cleared the work volume, and the machining process can be started. The *Process* task associated with the final DELAY block in the model segment represents this task.

Assuming that the SM_Execute environment variable has been set, Arena/SIMAN will evaluate the expression *LocalExecOn* associated with each task to determine whether the tasks should be executed or emulated. If the tasks are being executed, Arena/SIMAN writes the appropriate task to the TIQ whenever the entity reaches the DELAY block. The execution module reads the task from the task initiation queue and implements the task on the physical device(s). Once the physical device completes its task, the execution module writes a "task complete" message to the task completion queue and the simulation entity is free to move through the DELAY block in the simulation. This physical control mandates that simulation time be synchronized with wall time ("slow mode"). By resetting the SM_Execute environment variable, all tasks will be emulated (using DELAY blocks with normally distributed delay times in this example) and the simulation can run in "fast mode." Similarly, individual tasks can be emulated by adjusting the expression *LocalExecOn.*

During control mode operation, fast mode can also be used as a decision making tool. At any point during slow mode operation a *copy* of the simulation can be initialized with the current physical system state and run for some period in fast mode to evaluate the impact of the decision on the system performance. Assuming that the fast mode simulation runs sufficiently fast, several alternatives can be evaluated and the most promising alternative can be used by the control mode simulation. The use of simulation as a "look ahead" scheduling tool has been described previously by Wu and Wysk (1988), Harmonosky and Robohn (1991), and Cho (1993).

6. CONCLUSIONS

A simulation-based shop floor control system has been described in this paper. Enhancements to the Arena/SIMAN simulation language allow the simulation model used for design, performance analysis and planning, to also be used for direct shop floor control. Consequently the same control logic is reused and need not be implemented multiple times. Additionally, the simulation can also be used to predict future system performance based on current decisions by running in fast mode from the current physical system state. The results of the fast mode analysis can be used by the decision making functions in control mode.

The simulation-based control system described in this paper has been implemented and is currently being used in two laboratories: one at Texas A&M University and the other at the Pennsylvania State University. Initial experience with these labs appears quite promising. Further details about this wörk is available on the TAMCAM World Wide Web (WWW) home page at the URL address http://tamcam.tamu.edu/tamcam.html.

```
TRANSPORT: Pallet,PumaWKSTN,,,Move;
             .
             .
STATION,PumaWKSTN;
QUEUE,PumaWKSTN;
SEIZE: PUMA:
       SABRE;
DELAY: Normal(1,.15),,Pick;
DELAY: Normal(1,.15),,Put;
DELAY: Normal(15,3),,Process;
           .
           .
STATION,PumaWKSTN;
QUEUE,PumaWKSTN;
SEIZE: PUMA:
       LEADWELL;
DELAY: Normal(1,.15),,Pick;
DELAY: Normal(1,.15),,Put;
DELAY: Normal(15,3),,Process;


Experiment:

TASKS:    1,Move,LocalExecOn,"Move %1.0f TGID=%1.0f Loc=%1.0f",
            IDNUM,IDENT,MSQ(NS,IS):
          2,Pick,LocalExecOn,"Pick %1.0f TGID=%1.0f Plan=%f MHE=%s",
            IDNUM,IDENT,PLAN,STR(Robots,m):
          3,Put,LocalExecOn,"Put %1.0f TGID=%1.0f Plan=%f Sentity=%s
            slot=%1.0f",IDNUM,IDENT,PLAN,STR(Machines,m),slotID:
          4,Process,LocalExecOn,"Process %1.0f TGID=%1.0f Plan=%f",
            IDNUM,IDENT,PLAN;
```

Figure 4. Sample Arena/SIMAN code including the real-time control modifications.

7. REFERENCES

Cho, H., *An Intelligent Workstation Controller for Computer Integrated Manufacturing*, Ph.D. Thesis, Texas A&M University, 1993.

Drevna, M. and Kasales, C., "Introduction to Arena," *Proceedings of the 1994 Winter Simulation Conference*, M. Tew and S. Manivannan, Eds., IEEE Publishers, Piscataway, NJ.

Harmonosky, C. M. and Robohn, S. F., "Real-time Scheduling in Computer Integrated Manufacturing: A Review of Recent Literature," *International Journal of Computer Integrated Manufacturing*, Vol. 4, No. 6, pp. 331-340, 1991.

Smith, J. S. and Joshi, S. B., "Message-based Part State Graphs (MPSG): A Formal Model for Shop Floor Control," Texas A&M University Working Paper Series, 1993.

Wu, S. D. and Wysk, R.A., "Multi-pass Expert Control Systems - A Control/Scheduling System for Flexible Manufacturing Cells," *Journal of Manufacturing Systems*, Vol. 7, pp. 107-120, 1988.

Wysk, R. A., Joshi, S. B., and Pegden, C. D., "Rapid Prototyping of Shop Floor Control Systems for

Computer Integrated Manufacturing," ARPA project # 8881, 1992.

AUTHOR BIOGRAPHIES

JEFFREY S. SMITH is an assistant professor in the Industrial Engineering Department at Texas A&M University. His research interests are in shop floor control, manufacturing system design, analysis and control, and simulation. He is an active member of IIE, SME, and IEEE.

RICHARD A. WYSK is the Royce Wisenbaker Chair in Innovation at Texas A&M University. His primary research areas include computer aided process planning and flexible automation systems. He is the recipient of the IEE David F. Baker Distinguished Research Award (1993) and is a Fellow of IIE.

DAVID T. STURROCK is product manager in software development with Systems Modeling Corporation in Sewickly, Pennsylvania. He joined SMC in 1988 after more than 11 years of manufacturing experience. He has applied simulation techniques to problem solving in the areas of transportation systems, scheduling, plant layout, capacity analysis, and process design.

SANJAY R. RAMASWAMY is a Ph.D. student in the Industrial and Management Systems Engineering Department at the Pennsylvania State University.

GLEN D. SMITH is a Ph.D. student in the Industrial and Management Systems Engineering Department at the Pennsylvania State University. His research interests are in manufacturing system simulation, computer control of manufacturing systems, and artificial intelligence.

SANJAY B. JOSHI is an associate professor in the Industrial and Management Systems Engineering Department at the Pennsylvania State University. His research and teaching interests are in the areas of computer aided design and manufacturing with specific focus on computer aided process planning, control of automated flexible manufacturing systems, and integration of automated systems.