Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

SIMULATION OF FLEXIBLE CONTROL STRATEGIES

Glen D. Smith
D. J. Medeiros

Industrial and Manufacturing Engineering Department
Penn State University
University Park, PA 16802, U.S.A.

ABSTRACT

Flexibility in developing and testing operating policies for manufacturing systems requires the ability to quickly and easily change a simulation model to incorporate the new policy. To achieve this goal, the model of the system control strategy should be separated from the model of the physical system. Using a flexible manufacturing cell as an example, we first show how this separation can be accomplished, and then discuss desirable features in a language that would simplify the approach.

1 INTRODUCTION

Selection of an appropriate operating strategy is an important issue in many manufacturing systems; issues such as job scheduling, material handling system operating rules, buffer sizes and line rates can have a major impact on productivity (Norman et al. 1992). Simulation has long been recognized as a method of evaluating alternative system control policies in manufacturing systems, as evidenced by the Manufacturing Simulation track which has become a feature of the Winter Simulation Conferences.

In this paper, the term "control strategy" will be used to refer to a number of related decisions in manufacturing system operation, such as:

- sequencing of operations at a work center,
- scheduling operations across multiple work centers,
- assignment of resources (including manufacturing equipment, personnel, and material handling equipment) to specific operations, and
- assignment of operations to specific resources, including splitting or combining lots.

Examples of control strategies that are evaluated using simulation include:

- the effect of running a line in a "just-in-time" mode (Corbett and Yucesan 1993),
- the effect of various alternative process plans, or flexible process plans (Mauer and Schelasin 1993),
- the effect of using complex scheduling rules, such as scheduling to a bottleneck machine, lot splitting, or scheduling to minimize job cost or flowtime (Ernst and Matevosian 1993), and
- the effect of different allocation rules for interacting resources such as personnel, fixtures, and tooling (Kashyap and Khator 1994).

Development of a good control strategy (by testing and evaluating several alternatives) is particularly important in automated manufacturing systems (such as FMS) or semi-automated manufacturing systems (such as production lines). On-line data collection in such systems provides a basis for global control of the system by ensuring that information concerning system status is readily available. Control strategies tend to be complex because of part variety and interactions among resources (Evans et al. 1994). Furthermore, poor control strategies cannot be overcome by human intervention because the opportunities for such intervention are limited.

Although simulation is frequently used to evaluate alternative control strategies, the process is difficult and time-consuming. Most discrete event simulation languages employ an entity flow world view (Schriber and Brunner 1994); these languages typically support models in which the control logic is distributed throughout the simulation model via methods such as conditional branching of

entities or interaction among multiple resources. Changing from a push to a pull strategy, for example, often means completely rewriting the simulation model. Simulation based finite schedulers typically allow queue disciplines to be specified for a single operation or group of operations, but don't easily support the concept of refraining from processing a waiting job because a higher priority one will arrive in a short time period. Davis et al. (1993) discuss the difficulty of modeling control strategies in current simulation languages with specific reference to flexible manufacturing systems.

Because the control strategy is distributed throughout the model, major changes in strategy can mean major changes in the model code. The time required to build, verify, and validate a simulation model is substantial, and these steps must be repeated each time the model code is changed. To overcome this difficulty, new simulation languages have been proposed which use concepts such as autonomous agents (Lin and Solberg 1994) and objects (Davis et al. 1993) to model the control strategy.

Object-oriented programming has been recognized as an enabling technology for implementing a flexible control strategy. Goble (1994) shows that the language SIMOBJECT separates process and routing behavior, allowing routers to be easily replaced. Mize et al. (1992) present a conceptual approach which divides a model into three types of objects: physical objects, information objects, and control objects. Control elements are modeled as three classes: queue controllers, assembly queue controllers, and work order controllers. Bodner et al. (1993) model controllers as objects and illustrate deadlock detection and resolution in an FMS.

In this paper, we demonstrate that the separation of the control strategy from the system description can be implemented in a conventional simulation language, thereby combining the benefits of flexibility with existing simulation language technology. We illustrate the approach using SIMAN to model a FMS, then discuss features of a simulation language which would simplify the separation of system description and control.

2 EXAMPLE SYSTEM

An example small manufacturing system is described; the system is part of the Penn State Flexible Manufacturing System described by Smith et al. (1994), and is shown in Figure 1. The system will be used to illustrate some of the control decisions that must be incorporated in a manufacturing system control strategy, and a flexible method for doing so.

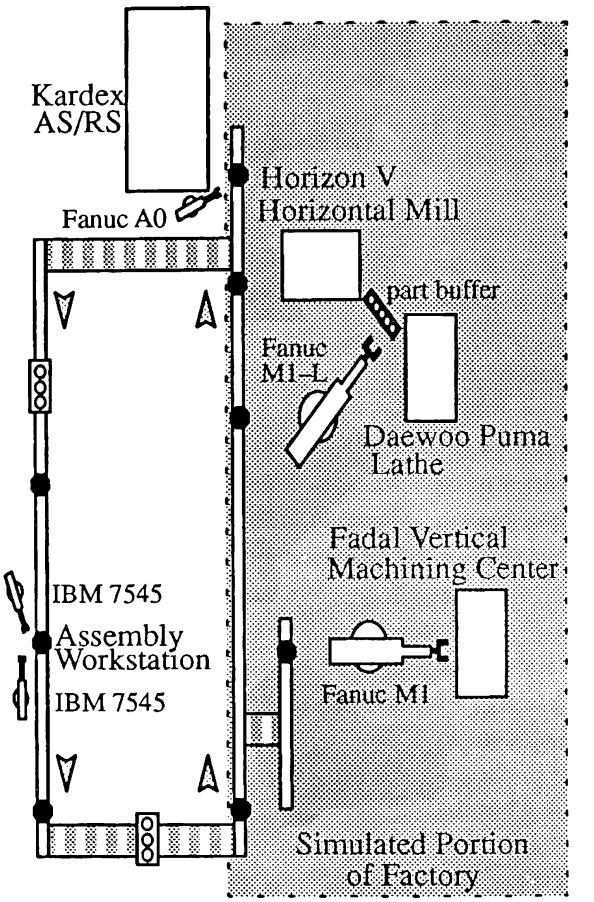

Figure 1: Flexible Manufacturing System

The system consists of an input point, 2 machining cells, and an exit point, with all of these connected by a transport system. The first processing cell contains a turning center and a horizontal machining center and the second contains only a vertical machining center. The input and exit points are located at the Kardex vertical storage system. Transportation is performed using a cart-on-track system. Cart stopping points are shown in Figure 1, and the direction of travel is indicated by the arrows. A variety of parts are produced in the facility and parts may have alternative process plans.

Parts enter this manufacturing system at the input

point. Here large part batches may be divided, or smaller batches may be combined, in accordance with the requirements of the transportation, tooling, and fixturing available. After batching, parts are fixtured or kitted then transported to the first processing station. The production route may be completely specified at the input point, or the routing decision may only include the first processing step to be performed. Decisions about batching and routing at the input point must consider the shop wide state of the transportation system, the load at each processing center, and the tooling and fixturing available.

Processing cells consist of work center(s), an industrial robot to move parts within the cell, and buffer space. When a part arrives at a processing center it may be loaded directly into a work center or may loaded into a buffer to wait. Similarly, when parts have completed processing at a work center they may be moved to another work center, moved directly to a transport device if one is available, or moved to an in-process buffer for storage. If dynamic routing is implemented this movement decision must include the determination of the next processing step to be performed. Refixturing may also be required based on the routing decisions made.

The parts are transported to the exit point after they have completed processing within the system. Parts are defixtured and de-kitted, and these components are returned to the input point for reuse. The original batches are also reassembled at the exit point for release from the system.

Even in this simplified version of a facility, routing determination is complex and requires information about the shop wide state of equipment, fixturing, tooling, and other jobs. Decisions made at one work center or cell may impact the alternatives available at another station or at a later time. Clearly, experimentation with various control strategies will be required to develop one which meets system objectives while retaining flexibility.

3 MODELING METHODOLOGY

Description of the model is divided into two sections. First, the model of the physical system is described, followed by the model of the control system.

3.1 Physical System

An entity representing the physical part is created in the simulation when an order is entered. At this time a data record of the part is also created in the simulation's control section. Communication between the control system and the part entity is performed through discrete channels. The control system issues commands to the part entity consisting of resources required and production parameters, such as processing times. Part entities are assumed to perform the commanded tasks using the specified resources and parameters. After completion of the issued commands, the part notifies the control system.

In our current implementation the physical system is described using SIMAN's station elements, and the communication channels are implemented as signals and global variables.

A station in the simulation represents a machine, manufacturing cell, or other piece of equipment in the physical facility. The simplest set of actions at a station are to process a part via a time delay then signal process completion (Figure 2). For this type of station, any required resources must have been previously allocated to the entity representing the part.

```
;
; * Machining Stations
;

STATION, 3 - 5;
DELAY: PartParam(ParmIndex,ProcTime);
ASSIGN:  SigIssue = PartID;
SIGNAL:  SigMach;
QUEUE,   M+10;
WAIT:    PartID, 1;
ROUTE:   RobotID;
```

Figure 2: Code for Machining Station

In the code shown in Figure 2, the ProcTime attribute represents the processing time at the machine tool. The assign and signal statements implement the notification to the control system that the part has completed processing. After notifying the controller of completion, the part waits, continuing to hold any allocated resources (in this instance, a machine tool resource), until receiving a command from the controller to continue.

More complex stations include additional actions, but are similar to the simple stations in that they also perform a task, signal task completion to the controller, and wait for

further instructions. In this implementation, robot stations (see Figure 3) are used to allocate both the robot and machine tool resources. This is done to satisfy the physical preconditions that the machine resource is not deallocated until the part has been removed from the machine by the robot. If such preconditions were not required, the machine tool could have been allocated in the previously described station.

```
; *
; * Robot Station
; *
        STATION, 1 - 2;
        BRANCH,  1:
          IF,    HeldRes == 0, GetRes:
          ELSE,  FreeRes;

FreeRes RELEASE:Equip(HeldRes);
GetRes  ASSIGN:
         RobotID =
          PartParam(ParmIndex,VRobot);
         HeldRes =
          PartParam(ParmIndex,VMach);
        QUEUE,    M;
        SEIZE:    Equip(RobotID),1:
                  Equip(HeldRes),1;
        DELAY:    MoveTime;
        RELEASE:  Equip(RobotID);
        ASSIGN:   SigIssue = PartID;
        SIGNAL:   SigRobot;
        ROUTE:    HeldRes;
```

Figure 3: Code for Robot Station

The branch statement releases any resources held by the part. The first assignment stores routing instructions for the entity representing the part. Because the robot to be used for machine loading and unloading is assigned to a specific cell, it can be determined in the cell logic. If this was not the case, the robot to be used for unloading the part from the machine would be contained in the control command to the part. The seize command obtains the robot and the machine that the part is to be moved to. After the move has been completed the robot resource (only) is released. The control system is notified that the movement has been completed and the robot is free to perform another task. At this point, the part is sent to the machine station to undergo processing.

3.2 Control Logic

In this implementation, we assume that control decisions are required only at the completion of tasks assigned by the control section. Entities in the control section represent the channels from parts in the physical section of the model. Each of these entities performs an appropriate action based on the state of the entire system. Figure 4 illustrates a portion of the control logic responsible for routing an order when it arrives to the system.

```
        BRANCH,1:
          IF,SigID==SigEnter,EvEnter:
          IF,SigID==SigRobot,EvRobot:
          IF,SigID==SigMach,EvMach:
          IF,SigID==SigDone,EvDone;

EvEnter BRANCH,1:
         IF,WIPAmount<MaxWIP,BringIn:
         ELSE, ResetEv;

BringIn ASSIGN:WIPAmount=WIPAmount+1;
        BRANCH,1:
          IF,Mach1WIP<Mach1Cap,DoM1:
          IF,Buff1WIP<Buff1Cap,DoB1:
          ELSE,OverCapErr;

DoM1    ASSIGN:Mach1WIP=Mach1WIP+1:
           PartParam(SigIssue,VMach)=
             Machine1:
           PartParam(SigIssue,VPTime)=
             Proc_Time_M1;
        SIGNAL:SigIssue:
               NEXT(ResetEv);

DoB1    ASSIGN:Buff1WIP=Buff1WIP+1:
           PartParam(SigIssue,VRobot)=
             Robot1:
           PartParam(SigIssue,VMach)=
             Buffer1;
        SIGNAL:SigIssue:
               NEXT(ResetEv);
```

Figure 4: Control Logic

To simplify exposition, the code fragment in Figure 4 shows control logic in which all parts are directed to follow the same production plan: they may wait to enter the system, be assigned to Machine1 or be assigned to Buffer1. If the number of parts currently in the system would exceed a specified quantity, the new part will be held in a queue of work waiting to enter the system (not shown). If the part is allowed to enter the system, it will be routed to Machine1 if sufficient capacity exists at the machine, or to Buffer1 otherwise.

If the part is to enter the system, the control entity

establishes a communication channel to the part (by utilizing global variables to pass information concerning the appropriate resource identifiers and processing time), then signals the part entity to proceed. The part entity in turn reads the communication channel and implements the instructions contained therein (see Figures 2 and 3). The variable 'SigIssue' is the identification of the part which issued the message to the control section.

As previously noted, the example has been greatly simplified for ease of presentation. The control system illustrated only considers the alternatives 'move to the buffer', 'move to the processing machine' or 'do not enter the system'. In the full simulation, the decision making logic is greatly expanded to allow flexibility of routing and scheduling. Each part type has a set of feasible process plans, one of which is implemented based on system status.

4 BENEFITS OF THE APPROACH

This explicit separation of the control strategy from the physical system logic allows us to 'plug-in' alternative decision making methods. Changes are confined to the section of the model responsible for implementing the control strategy, and verification activities can focus on this section of the model.

Additionally, this implementation allows the decision maker to perform operations on one part based on the actions posted by another part. Thus, quite complex state-dependent control strategies can be implemented in a straightforward manner. If the control strategy requires operations at specific times or intervals in addition to the discrete events as above, timer entities similar to the event catching control entities could be easily implemented. These timers would perform a delay instead of issuing a wait for a part message.

Implementing an approach that separates control strategy from the physical flow of entities provides benefits to the designers responsible for developing and implementing a system controller, as well as for users attempting to optimize system operations. Use of a conventional simulation language provides advantages in developing a model of the physical system, but is somewhat awkward for modeling control systems, as compared to the object-oriented approach. Explicit support for a control logic model in conventional simulation languages is required, perhaps through use of a modeling paradigm such as Petri Nets. Pointers or explicit message passaging functions would also simplify the modeling effort. This paper has illustrated some of the functions that would be performed by such a model, and how it could interact with the entity flow model.

REFERENCES

Bodner, Douglas A., Suzanne Dilley-Schneider, S. Narayanan, Uday Sreekanth, T. Govindaraj, Leon McGinnis, and Christine Mitchell. 1993. Object-Oriented Modeling and Simulation of Automated Control in Manufacturing. In *IEEE International Conference on Robotics and Automation*, vol 3, pp. 83-88.

Corbett, Charles and Enver Yucesan. 1993. Modeling Just-In-Time Production Systems: A Critical Review. In *Proceedings of the 1993 Winter Simulation Conference*, pp. 819-827.

Davis, Wayne J., Duane Setterdahl, Joseph Macro, Victor Izokaitis, and Bradley Bauman. 1993. Recent Advances in the Modeling, Scheduling and Control of Flexible Automation. In *Proceedings of the 1993 Winter Simulation Conference*, pp. 143-155.

Ernst, Thomas, and Avetik P. Matevosian. 1993. A Flexible Assembly Global Control Simulation. In *Proceedings of the 1993 Winter Simulation Conference*, pp. 897-903.

Evans, Gerald W., William E. Biles, and Michael W. Golway. 1994. Simulation of Advanced Manufacturing Systems. In *Proceedings of the 1994 Winter Simulation Conference*, pp. 141-148.

Goble, John G. 1994. SIMOBJECT: From Rapid Prototype to Finished Model - A Breakthrough in Graphical Model Building. In *Proceedings of the 1994 Winter Simulation Conference*, pp. 437-442.

Kashyap, Arun S., and Suresh K. Khator. 1994. Modeling of a Tool Shared Flexible Manufacturing System. In *Proceedings of the 1994 Winter Simulation Conference*, pp. 986-993.

Lin, Grace Y., and James J. Solberg. 1994. An Agent-Based Flexible Routing Manufacturing Control Simulation System. In *Proceedings of the 1994 Winter Simulation Conference*, pp. 970-977.

Mauer, John L., and Roland E. A. Schelasin. 1993. The Simulation of Integrated Tool Performance in Semiconductor Manufacturing. In *Proceedings of the 1993 Winter Simulation Conference*, pp. 814-818.

Mize, Joe H., Hemant Bhuskute, David Pratt, and Manjunath Kamath. 1992. Modeling of Integrated Manufacturing Systems Using an Object-Oriented Approach. *IIE Transactions*, vol. 24, no. 3, pp. 14-26.

Schriber, Thomas J., and Daniel T. Brunner. 1994. Inside Simulation Software: How It Works and Why It Matters. In *Proceedings of the 1994 Winter Simulation Conference*, pp. 45-54.

Smith, Jeffrey J., Richard A. Wysk, David T. Sturrock, Sanjay E. Ramaswamy, Glen D. Smith, and Sanjay B. Joshi. 1994. Discrete Event Simulation for Shop Floor Control. In *Proceedings of the 1994 Winter Simulation Conference*, pp. 962-969.

AUTHOR BIOGRAPHIES

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

D. J. MEDEIROS is an Associate Professor in the Department of Industrial and Manufacturing Engineering at Penn State University. She holds a B.S.I.E. from the University of Massachusetts and an M.S.I.E. and Ph.D. from Purdue University. Her research interests are in computer integrated manufacturing systems, material handling, and applications of coordinate measuring machines. She is a member of IIE and the College Industry Council on Material Handling Education.