Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

FINDING THE OPTIMAL PRODUCTION CONTROL POLICY USING THE PRODUCTION CONTROL FRAMEWORK

Sean M. Gahagan

Northrop Grumman Corporation
Electronic Systems Sector
Baltimore, MD 21240, U.S.A.

Jeffrey W. Herrmann

Institute for Systems Research
University of Maryland
College Park, MD, U.S.A.

ABSTRACT

The production control policy affects the performance of a manufacturing system. Evaluating a production control policy usually requires simulation modeling due to the complex interactions that occur. This paper introduces a technique that optimizes production control of single product flow shops under hybrid production control by using the Production Control Framework. This simulation modeling template is designed to explore the production control domain. The paper demonstrates how this template can be used in conjunction with existing simulation optimization software to find an optimal production control policy. The decision variables are location of the push-pull interface and the number of kanbans at each workstation. The objectives include improving customer service and reducing work-in-process inventory.

1 INTRODUCTION

Production control describes how the flow of material is regulated in a manufacturing system. There are two fundamental production control policies, push and pull, that can be combined in many ways. A push production control policy gives each workstation permission to process material based solely on the availability of the material. A pull production control policy requires a workstation to have, in addition to material, a signal from a downstream workstation that more processed material is required. A hybrid production control policy is one in which there are multiple workstations, some operating with a push policy and some operating with a pull policy.

Pull production control is often implemented using kanban systems. Pull production control is closely associated with the principles of just-in-time (JIT) and lean manufacturing (Slack, 1997). See also Hopp and Spearman (1996).

There are many variations on a basic kanban system. Berkley (1992) proposes a classification scheme for kanban systems. He classifies them based on blocking mechanism, number of cards and withdrawal strategy. Huang and Kusiak (1996) provide an overview of kanban systems and their variations. Buzacott and Shanthikumar (1993) analyze a generalized production authorization system that includes a variety of traditional schemes as special cases.

Hopp and Spearman (1996) describe a variation of kanban that uses elements of both push and pull called CONWIP, meaning constant work in process. In a CONWIP system, the removal of a finished product from the final inventory authorizes, if the work in process is less than the threshold, the creation of a new part at the start of the system instead of at the previous station. Such a system is easier to implement than kanban and offers greater flexibility with respect to variations in demand.

Push systems and pull systems illustrate a tradeoff between inventory levels and cycle time. Although kanban and pull systems effectively control WIP, push systems can provide greater throughput (Amin and Altiok, 1997).

Hopp and Spearman (1996) describe the *push pull interface* as the point in the manufacturing system at which upstream pull production control meets downstream push control. (This point is located at the exit of the input buffer within the interface workstation.) Production is matched to demand at the push pull interface. Every system, they argued, has such an interface. In a traditional push system, the push pull interface is located in a vendor's warehouse where purchasing matches customer demand to material requirements. In a pull system, the push pull interface is on a shelf where the customer withdraws a product directly. They contend that placing the interface at some intermediate point offers the advantages of both controlled WIP and product variety.

There are many good reasons to implement pull production control schemes. However, there have been few direct comparisons of push and pull schemes. Hopp and Spearman (1996) state that (compared to push production control) pull mechanisms can reduce the amount of inventory needed to achieve a specified throughput and that pull mechanisms are more robust (small changes have less impact on overall system performance). These results are based on analysis of open and closed queueing networks. Statements regarding more general production control schemes do not exist, to our knowledge. The problem of comparing push and pull production systems is compounded by the fact that while formulas do exist to calculate a reasonable number of kanbans in a pull system (Vollmann, Berry and Whybark, 1997; Shingo, 1989), there are no general, closed-form solutions to determine the system behavior as a function of the number of kanbans in the system for a multiple stage pull system.

Thus, discrete-event simulation is an important tool for evaluating different production control policies. Moreover, finding a production control policy that achieves the best tradeoff between customer service, work-in-process inventory, and other performance measures is a difficult task. To address this problem, this paper introduces a technique that optimizes production control of single product flow shops under hybrid production control by using the Production Control Framework. This simulation modeling template is designed to explore the production control domain. The paper demonstrates how this template can be used in conjunction with existing simulation optimization software to find an optimal production control policy. The decision variables are location of the push-pull interface and the number of kanbans at each workstation. The objectives include improving customer service and reducing work-in-process inventory.

The remainder of this paper is organized as follows: Section 2 describes the Production Control Framework. Section 3 discusses the problem setting. Section 4 describes the experiments that we conducted. Section 5 presents the results, and Section 6 concludes the paper.

2 PRODUCTION CONTROL FRAMEWORK

To facilitate the study of production control in a simulation environment, we developed the Production Control Framework (PCF), a set of modeling elements that can be used to represent a wide variety of production control configurations in an easily modified, numerical format (Gahagan and Herrmann, 2001). Push production control is relatively simple to model because a workstation is self-contained, needing no knowledge of the rest of the manufacturing system. Pull production control is more difficult to model because each work station must communicate with other workstations. By using a hierarchical framework, the PCF provides a structure for that communication. We used the PCF as the basis for the development of predefined modeling objects in a popular simulation application.

2.1.1 Overview

The PCF uses a three-layer, hierarchical framework to represent a manufacturing system. The structure used in the PCF 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.

At the top of the hierarchy is the Shop, which coordinates communication of material and information. Each shop has one or more Workstations that coordinate the processing of components. Each workstation has one or more Queues that store and order components until they are ready to be processed. Figure 1 illustrates the PCF model.

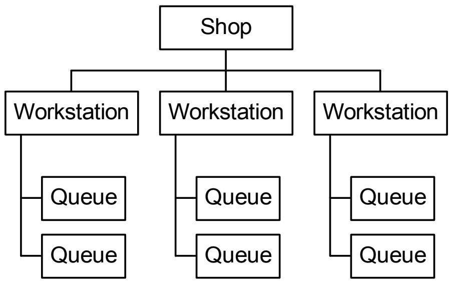


Figure 1: The PCF Model

2.1.2 Components

Traditionally, the term *component* refers only to physical parts. In the PCF it is used to describe material, demand, and resources. The framework uses four distinct component types:

Type 1: *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: *Demand 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 Components*, like demand, are signals transmitted through the system, but unlike demand, 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* are collections of Type 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 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:

$$\mathbf{c} = [c_1, c_2, \ldots, c_{nc}]^T \qquad (1.)$$

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
Demand	2	6
Resource	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 and workstation) represent the address of the component's next destination in terms of the production control framework. The destination addresses of material components and batch 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: Demand 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 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 rest of the batch attributes, listed in Table 6, record the number of each type of component in the batch. The batch attributes are used for production control bookkeeping when the batch is split up and subsequently reassembled at each control level.

2.1.3 Queue

A queue is a collection of similar objects, ordered according to some queue discipline (Law and Kelton, 2000). The simplest queue disciplines are based on the value of an object attribute and are ordered in either ascending or descending order. A PCF queue **q** has two parameters:

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

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

2.1.4 Workstation

The second level of the PCF hierarchy is the Workstation. A Workstation is a set of system resources and associated queues. This controller is responsible for coordinating two or more Queues to complete processes. A process is a task that requires time, material and system resources to complete. A Workstation **w** has four components:

$$\mathbf{w} = [\mathbf{C}, \mathbf{Q}, \mathbf{X}, \mathbf{D}] \qquad (3.)$$

C is a set of *nq* queue controller constituents (components), **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 *nx* post-process dispositions.

$$\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{nq}]^T \qquad (4.)$$

$$\mathbf{c}_i = [\mathbf{c}_{i1}, \mathbf{c}_{i2}] \qquad (5.)$$

c_{i1} and c_{i2} are the type and class of the components to be stored in queue *i*, respectively.

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{nq}]^T \qquad (6.)$$

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

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{nx}] \qquad (7.)$$

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{i,nq}, t_{is}, t_{ip}]^T \qquad (8.)$$

x_{ij} is the number of components from q_j necessary to carry out process *i*, *nx* is the number of processes that can be carried out by the workstation, and t_{is} and t_{ip} are the setup and processing times for process *i*, respectively.

$$\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{nx}] \qquad (9.)$$

$$\mathbf{d}_i = [d_{i1}, d_{i2}, \ldots, d_{i,nq}]^T \qquad (10.)$$

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 released to return to its point of origin (this does not apply to material components). If $d_{ij} = 2$, the component is to be subsumed into the resulting product (this applies only to material components). Material components are subsumed when they permanently become part of an assembly.

2.1.5 Shop

The third and highest level of the PCF 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 demand and resource components 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 **s** has four components:

$$\mathbf{s} = [\mathbf{W}, \mathbf{R}, \mathbf{P}, \mathbf{B}] \tag{11.}$$

W is a set of *nw* workstation controllers, **R** is a set of *nr* component generators, **P** is a set of *np* production control rules and **B** is a set of *np* process plans.

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{nw}] \tag{12.}$$

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

$$\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{nr}]^T \tag{13.}$$

$$\mathbf{r}_i = [r_{i1}, r_{i2}, r_{i3}, r_{i4}] \tag{14.}$$

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, and r_{i4} is the number of components to be generated at the beginning of each simulation run.

$$\mathbf{P} = [p_1, p_2, \ldots, p_{np}]^T \tag{15.}$$

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. If $p_i = 3$, the component is the first in a CONWIP loop. If $p_i = 4$, the component is the last in a CONWIP loop.

$$\mathbf{B} = [b_1, b_2, \ldots, b_{np}]^T \tag{16.}$$

b_i is the number of the workstation controller where material component class *i* is processed. If $b_i = nw + 1$, the component is a finished product and will be routed out of the system.

2.2 Hybrid Production Control Domain

In this paper, we explore the hybrid production control domain, a sub-set of the full PCF domain. Using PCF nomenclature, we can define production control domains explicitly. Once a PCF based model is defined, one can describe its production control policy using only **P** from the Shop element. For such a model **P** is

$$\mathbf{P} = [p_1, p_2, \ldots, p_x, \ldots, p_{np}]^T \tag{17.}$$

we describe the hybrid production control domain as

$$p_1 = 1 \tag{18.}$$

$$p_x = 2 \tag{19.}$$

$$p_{np} \in \{1,2\} \tag{20.}$$

$$p_i = 1 \text{ for } 1 < i < x - 1 \tag{21.}$$

$$p_i = 0 \text{ for } x + 1 < i < np \tag{22.}$$

Thus, for a system with *np* material classes, there are *np-1* variations of hybrid production control.

As Hopp and Spearman (1996) suggest, both traditional push production control and kanban production control can be shown to be special cases within the hybrid production control domain. For push production control, $x = 2$. For kanban production control, $x = np - 1$.

3 PROBLEM SETTING

In this paper, we study the effect of different hybrid production policies on the performance of a four-stage, single-product flow line. The flow line is shown in Figure 2, below.

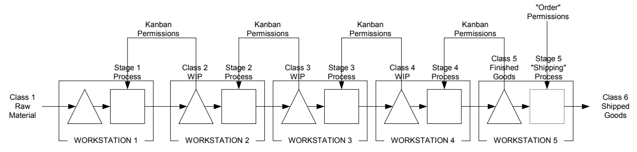


Figure 2: Four-Stage Flow Line

3.1 Definition

Using the PCF, a four-stage, single product flow line $\mathbf{s_4}$ can be expressed as

$$s_4 = \left[\begin{array}{c|cccc|c|c} w_1 & 2 & 2 & 1 & 5 & 0 & 1 \\ w_2 & 2 & 3 & 2 & 5 & 0 & 2 \\ w_3 & 2 & 4 & 3 & 5 & 0 & 3 \\ w_4 & 2 & 5 & 4 & 5 & 0 & 4 \\ w_5 & 2 & 6 & 5 & 5 & 2 & 5 \\ & 3 & 1 & 1 & 1 & & \\ & 3 & 2 & 2 & 1 & & \\ & 3 & 3 & 3 & 1 & & \\ & 3 & 4 & 4 & 1 & & \end{array}\right] \quad (23.)$$

where

$$w_i = \left[\begin{array}{cc|c|c|c} 1 & i & q_{i1} & 1 & 0 \\ 2 & i+1 & q_{i2} & 1 & 0 \\ 3 & i & q_{i3} & 1 & 1 \\ & & & 0 & \\ & & & Logn(1.0,0.1) & \end{array}\right] \quad (24.)$$

for $i \in \{1,2,3,4\}$,

$$w_5 = \left[\begin{array}{cc|c|c|c} 1 & 5 & q_{51} & 1 & 0 \\ 2 & 6 & q_{52} & 1 & 0 \\ & & & 0 & \\ & & & 0 & \end{array}\right] \quad (25.)$$

and

$$\mathbf{q}_{ij} = [5 \quad 0] \ \forall \ i,j \quad (26.)$$

Five workstations are required to model a four stage system. One workstation is required to model each stage and their upstream buffers. A fifth workstation is needed to provide a downstream buffer for the fourth stage. The process in the fifth workstation is defined as requiring zero time. As a default, the WIP of each product in the system has been set to five. Of course, this applies only to components controlled on a pull basis. All of the queues are controlled on a first-in-first-out policy. The system is configured for hybrid production control, specifically kanban production control. Since this system has five workstations, it can be used to model five different hybrid production control policies.

3.2 Model

We modeled the example system using a PCF-based template developed in Arena (Kelton, Sadowski and Sturrock, 2004). Figure 2 shows the user view of the example model.

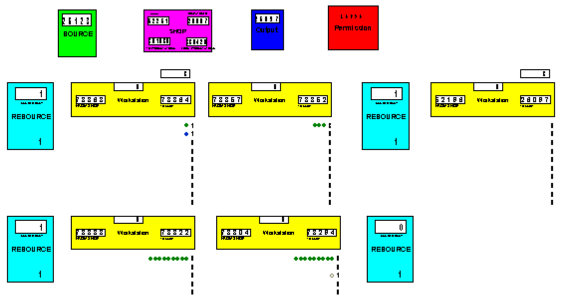


Figure 2: PCF Arena Model

The PCF elements alone are not sufficient to build a functioning model. Component entities must be created and destroyed for the model to function. The PCF template provides modules for the creation of material, permission and resource components. It also provides a sink module to destroy components that have completed processing.

By using the Arena platform, we are able to apply the built-in optimization engine, Optquest, to PCF models to find optimal production control policy and WIP levels.

The PCF template and the models used in this paper are available for review and download on our web site: <`http://www.isr.umd.edu/Labs/CIM/projects/cire/adaptable.html`>.

4 EXPERIMENTS

In this paper we perform a series of experiments using optimization software to demonstrate how the PCF template can be used to find the optimal production control and WIP levels for a generic flow line.

We chose to repeat a subset of the experiments Gaury et al. (2001) performed. Their experiments explored what they called Customized Pull Systems where every station could feed pull signals back to every other station in the system. This extremely flexible scheme is impractical and not found in practice. The scope of this paper is greatly reduced by our definition of the hybrid production control domain in which each station communicates only with the station immediately upstream.

4.1 Experimental Factors

Four experimental design factors were chosen to study the performance of the system. Gaury et al. defined a set of 12 process, demand, and performance factors. Our implementation of the PCF template is somewhat more limited

in what it can express, so our experiments consider the following four factors (summarized in Table 7):
Line Imbalance: A balanced line is one in which all of the stage have the same production rate. An unbalanced line has workstations with unequal production rates. The Degree of Imbalance (DI) characterizes this factor. Meral and Erkip (1991) define DI as

$$DI = \max\{TWC/N-\min(PTi); \max(PTi)-TWC/N\} \quad (27.)$$

where *PTi* is the mean Processing Time at workstation *i* in an *N*-station line, and *TWC*/*N* is the mean processing time at a workstation on the balanced *N*-station line. It was set to either 0, completely balanced, or 0.5, imbalanced.

Imbalance Pattern: Imbalanced lines can have the bottleneck at the last stage (a *funnel* pattern) or at the first stage (a reverse funnel pattern).

Processing Time Variability: The variability of processing time, which has a strong effect on system performance, was set to either 0.1 or 0.5.

Demand Rate / Capacity: The rate at which orders arrive, relative to the capacity of the system, was set to either 0.8 or 0.9.

The use of the PCF template allows us to use off-the-shelf optimization software to find the optimal production control and WIP levels for each experiment. Since this limited system has five possible production control configurations, we find the optimal WIP for each case in order to illustrate the difference this factor has on performance.

Table 7: Experimental Factors (Gaury et al., 2001)

Factor	Level		Letter
	+	-	
Line Imbalance	0	0.5	A
Imbalance Pattern	Funnel	Reverse Funnel	B
Processing Time CV	0.1	0.5	C
Demand Rate / Capacity	0.8	0.9	D

4.2 Experimental Design

A full factorial analysis was performed to examine this set of design factors. (Note that, if the line is balanced, the imbalance pattern is irrelevant.) Table 8, below, details the experimental design.

Each experiment was conducted with the push pull interface in each of the five possible configurations for a total of 60 experiments.

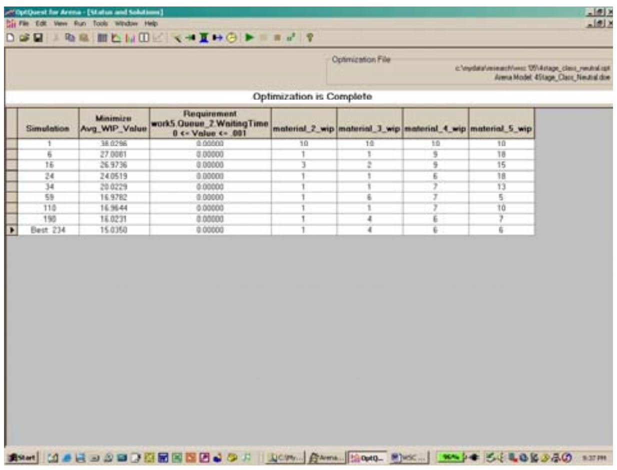
Each trial was run for a single replication of 24,000 time units with a warm-up period of 1,000 units.

Table 8: Experimental Design

Trial	A	B	C	D
1	+	n.a.	+	+
2	+	n.a.	+	-
3	+	n.a.	-	+
4	+	n.a.	-	-
5	-	+	+	+
6	-	+	+	-
7	-	+	-	+
8	-	+	-	-
9	-	-	+	+
10	-	-	+	-
11	-	-	-	+
12	-	-	-	-

4.3 Optimization Setup

Each trial configuration was optimized using Optquest, an optimization routine that is sold with Arena. The objective function to be minimized was the average total number of parts (material components) waiting in the system. For optimization purposes, the control parameters were the number of permission components issued to each workstation at the beginning of the run. For pull workstations, these permissions become the kanbans that authorize production at that station. Otherwise, the permissions are not used. These controls were limited to integer values from 1 to 50, with a recommended value of 10. A customer service requirement was imposed: the average waiting time for an incoming order at the push pull interface must be less than or equal to 0.001 time units. The optimization for each trial was set to run for 20 minutes. In a typical run, this resulted in over 200 permutations. Figure 3 shows the Optquest user interface. The parameter values used in the experiments are included in Appendix A.


OptQuest for Arena - [Status and Solutions]
File Edit View Run Tools Window Help

Optimization File
c:\mydata\research\wsc '05\4stage_class_neutral.opt
Arena Model: 4Stage_Class_Neutral.doe

Optimization is Complete

Simulation	Minimize Avg_WIP_Value	Requirement work5.Queue_2.WaitingTime 0 <= Value <= .001	material_2_wip	material_3_wip	material_4_wip	material_5_wip
1	38.0296	0.00000	10	10	10	10
6	27.0081	0.00000	1	1	9	18
16	26.9736	0.00000	3	2	9	15
24	24.0519	0.00000	1	1	6	18
34	20.0229	0.00000	1	1	7	13
59	16.9782	0.00000	1	6	7	5
110	16.9644	0.00000	1	1	7	10
190	16.0231	0.00000	1	4	6	7
Best: 234	15.0350	0.00000	1	4	6	6

Figure 9: Optquest User Interface

5 RESULTS

The experimental results show, consistent with our expectations, that increasing demand and variability requires more inventory to maintain good customer service. Full results are reported in Appendix B. In this table, cells are marked "N.F.F." ("no feasible found") when the optimization routine could not find a solution that satisfied the customer service requirement.

When the push pull interface is Stage 1, the system is pure push system, and the system performance doesn't depend upon the control parameters. Interestingly, in three of the trials (trials 4, 7, and 10), the push system cannot achieve the customer service requirement. In all of the pure pull systems (with the push pull interface at Stage 5), the system can achieve the customer service requirement, though that will require more WIP (the number of kanbans in the system) when demand or process variability is high.

When the push pull interface is at Stages 2, 3, or 4, the system can achieve the customer service requirement sometimes, particularly when demand is low or process variability is low.

Note that the customer service requirement may be too restrictive, since it was hard to find feasible solutions in some cases. Additional experiments are planned to further understand how changing the customer service requirements affect the performance of different production control policies.

6 CONCLUSIONS

In this paper we explored the potential of the PCF to facilitate the use of simulation optimization to find the optimal production control configuration for a generic flow line.

The trials were carried out very quickly due to the fact that there was little or no programming required to reconfigure the model and optimization engine for different production control configurations. Some coding was necessary to measure the objective function. However, once implemented, a single model was capable of emulating the entire production control domain for a given generic flow line.

Optquest proved to be a useful tool, but it was limited in its expression of constraints. No constraint could relate one input parameter to another. Thus, the ordinal constraints of the hybrid production control domain could not be fully automated. This was, however, only a small inconvenience.

The PCF template successfully demonstrated that it could be combined with optimization to find the optimal production control configuration for a generic flow line. The results of experiments on a simple flow line were consistent with studies on similar systems using traditional modeling elements.

Future research will consider larger generic flow lines with non-monotonic processing time imbalances. It will pursue the incorporation of more intrinsic objective measurement. It will also attempt to overcome the limitations of the simulation optimization tool.

APPENDIX A: EXPERIMENTAL VALUES

Trial	1	2	3	4	5	6	7	8	9	10	11	12
Demand Level	Low Demand (0.8)						High Demand (0.9)					
Process Variability	Low Process Variability (0.1)			High Process Variability (0.5)			Low Process Variability (0.1)			High Process Variability (0.5)		
Line Imbalance	Balanced	Funnel	Reverse Funnel	Balanced	Funnel	Reverse Funnel	Balanced	Funnel	Reverse Funnel	Balanced	Funnel	Reverse Funnel
Stage 1 process time	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5
Stage 1 variability	0.1	0.15	0.05	0.5	0.75	0.25	0.1	0.15	0.05	0.5	0.75	0.25
Stage 2 process time	1	1.17	0.83	1	1.17	0.83	1	1.17	0.83	1	1.17	0.83
Stage 2 variability	0.1	0.117	0.083	0.5	0.585	0.415	0.1	0.117	0.083	0.5	0.585	0.415
Stage 3 process time	1	0.83	1.17	1	0.83	1.17	1	0.83	1.17	1	0.83	1.17
Stage 3 variability	0.1	0.083	0.117	0.5	0.415	0.585	0.1	0.083	0.117	0.5	0.415	0.585
Stage 4 process time	1	0.5	1.5	1	0.5	1.5	1	0.5	1.5	1	0.5	1.5
Stage 4 variability	0.1	0.05	0.15	0.5	0.25	0.75	0.1	0.05	0.15	0.5	0.25	0.75
Order interarrival time	1.25	1.87	1.87	1.25	1.87	1.87	1.11	1.66	1.66	1.11	1.66	1.66
Order variability	0.125	0.187	0.187	0.125	0.187	0.187	0.111	0.166	0.166	0.111	0.166	0.166

APPENDIX B: EXPERIMENTAL RESULTS

Push Pull Interface	Optimal Value of Permissions for each Stage	Trial											
		1	2	3	4	5	6	7	8	9	10	11	12
Stage 2	Stage 1	1	2	1	N.F.F.	N.F.F.	N.F.F.	N.F.F.	2	1	N.F.F.	N.F.F.	N.F.F.
Stage 3	Stage 1	1	1	1		1			2	1			
	Stage 2	1	1	1		6			1	1			
	Total kanbans	2	2	2	N.F.F.	7	N.F.F.	N.F.F.	3	2	N.F.F.	N.F.F.	N.F.F.
Stage 4	Stage 1	1	1	1		1		1	1				
	Stage 2	1	1	1		2		1	1				
	Stage 3	1	1	1		4		2	1				
	Total kanbans	3	3	3	N.F.F.	7	N.F.F.	4	3	N.F.F.	N.F.F.	N.F.F.	N.F.F.
Stage 5 (Pull)	Stage 1	1	1	1	1	1	1	1	1	1	7	3	1
	Stage 2	1	1	1	1	6	4	6	1	4	5	2	1
	Stage 3	1	1	3	1	6	6	6	1	8	5	4	4
	Stage 4	1	1	13	3	2	6	3	1	5	10	4	5
	Total kanbans	4	4	18	6	15	17	16	4	18	27	13	11

REFERENCES

Amin, M. and T. Altiok, 1997. “Control policies for multi-product multi-stage manufacturing systems: an experimental approach”. International Journal of Production Research, 35(1), pp. 201-223.

Berkley, B. J. “A review of the kanban production control research literature”. Production and Operations Managament, 1(4), pp. 393-411. 1992.

Buzacott, John A., and George J. Shanthikumar. Stochastic models of manufacturing systems. Prentice Hall, Englewood Cliffs, NJ. 1993.

Gahagan, S.M., and J.W. Herrmann, “Improving simulation model adaptability with a production control framework”. Proceedings of the Winter Simulation Conference, Arlington, Virginia, December 9-12, 2001.

Gaury, E. G. A., H. Pierreval and J. P. C. Kleijnen, 2000. “An evolutionary approach to select a pull system among kanban, conwip and hybrid”. Journal of Intelligent Manufacturing, Volume 11, pp. 157-167.

Hopp, Wallace J., and Mark L. Spearman. Factory physics. Irwin/McGraw-Hill, Boston, 1996.

Huang, C.-C. and A. Kusiak. “Overview of kanban systems”. International Journal of Computer Integrated Manufacturing, 9(3), pp. 169-189. 1996.

Kelton, W. David, Randall P. Sadowski and David T. Sturrock. Simulation with arena, Third edition. McGraw-Hill, Boston, 2004.

Law, Averill M. and W. David Kelton. Simulation modeling and analysis, Third edition. McGraw-Hill, New York, 2000.

Meral S. and N. Erkip. “Simulation analysis of a JIT production line”. International Journal of Production Economics, Volume 24, pp.147-156. 1991.

Panwalkar, S.S., and Wafik Iskander. “A survey of scheduling rules”. Operations Research, Volume 25, Number 1, January-February, 1977, pp.45-61.

Shingo, Shigeo. A study of the toyota production system from an industrial engineering viewpoint. Productivity Press, Cambridge, MA, 1989.

Slack, Nigel (Ed.). The Blackwell Encyclopedia Dictionary of Operations Management. Blackwell Publishers Ltd., Oxford, UK, 1997.

Smith, J.S., Hoberecht, W.C. and Joshi, S.B. “A shop floor control architecture for computer integrated manufacturing” IIE Transactions, 28(10), pp. 783-794. 1996.

Vieira, Guilherme Ernani. Evaluating control architectures for flexible manufacturing systems from a response time perspective. PhD Dissertation Proposal, University of Maryland, College Park, MD, 1998.

Vollmann, Thomas E., William L. Berry and D. Clay Whybark. Manufacturing planning and control systems, Fourth edition. Irwin/McGraw-Hill, New York, 1997.

AUTHOR BIOGRAPHIES

SEAN M. GAHAGAN is a Principal Industrial Engineer at Northrop Grumman Corporation’s Electronic Systems Sector headquarters in Baltimore, MD. He is also a doctoral candidate at the University of Maryland. He received his B.S.M.E. from the University of North Carolina, Charlotte and his M.S. in Mechanical Engineering from the University of Maryland. He is a member of ASME, IIE, SME and

ΤΒΠ. His e-mail address is <sean.gahagan@ngc.com>.

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 current research interests include the design and control of manufacturing systems, the integration of product design and manufacturing system design, and decision-making systems in product development. He is a member of INFORMS, ASME, IIE, SME, and ASEE. His email and web addresses are <jwh2@umd.edu> and <www.isr.umd.edu/~jwh2/jwh2.html>.