Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

IMPROVING SIMULATION MODEL ADAPTABILITY WITH A PRODUCTION CONTROL FRAMEWORK

Sean M. Gahagan
Jeffrey W. Herrmann

Department of Mechanical Engineering
and Institute for Systems Research
University of Maryland
College Park, MD 20742, U.S.A.

ABSTRACT

Simulation models provide a powerful tool for the analysis of manufacturing systems, but their utility beyond the design stage of the system life cycle is hampered by the high cost of model maintenance. To reduce this cost, models must be made more adaptable. We believe that adaptability can be increased by separating the flow of material from the flow of information through a model system, especially with respect to changes related to production control. Coordination of these flows, however, requires a production control framework. In this paper, we propose a three-level, hierarchical production control framework and define the elements necessary to implement it in a simulation model. We demonstrate the use of this approach by considering a simple flow shop undergoing production control changes. We define the parameters of the shop using the framework and implement the changes with little effort.

1 INTRODUCTION

Simulation models are useful tools for the design and development of manufacturing systems, but their utility, while seldom fully exploited, extends well into the system life cycle. The reason that their capabilities are not fully utilized is due to the cost of maintaining an accurate simulation model. The cost of maintenance may be expressed in terms of man-hours necessary to update a simulation model, and with today's simulation tools, those hours can quickly mount without producing any value added for the organization. Two options exist to reduce the cost of model maintenance: reduce the number of hours required to make changes to a simulation model by making them more adaptable, or eliminate those hours altogether by automating the maintenance process. It is clear that fulfilling the former option may facilitate the latter.

The objective of this paper is to introduce the use of a production control framework to increase the adaptability and utility of simulation modeling for the study of manufacturing systems.

2 BACKGROUND

This section summarizes the authors' previous work in the area of simulation model adaptability and principal concepts used in the development of this research.

2.1 Simulation Model Adaptability

The dictionary defines adaptability as the ability to fit a specific or new use or situation, often by modification (Webster's, 1988). Simulation model adaptability is the ease with which a simulation model can be modified, either to conform to changes in the system it represents, or to demonstrate the effect of changes to the system. Simulation models that cannot be changed easily may not be changed at all and become obsolete (Banks, Gibson, 1998).

Herrmann, Lin, Ram and Sarin (2000) developed a measurement technique to study the adaptability of simulation models. Their technique measured adaptability in terms of user effort. Effort was measured by counting the number of actions a user was required to take in order to build and then modify a simulation model. The effort required to modify the model was then scaled relative to the effort required to build the entire model. They used two different simulation modeling packages to build models of a manufacturing system and then implement a set of changes. The results of their study indicated that changes to the structure of a manufacturing system require greater effort than changes to the model parameters and thus reduce adaptability.

A closer look at the models used in the study and the data derived from them yields further insight into the nature of simulation model adaptability. The manufacturing system used in the study, Camile Motor Works (McKay and Moore, 1991), demonstrates a variety of production control challenges, including conveyor and guided vehicle material handling, multiple machine dispatching and management of a variety of workers. However, the portions of the model that required the most effort to build and modify were those that had relatively simple production control rules. The reason for the increased effort in these areas was that the simulation packages featured pre-defined modules to represent the most difficult production control challenges, but few to represent simple, but uncommon, challenges. For example, the machine shop, a five unit FMS fed by an AGV network, proved more adaptable than the assembly department, where workers at five work benches assembled components into sub-assemblies. The reason for this dichotomy was that the simulation package used included pre-defined modules to represent the AGV network, such that the most complicated control rules were reduced to simple parameters. The purpose of this work is to develop modules that represent a very wide range of production control policies in a similarly parametric form.

2.2 Lean Manufacturing

Lean manufacturing is a term used to describe an operations management philosophy focused on reduction of waste in a manufacturing system. Lean systems are characterized by low levels of inventory, often facilitated by pull-type production control mechanisms (Womack and Jones, 1996). Lean systems are described in terms of value streams, sequences of tasks necessary to convert raw materials into finished goods. Value stream maps, a common tool for lean management, differentiates the flow of material from the flow of information through a manufacturing system (Rother and Shook, 1999). The differentiation of these two flows is the basis of this research.

2.3 Production Control

Production control can be generalized into two categories, push and pull. In a push production control, the flow of material is regulated at the first operation in the process. Once material is released to this operation, it continues through the system as fast as production resources allow. This type of control can result in high levels of work in process as material stacks up between a fast process and proceeding slow process. Pull production controls the amount of work in process, sometimes through the use of kanban cards, and governs production at an operation near the end of the process (Hopp and Spearman, 2000). The two production control policies meet at the push-pull interface. Every manufacturing system has a push-pull interface.

2.4 Objective

The objective of this research is to increase the adaptability of simulation models of manufacturing systems by developing a production control framework. Application of a production control framework will enable changes in production control, which are primarily structural, to be implemented parametrically.

3 DESCRIPTION

This section describes a three-level production control framework that increases the adaptability of simulation models with complex production control by reducing a wide range of production control policies to a set of simple parameters.

3.1 Production Control Framework

The production control framework described in this paper is a variation on the shop floor control architecture proposed by Smith, Hoberecht and Joshi (1996). Many other standard control architecture models have been proposed in the literature (Vieira, 1998), but the Smith model is unique among them in that it directly addresses the domain of shop floor control. The proposed production control framework adopts the same architecture as the Smith model, but adds greater detail at the lowest level in order to more completely describe the interaction of material and information on the shop floor. The proposed framework is shown in Figure 1.

Figure 1: Production Control Framework

The lowest level in the framework is the queue. Queues order the components to be processed. When signaled they release components for processing. The framework uses the term queue rather than the Smith model of equipment as the lowest level because the framework requires more than one queue to enable processing at a piece of equipment. The second level of the framework is the workstation. A workstation is a collection of equipment, tools and personnel, usually physically separated from other workstations. The workstation controller monitors the state of the queues in the workstation and determines if there are enough components to complete a process. If there are, it first signals the queues to release components and then processes them. The highest level of control is the shop. At the shop level, the overall production control policy is implemented. The shop controller determines if a workstation is to be operated in a push or pull control policy and coordinates the flow of information throughout the system.

3.2 Production Control Elements

To leverage the advantages of software re-use, each element in the framework must be defined in terms of parameters and the functions detailed in order to realize the framework in a simulation model.

3.2.1 Components

Traditionally, the term component referred only to the physical elements of a product, but in the proposed framework both material and information are considered components. We have identified four distinct component types to be used in the proposed framework:

Type 1, Material Components – Material components are components in the classic sense; physical inventory of raw materials that the system transforms into finished goods. Type 1 components may be discretized bulk items like meters of steel stock or they may be individual parts like nuts or bolts.

Type 2, Production Permission Components – Production permission components are signals transmitted through the system indicating that a transformation process can begin. Type 2 components are analogous to kanban cards or other physically realized production control mechanism that transmit a simple "Go" instruction.

Type 3, Resource Permission Components – Resource permissions, like production permissions, are signals transmitted through the system, but unlike production permissions, they provide specific information about how a process will be completed, specifically, what system resources are to be used in a transformation process. The number and type of resource permissions controls the utilization of system resources. If a system contains three processing machines, it also contains three resource permission components, one corresponding to each machine. Resource permissions are also used to control the utilization of workers, tooling and any other capacity-limited system resource.

Type 4, Batch Components – Batch components are collections of types 1, 2 and 3 components that are to be processed as a single unit. A batch component may be used to represent a part held in a fixture, a kanban card attached to a bin of material or any other collection of components that are processed together.

The disparate natures of the component types require different types and quantities of information to be carried with them in the form of component attributes. The framework accommodates these different requirements in the form of standardized attributes for each component type. The framework defined in this paper is represented in matrix-vector notation. This style of notation was chosen strictly as an organizational mechanism, rather than to facilitate any type of mathematical manipulation. Consequently, a component **c** is defined by a vector as follows:

$$\mathbf{c} = [c_1, c_2, \ldots, c_{nc}]^T \tag{1}$$

where c_i corresponds to component attribute i. The primary attribute, c_1, is the component type, defined as above. The number of component attributes, nc, is dependent on the component type, as shown in Table 1, below.

Table 1: Component Types and Number of Attributes

Component Type	c_1	nc
Material	1	14
Production Permission	2	6
Resource Permission	3	9
Batch	4	17

All components, regardless of type, share a set of five common attributes, c_i, as defined in Table 2, below.

Table 2: Common Component Attributes

i	Description
General Attributes	
1	Component Type
2	Component Class
Destination Attributes	
3	Shop
4	Workstation
Temporal Attribute	
5	Queue Entry Time

The component class attribute defines general categories within each component type. This attribute could be a part number, a machine class, a worker skill type or any other user defined subdivision within which the components are functionally equivalent. The destination attributes, shop, workstation and process, represent the address of the component's next destination in terms of the production control framework. The destination address of types 1 and 4 components are updated according to the component process plan each time they complete a process step. Destination attributes for other component types do not change and serve as a return address. The queue entry time attribute is used to order components for processing based on the order in which they arrived at a queue. The queue entry time attribute is updated each time the component enters a queue.

The remainder of the component attributes are functions of the component type. Some attributes provide data necessary for production control, some record data necessary to measure system performance and some perform both functions. Tables 3, 4, 5 and 6 describe the type-specific attributes, c_i, for component types 1, 2, 3 and 4, respectively. The tables also indicate the function of each attribute, where C indicates that the attribute is used for control, M indicates that the attribute is used for measurement and C/M indicates that it may be used for both.

Table 3: Type 1 Component (Material) Attributes

i	Attribute	Function
In Addition to the Attributes in Table 2		
Temporal Attributes		
6	Workstation Entry Time	C/M
7	Shop Entry Time	C/M
Queue Attributes		
8	Imminent Setup Time	C
9	Imminent Processing Time	C
10	Gross Imminent Processing Time	C
11	Due Date	C
12	Process Time Remaining	C
13	Processes Remaining	C
14	Static Slack Time	C

Temporal attributes are used to measure the time a component spends under the control of a given control element. Each temporal attribute is updated when the component visits a controller of the given type. Temporal attributes may also be used to order components in a queue. They are updated each time a component visits a controller of the given type. The queue attributes in Table 3 were chosen specifically because they are static in nature. That is, these attributes' values do not change while a component waits in queue. Although there is a wide range of dynamic queue attributes used in practice, not all simulation software is capable of implementing dynamic queue rules. For greater detail regarding queue attributes, see Panwalker and Iskander (1977).

Table 4: Production Permission Component Attributes

i	Attribute	Function
In Addition to the Attributes in Table 2		
Temporal Attribute		
6	Due Date	C/M

In Table 4, the only temporal attribute is due date. It is primarily used to measure the response time of a system. Each production permission component constitutes demand for products. How quickly the system fills such demand is an important measure of system performance. This measure may be improved if the due date attribute is also used to order components in queues.

In Table 5, the resource index attribute is used to specify a particular member of a resource class. Each member of a resource class is assigned a unique resource index. The time resource seized attribute is used to measure machine utilization. The queue attributes are used primarily to measure time-averaged utilization, but they may also be used to implement load balancing dispatching rules, based either on number of times a resource was accessed or the total time a resource has spent in use.

Table 5: Resource Permission Component Attributes

i	Description	Function
In Addition to the Attributes in Table 2		
Identification Attribute		
6	Resource Index	C
Temporal Attribute		
7	Time Resource Seized	M
Queue Attributes		
8	Cumulative Use, Occurrences	C/M
9	Cumulative Use, Time	C/M

Table 6: Batch Component Attributes

i	Description	
In Addition to the Attributes in Tables 2 and 3		
Batch Attributes		
15	Type 1 Component Quantity	C
16	Type 2 Component Quantity	C
17	Type 3 Component Quantity	C

For attribute indices 1-10 and 12-14, batch components take on the values of the primary material component in the batch. That is, the type 1 component with the lowest component class attribute. Attribute 11 is taken from the primary production permission component. The batch attributes are used for production control book-keeping when the batch is split up and subsequently reassembled at each control level.

The attributes described here are not intended to be a comprehensive list of attributes necessary to build a functioning simulation model, and may be augmented by others to facilitate realization in a particular simulation model. They may include object identification, sequence data, process plan or routing step number or subsequent step number.

3.2.2 Queue Controller

The lowest level of production control is the queue controller. A queue is a collection of similar objects, ordered according to some queue discipline (Law and Kelton, 1991). The simplest queue disciplines are based on the value of an object attribute and are ordered in either ascending or descending order. A queue controller **q** has four parameters:

$$\mathbf{q} = [q_1, q_2, q_3, q_4] \qquad (2)$$

q_1 and q_2 correspond to the type and class of the components in the queue, q_3 identifies the component attribute, c_i, to be used to order the queue and q_4 is the order gradient, where 0 indicates ascending, 1 descending and 2 gradient, where 0 indicates ascending, 1 descending and 2 random. By careful selection of the parameters q_3 and q_4, a number of material and resource sequence rules can be realized.

Material Sequences – Material sequences, also called queue disciplines or dispatching rules, refer to the parameters of types 1 and 4 component queues. The most exhaustive list of queue disciplines is Panwalker and Iskander (1977). Using the proposed framework, 16 of their 35 'Simple Priority Rules', and 8 of the 11 most commonly used in practice (Vollmann, Berry and Whybark, 1997) can be implemented. Table 7 lists the parametric descriptions of the eight queue disciplines and their common names.

Table 7: Material Queue Discipline Parameters

Queue Discipline	q_3	q_4
First Come / First Served (FCFS)	5	0
Shortest Processing Time (SPT)	9	0
Earliest Due Date (EDD)	11	0
Least Work Remaining (LWR)	12	0
Fewest Operations Remaining (FOR)	13	0
Slack Time (ST)	14	0
Least Setup (LSU)	8	0
Random (RAND)	Any	2

Resource Sequences – Resource sequences control the utilization of system resources. By modeling resource permissions as components and placing them in queues, queue control can be used to implement simple, static resource sequence rules. Table 8 lists the parametric descriptions of five resource sequence rules and their common names.

Table 8: Resource Sequence Rule Parameters

Dispatching Rule	q_3	q_4
First Available Resource	5	0
Preference Order	6	0
Least Accessed Resource	8	0
Least Used Resource	9	0
Random (RAND)	Any	2

Queues are common elements of simulation modeling software. The ability of a software package to order the objects in a queue is not uncommon and is a prerequisite for compatibility with the framework. Therefore, the function of a queue controller will not be described in this paper. To work within the framework, a queue must have the ability to indicate the number of elements it contains and it must be able to release a number of elements in response to a signal or method call. For more explicit details of queue operation, refer to Law and Kelton (1991). Queues, like components, require attributes in addition to those described in this framework to function. Those attributes, again, vary from package to package and will not be further defined here.

3.2.3 Workstation Controller

The second level of control is the workstation controller. A workstation is a set of system resources and associated queues. This controller is responsible for coordinating two or more queue controllers to complete processes. A process is a task that requires time, material and system resources to complete. A workstation controller **w** has three components:

$$\mathbf{w} = [\mathbf{Q}, \mathbf{X}, \mathbf{D}] \tag{3}$$

Q is a set of nq queue controllers in the workstation, **X** is a set of nx feasible process combinations and **D** is a set of nq post-process dispositions.

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{nq}]^T \tag{4}$$

$\mathbf{q}_i$ is a queue controller as described in the previous section.

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{nx}] \tag{5}$$

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{i,nq}]^T \tag{6}$$

x_{ij} is the number of components from q_j necessary to carry out process i and nx is the number of processes that can be carried out by the workstation.

$$\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{nx}] \tag{7}$$

$$\mathbf{d}_i = [d_{i1}, d_{i2}, \ldots, d_{i,nq}]^T \tag{8}$$

d_{ij} is the post process disposition of components from q_j after completing process i. If $d_{ij} = 0$, the component is to be included in an output batch. If $d_{ij} = 1$, the component is to be released to return to its point of origin. If $d_{ij} = 2$, the component is to be destroyed. If $q_{j1} = 1$, then $d_{ij} \in \{0,2\}$. If $q_{j1} \in \{2,3\}$, then $d_{ij} \in \{0,1\}$. This means that only type 1 components may be destroyed, but if they are not, they must be included in the output batch. Material components are functionally destroyed when they permanently become part of an assembly.

A workstation controller communicates with other elements of the framework through ports. Communication within the framework refers to the transfer of information or material from one control element to another. A workstation controller has two input ports and nq+1 output ports. The controller has one input port and one output port dedicated to communication with the shop controller. The workstation controller has one output port for each queue in the workstation and one input port that receives communication from the queues.

The workstation controller sleeps until it receives components from the shop controller or from one of its processes. When a component is received from the shop controller, the workstation follows a short sequence of steps to process the component.

1. If the component is type 1, 2 or 3, the workstation entry time attribute is updated.
2. If the component is type 4, the batch is split up. The type 1 and type 2 components are re-batched in pairs.
3. The type 1 and 2 components are routed to the port corresponding to a queue with the same component type and class. Type 4 components are routed to the type 1 queue with the same component class.
4. The class attributes of the type 3 components is compared to the class attributes for each of the type 3 queues. If there is a match, the component is routed to the appropriate queue. If there is no match, the component is routed back to the shop controller for return to its point of origin as recorded in its attributes.
5. The controller finds the first process combination $\mathbf{x}_i$ that matches the state of $\mathbf{Q}$.
6. If the controller finds a match, say $\mathbf{x}_i$, it signals $\mathbf{q}_j$ to release x_{ij} components to the queue input port. Any type 4 components that were stored in type 1 queues are split and the type 2 components that were part of the batch are routed to the shop controller for return to their point of origin. The remaining components are then formed into a new batch. The batch attributes are assigned based on the attributes of the type 1 component in the batch with the lowest component class number. The batch is deactivated for a period of simulation time equal to the setup and processing times specified by the batch attributes.
7. When the batch is reactivated at the end of processing it is split up and the disposition of each component is determined from the post-process disposition instructions. The component class, c_2, of each type 1 component is changed to match the class attribute of the type 2 component in the batch. Components to be destroyed are disposed from the model. Components to be released are routed to the shop controller for transfer back to their point of origin.
8. The remaining components are formed into a batch and routed to the shop controller for transfer to their next destination.

3.2.4 Shop Controller

The third and highest level of production control is the shop controller. In the same way that the workstation controller coordinates the operation of its queues, the shop controller coordinates the operation of its workstations. It is the responsibility of the shop controller to populate the system with production control and resource permissions at the beginning of each simulation run and to coordinate traffic between workstations to implement a coherent production control policy throughout the system. A shop controller $\mathbf{s}$ has four components:

$$\mathbf{s} = \left[\mathbf{W}, \mathbf{R}, \mathbf{P}, \mathbf{B}\right] \tag{9}$$

$\mathbf{W}$ is a set of nw workstation controllers, $\mathbf{R}$ is a set of nr component generators, $\mathbf{P}$ is a set of np production control rules and B is a set of nc process plans.

$$\mathbf{W} = \left[\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{nw}\right] \tag{10}$$

$\mathbf{w}_i$ is a workstation controller as described in the previous section and nw is the number of workstations in the shop.

$$\mathbf{R} = \left[\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{nr}\right]^T \tag{11}$$

$$\mathbf{r}_i = \left[r_{i1}, r_{i2}, r_{i3}, r_{i4}, r_{i5}\right] \tag{12}$$

r_{i1} and r_{i2} are the component type and class of the components to be generated, r_{i3} is the workstation where the component is to be assigned, r_{i4} is the number of components to be generated and r_{i5} is the simulation time at which they are to be generated.

$$\mathbf{P} = \left[p_1, p_2, \ldots, p_{np}\right]^T \tag{13}$$

np is the number of material component classes processed by the system, and p_i is the production control policy for material component class i. If p_i = 0, the control policy is push. If p_i = 1, the control policy is pull. If p_i = 2, the component is the push-pull interface (PPI), or control point, for the product.

$$\mathbf{B} = \left[\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{np}\right]^T \tag{14}$$

$$\mathbf{b}_i = \left[b_{i1}, b_{i2}, b_{i3}\right] \tag{15}$$

$\mathbf{b}_i$ is the process plan for material class i, b_{i1} is the number of the workstation controller where material component class i is processed, and b_{i2} and b_{i3} are the setup and processing times, respectively, for processing material class i at workstation b_{i1}. If $b_{i1} = nw + 1$, the component is a finished product and will be routed out of the system.

The shop controller communicates with other elements in the framework through ports. The shop controller has a pair of ports for input and output with the world outside the system. Like the workstation controller, the shop controller has one output port for each of the workstation controllers in the shop and one input port to receive communication back from them. The shop controller operates in two distinct modes. At the beginning of a simulation run, it creates, initializes and distributes components into the system to establish the initial condition of the system. There-after it coordinates communication between the system and the world and between the workstations.

A shop controller receives types 1, 2 and 3 components from the world. Type 1 components represent raw materials, type 2 components represent finished goods orders to be filled and type 3 components are system resources to be added to the system. When a shop controller receives a component from the world, it follows a short set of instructions, depending on the component type.

For a type 1 component of class *i*, the shop controller follows these instructions:

1. The controller sets c_7, the shop entry time, to the current simulation time and sets c_4, c_8, c_9 and c_{10}, the destination workstation, imminent setup time and imminent processing time to b_{i1}, b_{i2}, b_{i3} and $b_{i2}+b_{i3}$ respectively.

3. It would also set c_{12}, c_{13} and c_{14}, the process time remaining, processes remaining and static slack time, but these attributes require bill of material information that is not currently included in the definition of the framework.

4. The controller then routes the component to c_4, the destination workstation.

For a type 2 component of class *i*, the shop controller sets c_4, the destination workstation, to b_{i1}, the PPI workstation for product *i*. In some systems and under some production control policies, there may be more than one PPI workstation. If this were the case, bill of material information would be required to determine the number and destinations of duplicate production permission components. As indicated above, that information is not yet included in this framework.

For type 3 components, the controller routes them directly to workstation c_4.

A shop controller receives type 4 components from its workstation controllers. The controller splits the batch up, then processes the constituent components individually before reforming the batch.

For type 1 components of class *i*, the controller sets c_4, c_8, c_9 and c_{10}, the destination workstation, imminent setup time and imminent processing time to b_{i1}, b_{i2}, b_{i3} and $b_{i2}+b_{i3}$ respectively. The components are then set aside until the rest of the components in the batch are finished processing and a batch is reformed.

For type 2 components, the controller checks the production control policy for the class. For a type 2 component of class *i*, the controller checks p_i. If $p_i = 0$, push, the component is immediately routed to workstation c_4. If $p_i = 1$, pull, the component is set aside. If $p_i = 2$, PPI, the component is disposed from the simulation.

Type 3 components, like type 1, are simply set aside until the rest of the components in the batch are finished.

When all of the components in a batch have been processed by the shop controller, the batch is reformed, minus any type 2 components that were routed elsewhere or disposed. The resulting type 4 component is then routed to workstation c_4. If $c_4 > nw$, the controller routes the component to the world output port.

4 APPLICATION: AN EXAMPLE

The effectiveness of the production control framework is best illustrated through an example. It is applied here to a three stage flow shop producing a single product.

In the first stage, two subassemblies are processed by two different machines. In the second stage, the subassemblies are combined into a finished product by one of two identical machines. In the final stage, the finished products are packaged for shipping before they leave the system. The system is undergoing lean transformation and the system manager wants to simulate the effects of moving the control point from the first stage, where the legacy MRP system currently controls the system, to the third stage, where customer demand can directly drive production. Figure 2 illustrates the manufacturing system.

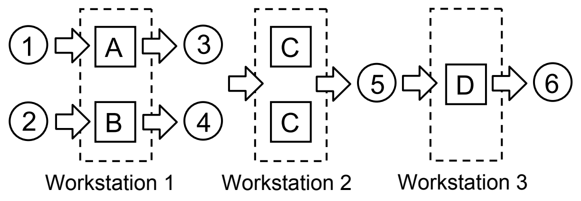

Figure 2: Flow Shop Example

4.1 System Definition

Applying the framework, a shop controller **s** is

$$\mathbf{s} = \left[\mathbf{W} \mid \mathbf{R} \mid \mathbf{P} \mid \mathbf{B}\right] \tag{16}$$

The shop controller for this case is defined as

$$\mathbf{s} = \left[\begin{array}{c|ccccc|c|ccc}
\mathbf{w}_1 & 2 & 3 & 1 & 1 & 0 & 0 & 1 & b_{12} & b_{13} \\
\mathbf{w}_2 & 2 & 4 & 1 & 1 & 0 & 0 & 1 & b_{22} & b_{23} \\
\mathbf{w}_3 & 2 & 5 & 2 & 2 & 0 & 0 & 2 & b_{32} & b_{33} \\
 & 2 & 6 & 3 & 1 & 0 & 0 & 2 & b_{42} & b_{43} \\
 & 3 & 1 & 1 & 1 & 0 & 0 & 3 & b_{52} & b_{53} \\
 & 3 & 2 & 1 & 1 & 0 & 0 & 4 & b_{62} & b_{63} \\
 & 3 & 3 & 2 & 2 & 0 & 0 & & & \\
 & 3 & 4 & 3 & 1 & 0 & 0 & & &
\end{array}\right] \tag{17}$$

The system has three workstations, so **W** contains three rows.

There are eight component generators in the shop controller, so there are eight rows in **R**. Looking at the first column, there are four each of production permission (type 2) and resource permission (type 3) generators. Column two indicates that each type 2 component is assigned to one of the four classes of subassemblies, and each of the type 3 components is assigned to one of the four classes of machines in the system. This is consistent with the fact that each machine in the system processes only one subassembly. Columns three and four show that the two types of components are assigned to the three workstations in equal numbers. According to column five, all of the components are introduced to the simulation at time 0.

There are six material component classes in the system, so **P** has six rows. Since the shop is initially using a purely push-type production control policy, all of the entries are zeros. This means that when batches visit the shop controller after processing, the type 2 components will be released to return to their original workstation.

Again, there are six material component classes in the system, so **B** has six rows. Column one indicates the workstation where the component is processed. Note that **B** indicates material class 6 is processed in workstation 4. There is no workstation 4 in the system, so this simply indicates that the shop controller will route these components to the world output port. Columns two and three contain the setup and processing times for each component class. They remain undefined since these parameters are not critical to this illustration.

With the shop controller defined, the workstation controllers can now be designed in accordance with the framework.

$$\mathbf{w} = [\mathbf{Q} \mid \mathbf{X} \mid \mathbf{D}] \tag{18}$$

The workstation controllers and queue controllers are generated in parallel to ensure row continuity. The workstation controller for workstation 1 is defined as

$$\mathbf{w}_1 = \left[\begin{array}{cccc|cc|cc} 1 & 1 & 5 & 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 5 & 0 & 0 & 1 & 0 & 0 \\ 2 & 3 & 5 & 0 & 1 & 0 & 0 & 0 \\ 2 & 4 & 5 & 0 & 0 & 1 & 0 & 0 \\ 3 & 1 & 5 & 0 & 1 & 0 & 1 & 0 \\ 3 & 2 & 5 & 0 & 0 & 1 & 0 & 1 \end{array}\right] \tag{19}$$

For workstation 1, there are six queues; one for each of the two material classes processed there, one for each of the production permissions for the processes and one for each of the two different processing machines in the workstation. Columns three and four indicate that each of the queues are ordered by ascending entry time, or first-come-first-served.

Workstation 1 produces two subassemblies, so there must be at least two feasible process combinations. There is one column in **X** for each feasible process combination. In this case then, there are exactly two combinations because there are two columns in **X**. The first combination, column one of **X**, indicates that one class 1 raw material component and one class 1 resource permission component are required to produce one class three subassembly. The second column defines the requirements to produce a class 4 subassembly.

Since there are two process combinations in the workstation, so there must be two dispositions and therefore two columns in **D**. The dispositions indicate that the resource permission components remain in the workstation after processing, but the other components are batched and routed to the shop controller.

The remaining two workstation controllers are defined

$$\mathbf{w}_2 = \left[\begin{array}{cccc|c|c} 1 & 3 & 5 & 0 & 1 & 0 \\ 1 & 4 & 5 & 0 & 1 & 2 \\ 2 & 5 & 5 & 0 & 1 & 0 \\ 3 & 3 & 5 & 0 & 1 & 1 \end{array}\right] \tag{20}$$

$$\mathbf{w}_3 = \left[\begin{array}{cccc|c|c} 1 & 5 & 5 & 0 & 1 & 0 \\ 2 & 6 & 5 & 0 & 1 & 0 \\ 3 & 4 & 5 & 0 & 1 & 1 \end{array}\right] \tag{21}$$

In this initial condition, production control in the system is governed by the release of raw materials. The push-pull interface is located outside the system.

Transforming the system to a pull-type production control policy is a simple parametric change. To move the push-pull interface, change **P** in the shop controller to

$$\mathbf{P} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2]^T \tag{22}$$

Material component class 6 is now the PPI, and the shop has been transformed to pull production control. When batches visit the shop controller after processing, the type 2 components will now continue to the next workstation with the batch.

This simple flow shop model illustrates the effectiveness of the proposed production control framework to model production control policy parametrically.

5 SUMMARY AND FUTURE WORK

This paper described the design of a framework with which to implement production control in a simulation model of a manufacturing system. It does so by differentiating the flow of material from the flow of information in the system, but uses the same techniques to control both. An information model for the system was defined and an example was presented to demonstrate its use.

Current work focuses on realizing the framework by building a library of production control modules with which to test its effect on simulation model adaptability. The simple manufacturing system described in this paper, and others, will be modeled with and without the framework, and the adaptability measured. The modules will incorporate bill of materials information.

Future plans include further refining the controller modules by increasing the framework's ability to represent material handling elements and to implement variable process batch sizes. Plans also include evaluation of automated generation of simulation models from the framework proscribed by this paper.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation through Collaborative Investigations in Industrial and Systems Engineering: North Carolina A&T State University, University of Maryland and Georgia Institute of Technology. This work was also supported by Northrop Grumman Corporation through the Institute for Systems Research Industrial Fellowship Program.

REFERENCES

Webster's Ninth New Collegiate Dictionary. 1988. Springfield, MA: Merriam-Webster.

Banks, J., and R. Gibson. 1998. Simulation evolution. *IIE Solutions* November: 26-29.

Herrmann, J. W., E. Lin, B. Ram, and S. Sarin. 2000. Adaptable simulation models for manufacturing. In *Proceedings of the 10th International Conference on Flexible Automation and Intelligent Manufacturing, Volume 2*. College Park, MD. University of Maryland, Department of Mechanical Engineering.

Hopp, W. J., and M. L. Spearman. 2000. *Factory physics: Foundations of manufacturing management, 2nd edition*. Boston: Irwin/McGraw-Hill.

McKay, K. N., and J. B. Moore. 1991. *Intelligent manufacturing management program state of the art scheduling survey*. Arlington, TX: Consortium for Advanced Manufacturing International.

Law, A. M., and W. D. Kelton. 1991. *Simulation modeling & analysis, 2nd edition*. New York: McGraw-Hill.

Panwalker, S. S., and W. Iskander. 1977. A survey of scheduling rules. *Operations Research* 25 (1): 45-62.

Rother, M. and J. Shook. 1999. *Learning to see: Value stream mapping to create value and eliminate muda, version 1.2*. Brookline, MA: Lean Enterprise Institute.

Smith, J. S., W. C. Hoberecht and S. B. Joshi. 1996. A shop floor control architecture for computer-integrated manufacturing. *IIE Transactions* 28: 783-794.

Vieira, G. E. 1998. Evaluating control architectures for flexible manufacturing systems from a response time perspective. Doctoral dissertation proposal. Department of Mechanical Engineering, University of Maryand, College Park, Maryland.

Vollmann, T. E., W. L. Berry and D. C. Whybark. 1997. *Manufacturing planning and control systems, 4th edition*. New York: Irwin/McGraw Hill.

Womack, J. P. and D. T. Jones. 1996. *Lean thinking: Banish waste and create wealth in your company*. New York: Simon & Schuster.

AUTHOR BIOGRAPHIES

SEAN M. GAHAGAN is a graduate student and research assistant in the Department of Mechanical Engineering and Institute for Systems Research at the University of Maryland. He is a Northrop-Grumman / Institute for Systems Research Fellow and an American Society of Naval Engineers Scholar. He received his B.S.M.E. from the University of North Carolina-Charlotte in 1995. He is a member of ASME, SME, TBΠ and ASNE. His email address is `<sgahagan@isr.umd.edu>`.

DR. JEFFREY W. HERRMANN is an associate professor at the University of Maryland, where he holds a joint appointment with the Department of Mechanical Engineering and the Institute for Systems Research. He is the director of the Computer-Integrated Manufacturing Laboratory. He received his B.S. in applied mathematics from Georgia Institute of Technology and his Ph.D. in industrial and systems engineering from the University of Florida. His research interests include process planning, production scheduling, manufacturability evaluation, and manufacturing facility design. He is currently investigating the design and control of manufacturing systems and the integration of product design and manufacturing system design. He is a member of INFORMS and ASME. His email and web addresses are `<jwh2@eng.umd.edu>` and `<www.isr.umd.edu/~jwh2/jwh2.html>`.