ABSTRACT

Title of Document:	SIMULATION AND OPTIMIZATION OF PRODUCTION CONTROL FOR LEAN MANUFACTURING TRANSITION
	Sean M. Gahagan, Ph.D., 2008
Directed By:	Associate Professor Jeffrey W. Herrmann, Department of Mechanical Engineering

Lean manufacturing is an operations management philosophy that advocates eliminating waste, including work-in-process (WIP) inventory. A common mechanism for controlling WIP is “pull” production control, which limits the amount of WIP at each stage.

The process of transforming a system from push production control to pull is not well understood or studied. This dissertation explores the events of a production control transition, quantifies its costs and develops techniques to minimize them. Simulation models of systems undergoing transition from push to pull are used to study this transient behavior.

The transition of a single stage system is modeled. An objective function is introduced that defines transition cost in terms of the holding cost of orders in backlog and material in inventory. It incorporates two techniques for mitigating cost: temporarily deferring orders and adding extra capacity. It is shown that, except when backlog costs are high, it is better to transform the system quickly. It is also demonstrated that simulation based optimization is a viable tool to find the optimal transition strategy.

Transition of a two-stage system is also modeled. The performance of two simple multi-stage transition strategies is measured. In the first, all of the stages are transformed at the same time. In the second, they are transformed one at a time. It is shown that the latter strategy is superior. Other strategies are also discussed.

A new modeling formalism, the Production Control Framework (PCF), is introduced to facilitate automated searches for transition strategies in more complex systems. It is a hierarchical description of a manufacturing system built on a novel extension of the classic queue server model, which can express production control policy parametrically.

The PCF is implemented in the form of a software template and its utility is shown as it is used to model and then find the optimal production control policy for a five stage system.

This work provides the first practical guidance and insight into the behavior and cost of Lean production control transition, and it lays the groundwork for the development of optimal transition strategies for even the most complex manufacturing systems.

SIMULATION AND OPTIMIZATION OF PRODUCTION CONTROL FOR LEAN MANUFACTURING TRANSITION

By

Sean M. Gahagan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Dedication

This dissertation is dedicated to my son, Owen Gahagan.

Owen,

I hope that you dream big dreams and devote yourself to making them a reality. This dissertation is proof that if you work hard and don’t give up, you can accomplish anything you set your mind to. And maybe, just maybe, it’s proof that your dad isn’t as dumb as he seems.

Acknowledgements

This research was supported financially by my employer, Northrop Grumman Corporation. As a full-time student, they supported my work through the Northrop Grumman ISR Fellowship, and as a full-time employee through their generous and exemplary education reimbursement program.

This work was supported personally by my advisor, Dr. Jeffrey Herrmann. In the nine years (!) I've known him he has been my teacher, my mentor, my advocate and my friend. Many things in my life have changed since I first visited his office in the spring of 1999, but his generosity, intelligence and good nature have remained invaluable constants. Without his patience and encouragement, this would not have been possible. Thank you, Dr. H.

I was supported in every other way during the writing of this dissertation by my wife, Melissa. Research teaches us that every new chapter contains new ideas and a different perspective on the world. I have no doubt that this new chapter of our lives that we are writing together will be the best ever.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

Lean manufacturing is an operations management philosophy focused on reducing waste in a manufacturing system. Lean identifies many different types of waste, among them the waste of overproduction – making products and building inventories for which there is no current demand. Installing pull production control policies is an important part of implementing Lean manufacturing in high-volume, repetitive manufacturing systems. Production control policies, which dictate when manufacturing resources should work, affect important measures of manufacturing system performance, including cycle time and work-in-process inventory. Pull production control policies have been shown to improve manufacturing system performance by linking production control to customer demand. However, transforming a system governed by push to one controlled with pull has not been studied extensively. The different ways to transform a system, the behavior of a system during transformation and the real world costs of Lean transition have never been measured. As a result, risk-averse manufacturers have been slow to adopt Lean practices. Using simulation models of manufacturing systems, one can study the effects of different types of production control rules on performance metrics. One can also study the behavior of a system undergoing Lean transition. In doing so, this research aims to shed light on the transformation process, giving a would-be practitioner tools and techniques for Lean transition and realistic expectations of system performance during the transformation.

1.1 Lean Manufacturing

Lean manufacturing is a western adaptation of the Toyota Production System, developed by the Japanese carmaker and most famously studied (and the term "Lean" coined) in *The Machine That Changed the World* (Womack, 1996). Taiichi Ohno, the engineer commonly credited with development of the Toyota Production System, and therefore Lean, identified seven types of waste: defective products, unnecessary finished products, unnecessary work in process, unnecessary processing, unnecessary movement (of people), unnecessary transportation (of products) and unnecessary delays. Lean focuses on eliminating these wastes from a manufacturing system. In particular, this work is interested in the second and third types – unnecessary finished goods and work in process. The Lean answer to these wastes is to link production at each step in the process with the subsequent process (or the consumer for finished goods). At Toyota, they use kanban (a Japanese word for "shop sign") cards attached to each sub-assembly that are sent back to the producer each time one is used. The cards then become a signal to produce one more. As a result, the number of cards in the system controls the amount of work in process.

1.2 Production Control Policies

Understanding production control policies is critically important to effecting Lean transition. *Push* production control is similar to make-to-order; while *pull* production control is similar to make-to-stock. In practice, many facilities use *hybrid* production control, which combines push and pull policies. Properly implementing Lean manufacturing requires information not only about what the production control policies should be (where to go) but also about the best production control transition strategy (how to get there). The transition strategy defines how the manufacturing system will produce parts during the change from the existing production control scheme to the new one.

Despite the wide acceptance of the fundamental principles of Lean manufacturing, its implementation in American manufacturing enterprises has been slow. Reasons offered to explain this disconnect vary from the technical to the cultural and philosophical. One technical obstacle that must be overcome in order to implement Lean manufacturing more widely is its seeming incompatibility with materials requirements planning (MRP), the most common production management system used in American factories. Although some authors describe ways to use pull production techniques and MRP together, none address the issue of the transition from push to pull in more than anecdotal form. Because this transition has the potential to be very costly, few manufacturers are willing to commit their systems to such a poorly understood change.

1.3 Production Control Transition

The objective of Lean production control policy is to move the interface for customer orders from the beginning of the production line towards the end of the production line. This interface is sometimes called the “push-pull interface” or “inventory/order interface”

(Hopp and Spearman, 2000). This change reduces the average time that customers wait for fulfillment of their orders.

The fundamental element of this transition is the conversion of a single processing stage from push (production based on customer demand) to pull (production based on the status of the downstream operations, using kanban cards as a signaling mechanism). This requires that the stage produce items to which the kanban cards can be attached; however, no customers have requested the items. Therefore, during this transition, the stage experiences a surge in its workload as it attempts to build a kanban inventory while it is also processing regular customer orders. The surge may overwhelm the capacity of the station, resulting in a backlog that would adversely affect customer lead times. To prevent this, I propose two temporary mitigating techniques – adding more resources or deferring some of the customer orders.

1.4 Simulation

Simulation modeling offers a reasonable approach for studying this transition process and a laboratory in which to explore different transition strategies. However, because simulation software (based on simple servers and queues) has been designed to make it easy to model push production control systems, modeling hybrid production control schemes is difficult and modifying them is time-consuming. In order to use simulation to study the design and implementation of production control policies for Lean manufacturing systems, new simulation modeling techniques are needed.

I propose that a new modeling framework is necessary to fully describe a system controlled by Lean production control. Such a framework enables simulation modeling and optimization of both the final state of the system and the trajectory of changes necessary to reach it.

1.5 Motivation

This research is motivated by the view that manufacturing systems research has not yet provided a complete understanding of Lean manufacturing implementation. There is a great deal of writing about how the system should work after the transition from push to pull, but little about the transition process or the associated costs. The primary sources are anecdotal accounts and analytical models that represent only special cases of production control. While these provide some insight, better understanding is needed, especially for managers who are changing production control policies in order to implement Lean manufacturing.

Moreover, this research is motivated by my personal experience as a practitioner of Lean principles in real world manufacturing systems. Lean principles are radically different from the way most factories operate day-to-day. As a result, they are often received with a fair amount of suspicion and mistrust. Many Lean case studies focus on the miraculous turnaround of businesses on the brink of disaster. Businesses in such a situation are much more willing to gamble on radical change. The Lean literature has little to offer a reasonably successful, risk-averse manufacturing enterprise seeking to weigh the costs and benefits of implementing Lean practices. This is especially true when considering how transforming the manufacturing system will disrupt normal operations and how much the transformation itself will cost. How much does Lean cost? This is the question that I ask, and I believe simulation of Lean transition is the answer.

1.6 Objectives

The objective of this dissertation is to use simulation models of systems undergoing changes in production control policies, especially those used for Lean, in order to better understand the mechanisms and costs of those changes, to develop techniques to mitigate those costs and to employ simulation-based optimization to find the least expensive change strategy.

To do so, new simulation modeling techniques will be developed. Discrete event simulation can represent a wide variety of production control techniques, including Lean production control. However, current approaches are not adequate. Simulation methodology and software are designed to exclude transient behavior – the focus of this work. Techniques will be developed to model and measure manufacturing systems in transition.

The costs of production control transition will be clearly defined. Once measurement of systems undergoing transition is possible, those measurements must be related to an objective function that is meaningful to manufacturing system stakeholders.

Simple systems undergoing transition will be modeled and their behavior under various transition policies will be measured. Using the new, transient domain simulation modeling techniques, and with the aid of a meaningful objective function, the effectiveness of different transition strategies and mitigation techniques will be evaluated.

New simulation model objects will be developed to model elements of a manufacturing system with production control as a parametric feature. To do so, a new modeling paradigm will be introduced that considers the flow of information and demand as well as material through a manufacturing system. Simulation software will be developed to implement these new objects.

It will be demonstrated that simulation-based optimization, combined with new simulation model objects above, make it possible to find the optimal production control configuration for a general system.

The remainder of this dissertation is as follows: Chapter 2 reviews relevant literature. Chapter 3 explores the transition of a single stage system from push to pull. Chapter 4 extends the single stage lessons to a multi-stage system. Chapter 5 begins the discussion of a simulation modeling framework and introduces the Production Control Framework (PCF). Chapter 6 describes how the PCF can be used to optimize production control of a simple system. Chapter 7 summarizes my findings and addresses extensions of this research.

Chapter 2 Literature Review

Although Lean Manufacturing is a relatively new discipline in the West, it is based on well-understood operations management principles. As a result, one can look to the literature for a broad foundation upon which to base this discussion.

2.1 Production Control

Production control policies have an important impact on manufacturing system performance and production control is the mechanism through which this work brings about Lean transformation. It is an area of study well-addressed in the literature. This section briefly reviews literature on different types of production policies and dynamic scheduling techniques. (Methods that create and update production schedules are beyond the scope of this research.)

Dynamic scheduling does not create production schedules. Instead, decentralized production control methods dispatch jobs when necessary and use information available at the moment of dispatching. Such schemes use dispatching rules or other heuristics to prioritize jobs waiting for processing at a resource (Church and Uzsoy, 1992; Fang and Xi, 1997; Perkins and Kumar, 1989). Some authors refer to dynamic scheduling schemes as on-line scheduling or reactive scheduling (Sabuncuoglu and Karabuk, 1999; Li, Shyu and Adiga, 1993; Olumolade and Norrie, 1996).

Dispatching rules and pull mechanisms are used to control production without a production schedule. When a machine becomes available it chooses from among the jobs in its queue by using a dispatching rule that sorts the jobs by some criteria. Common dispatching rules employ processing times and due dates in simple rules and complex combinations. Some dispatching rules are extensions of policies that work well on simple machine scheduling problems (e.g. Shortest Processing Time and Earliest Due Date). The computational effort of dispatching rules is low when simple rules (like SPT or EDD) are used. However, some dispatching rules require a large amount of information, and the job priorities must be recalculated at every dispatching decision. Panwalkar and Iskander (1977) provide an extensive list of dispatching rules. They categorize these rules into five classes: simple dispatching rules, combination of simple rules, weighted priority indexes, heuristic scheduling rules, and other rules.

Green and Appel (1981) examine the problem of job shop scheduling by asking the following questions: What traditional dispatching rules do experienced schedulers select? Would dispatch rule selection be influenced by urgency? Would schedulers select a dispatch order based on organizational influence or peer pressure? The authors asked schedulers in a number of plants to denote which of the following rules they used: due date, slack, operation due date, slack per operation, shortest processing time, first come first served, program in greatest trouble, or friend needs a favor. The authors report that influence systems affect scheduling. The program in greatest trouble rule (a coalition rule) was highly valued, but friend needs a favor (an individual rule) was rejected. Traditional and theoretical rules were not highly valued.

Pull mechanisms such as kanban cards and constant inventory (CONWIP) order release policies add production authorization cards to the system so that a resource can work only when both material and cards are available. Hopp and Spearman (2000) provide a good introduction to these topics. Buzacott and Shanthikumar (1993) analyze a generalized production authorization system that includes a variety of traditional schemes as special cases.

Dynamic scheduling is closely related to real-time control, since decisions are made based on the current state of the manufacturing system. Controlling a manufacturing system so that it maintains a desired inventory position (in work-in-process or finished goods) is a common strategy when there is steady demand for each product. There may be multiple process flows (routes), but they are known, and each one has a steady throughput of jobs following that flow. This consistency makes kanban, other pull-based mechanisms, hedging points, and base stock policies feasible. The system works to maintain a low level of work-in-process, but the consistent demand insures that this inventory turns over regularly. See, for example, Hopp and Spearman (2000), Gershwin (1994), and Bispo and Tayur (2001).

Gershwin (1994) reviews literature of control theoretic models of manufacturing systems. The models are used to develop rules for deciding which action to take and when to take it in response to random disruptions. For instance, these control policies can be implemented as dispatching rules or hedging-point policies.

A number of papers (e.g., Perkins and Kumar, 1989; Kumar, 1994; Chase and Ramadge, 1992) have studied the control of dynamic manufacturing systems. Specifically, they have described classes of dispatching rules that identify which waiting job a resource should process next. For machines without setup times, the proposed dispatching rules are a class of least slack policies that prioritize each job by the difference between its due date (or some surrogate) and the expected amount of time until the job is completed. For resources with setup times, the proposed dispatching rules focus on completing all waiting jobs of one type before performing a setup and processing jobs of another type. All of the rules studied keep a machine working if there are any jobs waiting for processing. (That is, the machine cannot ignore waiting jobs.) Kumar (1994) summarizes the results of work on the stability and performance of these policies. This important work demonstrates why certain classes of dispatching rules work well and provides guidance when selecting dispatching rules. However, there exist dynamic manufacturing systems for which these types of dispatching rules are inappropriate or suboptimal. For example, Chase and Ramadge (1992) demonstrated that there exist idling policies that have superior performance. For a single machine operating in a dynamic, stochastic environment, Markowitz and Wein (2001) present dynamic cyclic policies that minimize the long-run expected average costs of earliness, tardiness, holding, and setups.

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 (2000) 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 hybrid production control schemes do not exist, to my knowledge.

All of these references describe different methods for production control of a manufacturing system. However, none of them describe a general framework in which these production control techniques can be related and described with respect to each other. To find the optimal production control policy for a general system, a framework must be created that allows all of these techniques to be described, preferably in a numerical format that can be easily manipulated by computerized searches of the production control domain. This dissertation will propose such a framework.

2.2 Simulation Modeling

Simulation modeling is the principal means of exploring production control used in this research. Numerous sources describe the use of simulation for predicting performance, comparing alternatives, and optimizing system designs. Law and Kelton (1991), a well-known text on discrete-event simulation, discusses the simulation of manufacturing systems. In many cases, simulation studies have been used to gain insight into the behavior of manufacturing systems under different types of control policies (e.g., different dispatching rules) or to determine the accuracy of analytical models. Vollmann, Berry, and Whybark (1997) review a number of results, for instance.

These references demonstrate the utility of simulation to analyze the steady-state performance of production control policies. Simulation in general is focused on steady state performance of models. Modern simulation methodologies and software tools are specifically designed to limit transient effects on measurements. This work is concerned exclusively with transient behavior of systems undergoing a change from one production control policy to another. To study such systems using simulation models, this dissertation introduces new techniques to set up and conduct experiments and to collect performance data during transient behavior.

2.3 Lean Manufacturing and the Toyota Production System

Lean Manufacturing, also known as just-in-time manufacturing, is a Western adaptation of the Toyota Production System, a business philosophy developed by that Japanese carmaker in the 1950's. Originally conceived as a way to maximize the use of the company's limited post-war manufacturing resources, it spread beyond just manufacturing and became a central tenet of their corporate culture to eliminate waste in all processes. Numerous books and articles have appeared to discuss the topic and its roots in the Japanese automotive industry (e.g., Schonberger, 1982; Womack and Jones, 1996; Hopp and Spearman, 2000; Askin and Goldberg, 2002). For controlling manufacturing operations, this philosophy advocates pull production control policies such as kanban. However, Lean manufacturing advocates other techniques such as employee cross-training, job rotation, continuous improvement, just-in-time purchasing, setup and other variability reduction, production smoothing, cellular layouts, and total quality management.

Liker (1997) describes a sequence of phases that a manufacturing facility must visit to become Lean: process stabilization, continuous flow, synchronous production, pull authorization, and level production. Such anecdotes are useful advice for managers and provide a general framework for becoming Lean, although they do not provide specific strategies for changing production control schemes.

The Lean literature, though large and ever-growing, is incomplete due to its focus on the description of Lean systems in steady state operation, ignoring the challenge and costs of Lean implementation in non-Lean systems. This research focuses exclusively on this neglected, but critically important facet of Lean practice. It proposes methods to find both the optimal steady state Lean production control configuration, but also the optimal implementation strategy. Objectives for these optima are stated in monetary terms, rather than in broad generalities.

2.4 Lean Transition

Although much has been written about how Lean Manufacturing systems should work, very little is said about how to change an ordinary system to make it Lean. The literature is nearly silent on the subject. Most Lean books are anecdotal at best and deal principally with cultural change management. There has been little scientific study applied to the mechanisms and effects of Lean transition on existing manufacturing systems.

The transient behavior of manufacturing systems is rarely studied. Most researchers focus on the steady-state behavior. Analyzing the transient behavior of a queueing system (for instance) is sometimes feasible but remains complicated and too specialized. Simulation studies often deliberately ignore the transient behavior by letting simulation runs finish a warm-up period before collecting statistics.

The study of discrete event systems has yielded techniques like Markov chains for determining, given an initial state, the probability distribution of the system state as the system changes over time. See Cassandras and Lafortune (1999) for an introduction to these topics.

Hopp and Spearman (2000) discuss the mechanics of push and pull production control, their role in Lean transition and even sketch out a Lean transition scheme. However, they limit their analysis to the "before" and "after" steady state conditions. In fact, to my knowledge, there is no discussion of the transient effects of Lean production control transition in the literature.

Queueing literature though does discuss the effects of non-stationary arrival rates on system performance. Hall (1991) discusses ways to model systems with non-stationary arrival rates. He explains that the size of changes in arrival rate relative to capacity dictate which modeling technique to use. For systems in which the arrival rate is always much lower than capacity, steady state approximations can be used. In systems where the arrival rate is much larger than capacity, a fluid flow approximation is more appropriate. For systems where the arrival rate is close to capacity, he suggests that only simulation can accurately model system performance.

Meerkov and Zhang (2008) recently presented the first meaningful work in the area of transient behavior of manufacturing systems. They studied the effect of different levels of reliability on the time a system takes to reach steady state from a standing start with varying levels of initial buffer occupancy. Their work is a compelling start, but they did not address transient effects due to changes in the manufacturing system itself, nor did their work attempt to address the cost of transient effects. While they provide good advice for how to manage a static system subject to periodic interruptions (shift changes), this work is focused on how to manage dramatic one-time changes to the manufacturing system.

2.5 Optimizing Production Control Transition Strategies

This research studies the effects of changing the production control policy of a manufacturing system. These types of changes are important to manufacturers implementing Lean manufacturing initiatives. It proposes doing so by using simulation based optimization to find not only the optimal end condition production control policy, but also the series of interim production control policies, each optimized for lowest cost.

Simulation models provide only approximate measures of manufacturing system performance. As a result, automated optimization algorithms for use with simulation models must be carefully designed in order to provide credible results. Simulation optimization refers to techniques that use simulation to solve stochastic optimization problems. This may be done because it is impossible (or difficult) to evaluate the objective function explicitly. For reviews of simulation optimization techniques, see Banks (1998), Fu (1994), and Pflug (1996). For instance, Pflug identifies two classes of methods: black box methods and white box methods. Black box methods use simulation to estimate the objective function and an optimization algorithm to search for the best solution. White box methods use a more sophisticated simulation program that can estimate gradients. Consequently, the optimization algorithm is a gradient-based technique.

Most of the techniques presented consider solutions with continuous variables. For instance, finite difference stochastic approximation (FDSA), introduced by Kiefer and

Wolfowitz (1952), has been applied extensively for continuous optimization. Spall (1998) describes the implementation of simultaneous perturbation stochastic approximation (SPSA) for continuous optimization problems.

Local search techniques move from one feasible point to another in search of the optimal solution. These techniques vary in the choice of the neighborhood structure, the decision strategy when moving from the current alternative to the next alternative and the method for obtaining estimates of the optimal solution. See, for instance, Andradottir (1995, 1996), Alrefaei and Andradottir (1995, 1999), and Yan and Mukai (1992).

The gradient-based discrete optimization techniques obtain estimates of the gradient of the expected system performance, with respect to a discrete parameter. The common techniques for estimating the gradient include finite differences and simultaneous perturbation methods. Gerencser, Hill and Vago (1999) proposed a fixed gain version of SPSA and applied it to a class of discrete resource allocation problems formulated by Cassandras, Dai and Panayiotou (1998).

These references show that simulation based optimization is a powerful tool for finding the optimal design of manufacturing systems. However, this tool has never been applied to systems undergoing transient behavior. This dissertation will employ simulation based optimization to reduce the effect and/or cost of transient behavior in a manufacturing system.

2.6 Summary

This work is intended to provide practitioners with a better understanding of the Lean transition and unambiguous guidance and/or tools to minimize the cost of implementing Lean.

To do so, a clearer understanding of the cost of transition is necessary. The Lean literature rarely mentions cost during transition, and to the best of my knowledge, has never attempted to quantify it. An explicit definition of the cost(s) of transition is required so that it can be measured and ultimately controlled.

Simulation modeling provides a useful tool with which explore the costs of transition and test mechanisms with which to control it. However, simulation modeling is typically used to study systems in steady state. In fact, the simulation literature addresses transient behavior as a factor to be eliminated. By its nature, this work is interested in the transient behavior of systems. Thus, a new simulation technique is needed to study a system exhibiting exclusively transient behavior.

If it can be shown that simulation is a viable tool for studying Lean transition, new modeling techniques will be required to facilitate experimentation with different production control schemes. Push type production control is relatively easy to simulate, but pull is much more complex. Like its real-world counterpart, it requires much more interaction between its elements. New modeling objects are needed to ease the programming burden of modeling different production control schemes.

Simulation applications today almost all come equipped with powerful optimization engines that could be used to find the optimal production control scheme for a system. However, these engines typically can change only parametric features of a model. Thus, a new modeling technique that can express production control schemes parametrically is required in order to unlock the power of simulation-based optimization of production control.

If the existing literature can be extended in these ways, Lean practitioners will finally have the tools they need to make informed decisions about how best to manage the cost of changes to their systems.

Chapter 3 Production Control Transition, Single Stage

The atomic element of Lean transition is the conversion of a single stage manufacturing system from push production control to pull. This chapter examines that conversion – the events that characterize such a transformation, the costs incurred, techniques to mitigate those costs and the effect of business considerations on the optimal transition of a single stage.

3.1 Introduction

Firms that implement Lean principles commonly adopt pull production control techniques (especially kanban cards) to limit work-in-process inventory and coordinate production activities. At the same time, they are moving the interface for customer orders from the beginning of the production line towards the end of the production line. This interface is sometimes called the “push-pull interface” or “inventory/order interface” (Hopp and Spearman, 2000). This change reduces the average time that customers wait for fulfillment of their orders.

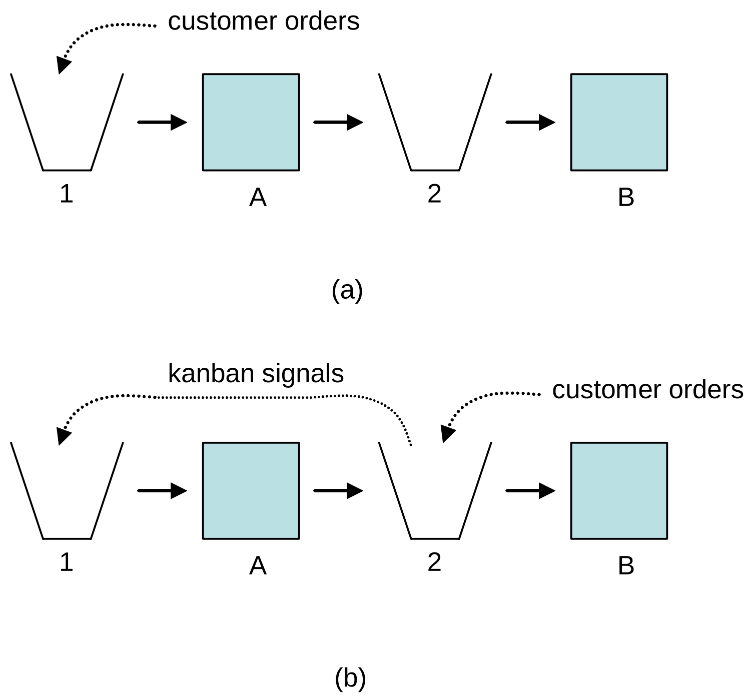


Figure 1. A two-stage manufacturing system. (a) The inventory/order interface is at the raw material buffer 1. Customer orders trigger work at workstation A using raw material. Items that workstation A completes then go to buffer 2, where workstation B processes them. (b) The inventory/order interface is at the work-in-process buffer 2. Customer orders trigger work at workstation B, which removes items from buffer 2, which sends production authorization signals to workstation A.

The fundamental element of this transition is the conversion of a single processing stage from push (production based on customer demand) to pull (production based on the status of the downstream operations, using kanban cards as a signaling mechanism). This requires that the stage produce items to which the kanban cards can be attached; however, no customers have requested the items. Therefore, during this transition, the stage experiences a surge in its workload as it attempts to build a kanban inventory while it is also processing regular customer orders. The surge may overwhelm the capacity of the station, resulting in a backlog that would adversely affect customer lead times. To prevent this, we propose two temporary mitigating techniques – adding more resources or deferring some of the customer orders. The purpose of this chapter is two-fold; to determine the best way to model this transition process and, using this model, to explore optimal mitigation policies to reduce the cost of Lean transition.

The Lean literature unanimously advocates the transition of push production control to pull where possible (Hopp and Spearman, 2000; Liker, 2004; Shingo, 1989; Slack, 1997; Womack, Jones and Roos, 1991), but very little is written about the mechanics of the transition process or the behavior of systems in Lean transition. Hopp and Spearman (2000) discuss the mechanics of push and pull production control, their role in Lean transition and even sketch out a Lean transition scheme. However, they limit their analysis to the "before" and "after" steady state conditions. In fact, to our knowledge, there is no discussion of the transient effects of Lean production control transition in the literature. Queueing literature though does discuss the effects of non-stationary arrival rates on system performance. Hall (1991) discusses ways to model systems with non-stationary arrival rates. He explains that the size of changes in arrival rate relative to capacity dictate which modeling technique to use. For systems in which the arrival rate is always much lower than capacity, steady state approximations can be used. In systems where the arrival rate is much larger than capacity, a fluid flow approximation is more appropriate. For systems where the arrival rate is close to capacity, he suggests that only simulation can accurately model system performance.

The remainder of this chapter is organized as follows. Section 3.2 describes the problem and gives an example. Section 3.3 describes the modeling approach for different cases and compares the performance of the models. Section 3.4 presents the results of an optimization study. Section 3.5 concludes the chapter.

3.2 Problem Setup

In this work we study a single stage of a production system as it transitions from push production control to pull and moves the customer order interface from immediately before this stage to immediately after it. The stage currently receives customer orders (one item per order) at a rate of λ_a orders per unit time, but orders could be deferred. The order deferral rate, a decision variable, is λ_d orders per unit time. There is a never-ending supply of raw material. The stage has r resources to manufacture items, but more resources could be added. The number of resources added, a second decision variable, is denoted by r_+. Each resource processes items at a rate of μ_r items per unit time. The resource will process an item only if there are orders or kanban cards that have arrived but have not been completed. The total processing rate of the stage is $r\mu_r$ items per unit time. Thus, the stage can be modeled as a *G/G/m* queueing system (Hopp and Spearman, 2000). This notation says that the distribution of the interarrival and service times are general, hence *G/G*, and that the system has *m* servers. Hopp and Spearman (2000) provide approximations to describe the behavior of such a system.

We characterize the transition from push to pull in terms of three events. The first event is the arrival of the first kanban card, which occurs at time $t = t_0$. We assume that the number of kanban cards, n_k, is predetermined. The kanban arrival rate, the third decision variable, is constant so that one card arrives every $1/\lambda_k$ time units. The last kanban card arrives at $t_1 = t_0 + (n_k - 1) / \lambda_k$. Clearly t_1 is greater than t_0. The total arrival rate for $t_0 < t < t_1$ is $\lambda_k + \lambda_a - \lambda_d$. If $\lambda_k + \lambda_a - \lambda_d > (r + r_+)\mu_r$ then a backlog of customer orders and kanban cards accumulates during this time ($t_0 < t < t_1$). The final transition event occurs at t_2 when the item for the last kanban card is completed. At this point, the stage can be converted to pull production control because customer orders can now be satisfied from the downstream buffer that is now full of items (with kanban cards).

If $\lambda_k + \lambda_a - \lambda_d << (r + r_+)\,\mu_r$, then t_2 could be as little as $t_1 + 1/\mu_r$. However, if $\lambda_k + \lambda_a - \lambda_d >> (r + r_+)\,\mu_r$, and a backlog of size n_b orders forms, then t_2 could be as large as $n_b / ((r + r_+)\,\mu_r - \lambda_a + \lambda_d)$.

We are interested in how the system behaves during this transient phase and how the three decision variables affect the performance of the system.

3.2.1 Decision Variables

There are three decision variables for managing the transition process. Since the number of kanban cards is assumed to be fixed, the first decision variable is the rate of their introduction λ_k, where $0 < \lambda_k < \infty$. (As an alternative, one could specify the length of the transition, $t_1 - t_0$, where $\lambda_k = (n_k - 1)/ (t_1 - t_0)$.) The second decision variable is the number of additional resources r_+, where r_+ is an integer on $0 \le r_+ < \infty$. The third decision variable is the deferral rate of customer orders λ_d, where $0 \le \lambda_d \le \lambda_a$.

3.2.2 Transition Cases

From this, we can identify three distinct conditions under which transition takes place based on the ratio of arrival rate and processing rate. This ratio, also called the traffic intensity, is central to our discussion, so we define the transition traffic intensity as ρ, where

$$\rho = \frac{\lambda_k + \lambda_a - \lambda_d}{(r + r_+)\mu_r} \tag{1.}$$

We may then define our transition cases with respect to traffic intensity, since it directly affects how much of the transition time occurs before and after t_1.

- **CASE 1: Arrival Rate is Lower than Processing Rate: $0 < \rho << 1$**

  In this case, the increased arrival rate does not completely consume the available capacity, and no backlog is created during transition. The kanban cards are processed as they arrive and the transition is nearly complete at t_1, minimizing t_2.

- **CASE 2: Arrival Rate is Higher than Processing Rate: $\rho >> 1$**

  In this case, the increased arrival rate completely consumes the available capacity, and a backlog of kanban cards and customer orders is created during transition. Here t_1 may be minimized as more, possibly all, of the kanban cards are processed after the final arrival.

- **CASE 3: Arrival Rate is Equal to Processing Rate: $\rho \approx 1$**

  In this case, the increased arrival rate nearly equals the available capacity. A backlog of customer orders and kanban cards may be created. When arrivals and capacity are nearly in balance, the formation of a backlog becomes more dependent on variation in processing times. For this condition, t_1 may be at any point between t_0 and t_2.

An interesting feature of this problem is the fact that there are decision variables within this identification scheme, meaning that the nature of the problem itself is a function of the inputs.

3.2.3 Transition Objective

Holding customer orders and kanban cards in a backlog, holding completed items (with kanban cards) in inventory, adding resources, and deferring orders all incur costs. Our goal is to find the values for the decision variables that minimize the total cost, which we denote as C_{tot}:

$$C_{tot} = C_d + C_r + C_i + C_b \qquad (2.)$$

where C_d is the cost of deferring orders, C_r is the cost of adding resources, C_i is the cost of holding items with kanban cards in the downstream queue, and C_b is the cost of holding customer orders and kanban cards waiting for processing in the backlog. We can further define these cost components in terms of the system variables we have already defined:

$$C_d = \lambda_d \, (t_2 - t_0) \, c_d \tag{3.}$$

$$C_r = r_+ \, (t_2 - t_0) \, c_r \tag{4.}$$

$$C_i = (n_k/2) \, (t_2 - t_0) \, c_i \tag{5.}$$

$$C_b = c_b \int_{t_0}^{t_2} \mathbf{Q}(t) dt \tag{6.}$$

In the equations above, c_d is the cost of a deferred order, c_r is the cost per unit time of an additional resource, c_i is the cost per item per unit time of holding a processed item, and c_b is the cost per order (or card) per unit time of holding customer orders and kanban cards waiting for processing. Q(t) is the backlog, the number of customer orders and kanban cards waiting for processing in the upstream queue, at time t during the transition period. We note that, as the deferral rate λ_d increases, the deferral cost increases, but the other costs decrease due to the smaller backlog and shorter transition time. Similarly, as r_+ increases, the resource cost increases, but the other costs decrease due to the smaller backlog and shorter transition time. Increasing the kanban introduction rate λ_k should reduce the transition time unless it is too large, in which case excessive demand increases the backlog.

3.2.4 Example

To illustrate these concepts, consider the following example. Table 1 lists the key parameters of a manufacturing system.

Table 1. Example for Comparison

Parameter	Value
Order arrival rate λ_a	75 items per unit time
Service rate μ_r	10 items per unit time
Number of resources r	10
Kanban cards to be introduced n_k	240
Decision Variable	Value
Number of resources added r_+	10
Order deferral rate λ_d	0

Before it transitions from push to pull, traffic intensity is 0.75. In this case, a downstream kanban buffer of 240 units is desired, which equates to 2.4 time units worth of buffer between this system and a downstream process. The nature of the transition is dictated by the rate at which the kanban cards are introduced since the system must process the cards at the same time it addresses new customer orders. To observe these effects and gain more insight into the transient behavior of a system in transition, we built a simulation model of this manufacturing system that provides time plots of queue lengths and resource availability. We consider three different transition scenarios.

If we introduce the cards very slowly, the impact on the system performance might be negligible, but time required to complete the transition stretches out, which bears its own cost. If the cards are introduced at a rate of 5 per unit time, the combined order and card arrival rate equals 80 items per unit time. Thus, traffic intensity during kanban introduction equals 0.8. Even without adding extra resources during the transition, this increase in traffic intensity will have only a modest impact on system performance. However, the transition will take 48 time units. During that time, the inventory waiting with kanban cards after being processed will incur a holding cost of $C_i = 5760\ c_i$. Figure 2 shows the output plot from the model showing the number of orders in backlog as the inventory buffer fills. In the plot, the pink line shows the number of orders in backlog, the blue line indicates the number of processing resources in use at a given time and the yellow line shows the number of parts in the kanban inventory. One can see that there is little visible difference in backlog or resource utilization before and after the start of kanban card introduction begins at $t = 2$. As predicted, the kanban inventory fills at about $t = 50$, 48 time units after the start of transition.

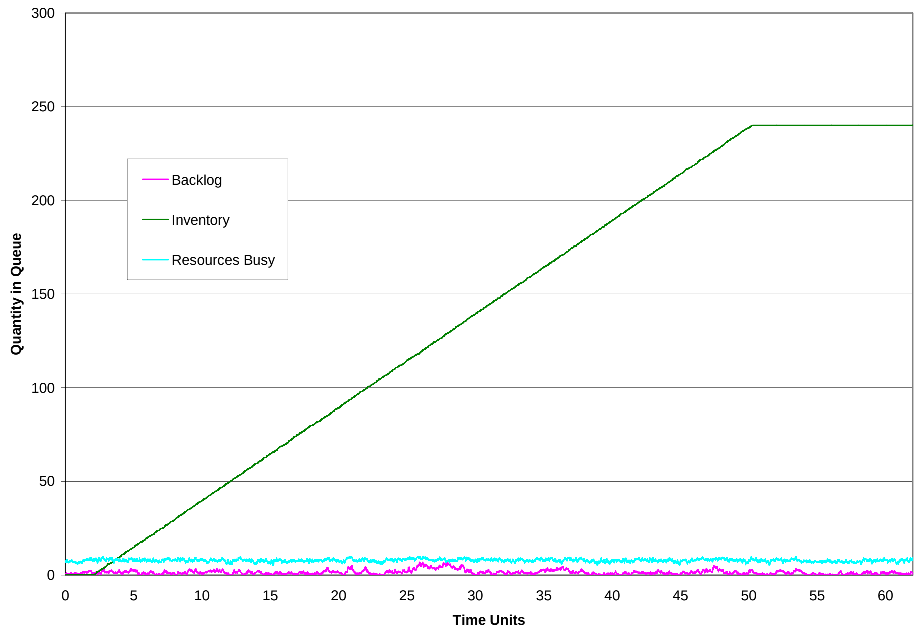


Figure 2. A plot of quantity in queue versus time for a single-stage transition, $\lambda_k = 5$

If the kanban cards are introduced at a higher rate, we observe a situation where the system may be overloaded if not for the addition of extra resources. By introducing the cards at a rate of 60 per unit time, the traffic intensity climbs to 1.35. Without additional resources, the system will see a quickly increasing backlog throughout the 4 time unit card introduction phase, followed by a slow reduction in the backlog over a period of 5 to 6 time units. System performance will be severely degraded during the entire period. However, if $r_+ = 4$ additional resources are employed, the traffic intensity will decrease to 0.96. System performance will still suffer, but the effects will be substantially mitigated, and the transition duration will be shortened. Figure 3 shows the plot for this scenario, without resource mitigation. One can see that the backlog increases steadily after the kanban cards are introduced until the inventory is filled. Once the inventory is filled, the backlog disappears at roughly the same rate at which it first appeared. During the entire overload period, all ten of the resources remain busy. At about $t = 16$ the backlog and resource utilization return to pre-transition levels.

This effect is even more pronounced if the cards are introduced all at once or at an extremely high rate, which has the same effect. If the cards are introduced all at once, there is an immediate spike of backlog that gradually tapers back to the initial steady state. The addition of extra resources would change the result by increasing the rate at which the backlog is served, but would not change the fact that new customer orders that arrive during the transition would wait in line behind all of the kanban cards. In Figure 4, which illustrates this scenario, the introduction of the kanban cards causes an immediate jump in the backlog, followed by a steady decrease. As in the previous example, the transition ends at about $t = 16$, but with a very different backlog profile, which could result in radically different transition costs.

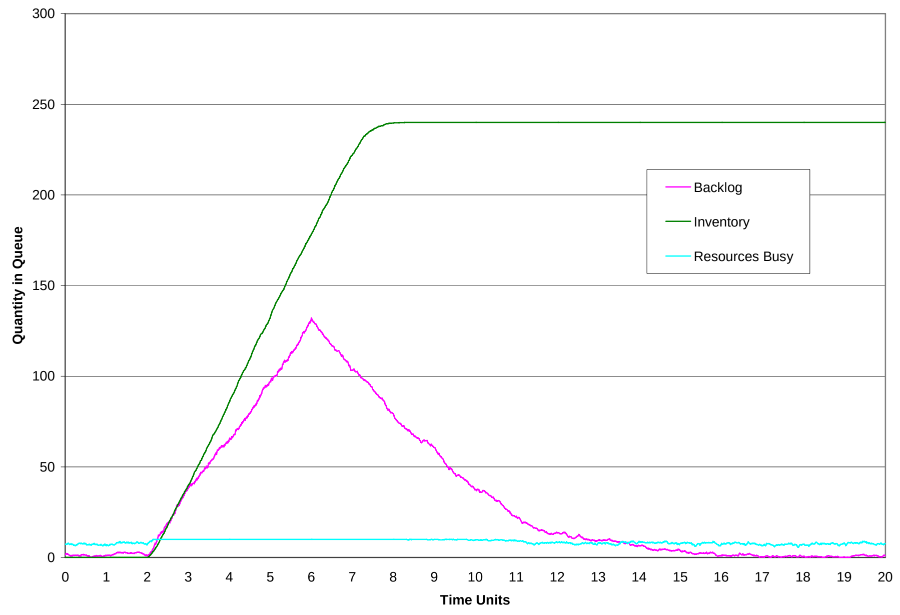


Figure 3. A plot of quantity in queue versus time for a single-stage transition, $\lambda_k = 60$

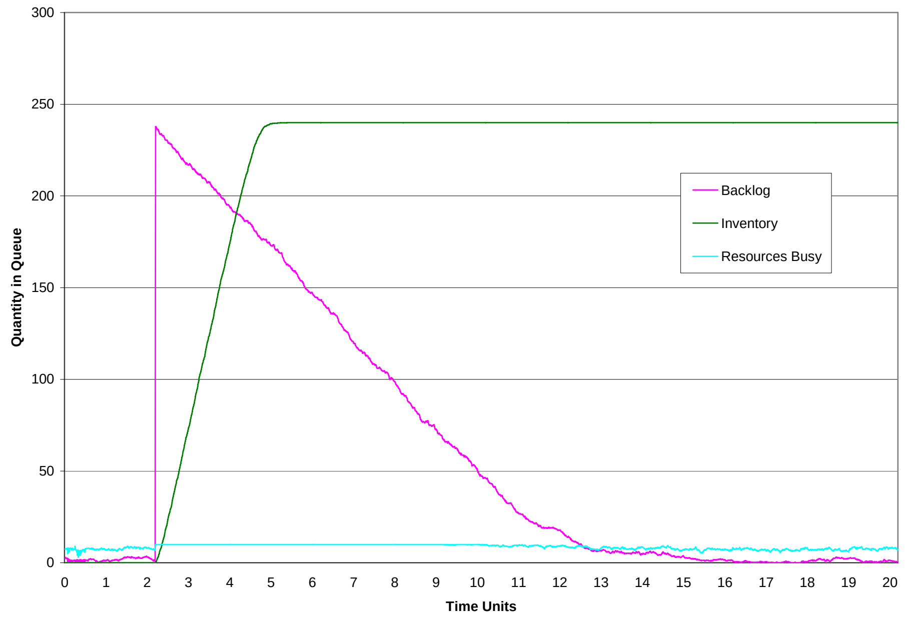


Figure 4. A plot of quantity in queue versus time for a single-stage transition, $\lambda_k = \infty$

By varying the kanban card arrival rate, we can observe a range of transition traffic intensities and measure the transition costs. Figure 5 illustrates the results of this exercise. It shows the change of each cost component and the transition duration as the traffic intensity during the kanban introduction period changes. The backlog was measured as the average number of orders in queue to be processed. The inventory was measured as the average number of items with kanban cards in the downstream kanban queue. For a relative comparison, the cost rates are all set to 100.

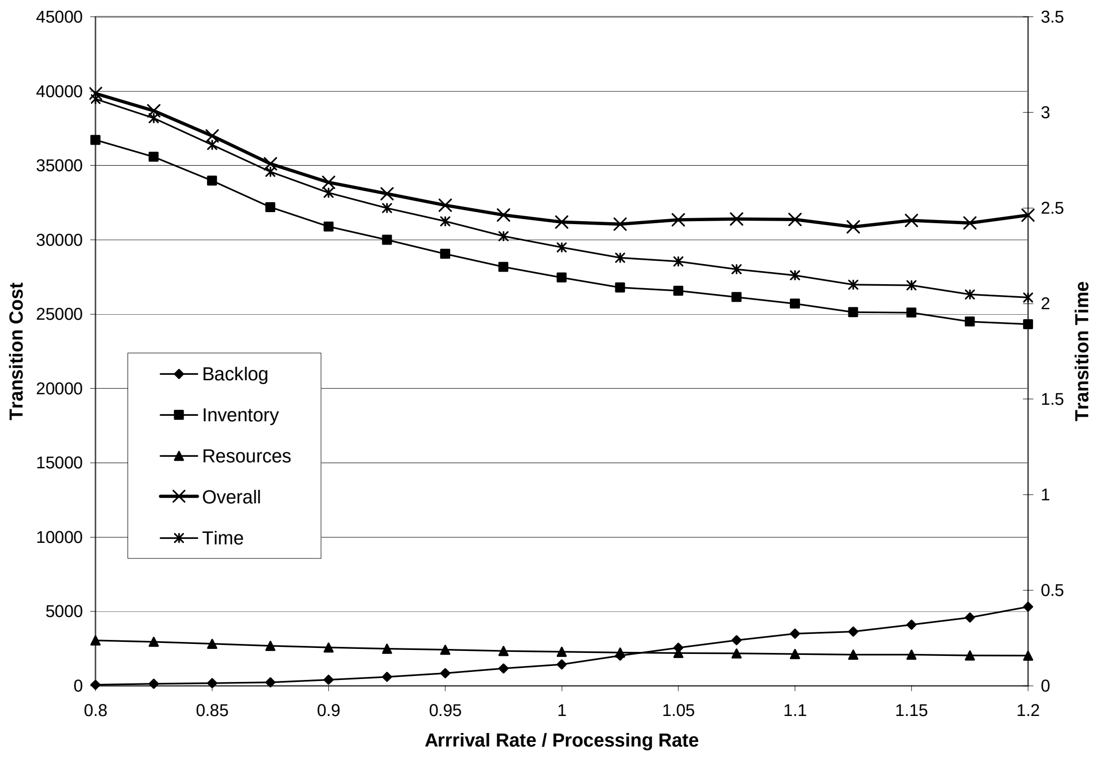


Figure 5. Transition cost and time versus traffic intensity

When the arrival rate is exactly balanced with the processing rate of the system (i.e., ρ is close to 1), the transition cost curve flattens and may even be at a minimum. When the kanban arrival rate is too low, the processing resources are underutilized and transition time is needlessly lengthened. When the kanban arrival rate is too high, the backlog and inventory costs increase the overall cost. We see here that the effect of backlog and resource costs are small compared to the contribution of inventory, but a real-world system with real cost rates could be quite different and drastically change these relationships.

We wish to develop a tool to find the optimal transition policy for any system undergoing push to pull transition. Since we wish to employ simulation based optimization, which will run a model at many different values of the decision variables, we need models that don’t require a long run time.

3.3 Modeling Approach

We consider three different techniques to model this process: steady state approximation, deterministic fluid flow approximation and discrete event simulation. Hall (1991) proposed that these three modeling techniques are the best candidates for analyzing systems with non-stationary arrival rates. Hall’s categorization of these systems corresponds with our case definitions discussed above. The following sections describe the models.

3.3.1 Case 1: Steady State Model, Arrivals << Capacity

First, we consider the case where the increased arrival rate is much smaller than the capacity of the resource. That is, where ρ is greater than zero, but less than one. In this case we use a stochastic steady state (SSS) approximation. For $t < t_0$, we assume the system is in a steady state. For $t_0 < t < t_2$ we assume that the system switches to a second steady state. For $t > t_2$, the system reverts to a third steady state similar to the first. Since the surge never exceeds capacity, there is no significant backlog to deal with at the end of the transition and $t_2 = t_1 + CT_q + 1/\mu_r$, where CT_q is the average time an order spends in the queue before being served. To estimate CT_q, we assume that the system has had time to reach steady state at $t = t_1$ and choose to use the approximation given by Hopp and Spearman (2000) for a *G/G/m* server. A *G/G/m* server is one where the variability of both the interarrival and processing times can be described generally (hence the G/G for General/General) using their respective coefficients of variation, c_a and c_e, respectively and where the number of servers is m. Using this approximation we can derive the following expression for t_2:

$$t_2 = t_1 + CT_q + \frac{1}{\mu_r} = t_0 + \left(\frac{n_k - 1}{\lambda_k}\right) + \left(\frac{c_a^2 + c_e^2}{2}\right)\left(\frac{\rho^{\sqrt{2(r+r_+ +1)}-1}}{1-\rho}\right)\frac{1}{(r+r_+)\mu_r} + \frac{1}{\mu_r} \tag{7.}$$

Using the same assumptions, we can approximate the average backlog before t_1 by substituting the approximation for CT_q into Little's Law:

$$Q(t_1) = CT_q(\lambda_k + \lambda_a - \lambda_d) = \left(\frac{c_a^2 + c_e^2}{2}\right)\left(\frac{\rho^{\sqrt{2(r+r_+ +1)}-1}}{1-\rho}\right)\frac{\lambda_k + \lambda_a - \lambda_d}{(r+r_+)\mu_r} \tag{8.}$$

For the period between t_1 and t_2, the backlog at t_1 disappears (by the definition of t_2), but more customer orders arrive. The expected number of new arrivals is $Q(t_2) = \lambda_a(t_2 - t_1)$. The average backlog between t_1 and t_2 is therefore the average of $Q(t_1)$ and $Q(t_2)$. We can then approximate the total backlog as follows:

$$\int_{t_0}^{t_2} Q(t)dt = Q(t_1)(t_1 - t_0) + \frac{(Q(t_1) + Q(t_2))}{2}(t_2 - t_1) \qquad (9.)$$

Thus we have a way to estimate the cost of Case 1 transitions.

3.3.2 Case 2: Fluid Flow Model, Arrivals >> Capacity

Next, we consider the case where the increased arrival rate is much greater than the processing capacity. That is, where ρ is greater than one. To model this transition, we use a deterministic model called a fluid flow approximation model. A deterministic fluid flow (DFF) approximation model is one in which the flow of arrivals and departures are modeled as continuous variables.

Figure 6 shows a fluid approximation model of our system in transition. In this figure, the blue line represents the cumulative number of customer order and kanban card arrivals during the transition. The red line shows the cumulative number of completed items. The slopes of these lines are equivalent to the arrival rate and processing rate. Evaluating this model is straightforward geometry. We address the transition in two phases; $t_0 < t \leq t_1$ and $t_1 < t \leq t_2$. Recall that $t_1 = t_0 + (n_k - 1) / \lambda_k$. In the first phase, the backlog is building. In the second phase it is being consumed. We first solve to find the backlog, $Q(t_1)$.

$$Q(t_1) = \left(\frac{n_k - 1}{\lambda_k}\right)\left(\lambda_a + \lambda_k - \lambda_d - (r + r_+)\mu_r\right) \qquad (10.)$$

We can then use this result to find t_2:

$$t_2 = t_0 + \frac{n_k - 1}{\lambda_k} + \frac{Q(t_1)}{-\lambda_a + \lambda_d + (r + r_+)\mu_r} \qquad (11.)$$

And substituting $Q(t_1)$ from above:

$$\int_{t_0}^{t_2} Q(t)dt = \frac{Q(t_1)}{2}(t_2 - t_0) \qquad (12.)$$

Because the accumulation and consumption of the backlog are linear in this approximation, the total backlog is simply half of the maximum backlog, which occurs at t_1, applied over the entire transition period.

$$\int_{t_0}^{t_2} Q(t)dt = \frac{Q(t_1)}{2}(t_2 - t_0) \qquad (13.)$$

These equations provide a straightforward way to calculate the cost of Case 2 transitions.

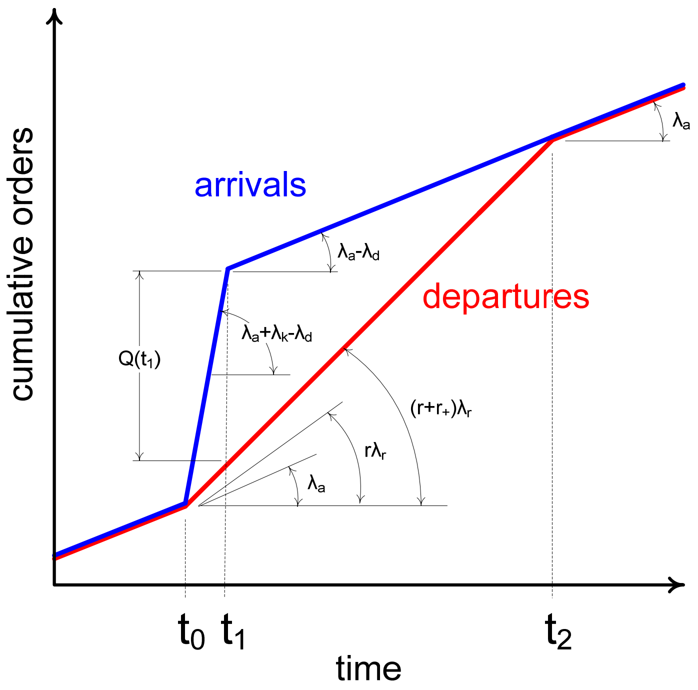


Figure 6. Deterministic fluid flow (DFF) approximation model

3.3.3 Case 3: Simulation Model, Arrivals ≈ Capacity

Finally, we address the case where the arrival rate is approximately equal to capacity. That is, where ρ is approximately one. In this condition, Hall (1991) recommends the use of simulation to model the system. Simulation is a very powerful, but computationally expensive, modeling technique. We did build a simulation model of our system using Arena (Kelton, Sadowski and Sturrock, 2004). Figure 7 shows the simple system as modeled.

The simulation model itself is fairly straightforward, but collecting good performance data from a potentially short, transient period requires careful setup of the replication parameters. My model uses a warm-up period of 10,000 times the order processing time, which it repeats for each replication. It maintains a count of how many kanban cards are in the system and it stops the replication when the last card exits. I use 100 replications. To evaluate performance we used the default reports which provide statistics on number of orders in backlog and replication length.

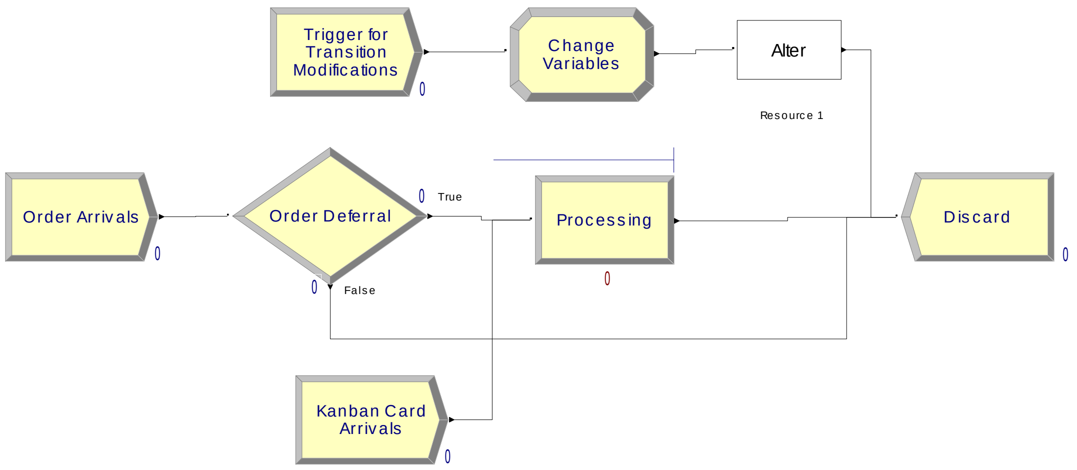


Figure 7. Simulation model of a single stage system, modeled using Arena

3.3.4 Comparison

In order to compare the models, we used them to estimate the performance of a system over a range of transition arrival rates centered about the capacity of the system. For this comparison, we consider again the example from Section 2.

We varied the kanban introduction rate, λ_k, from 85 to 165 items per time unit, which caused the traffic intensity to vary from 0.8 to 1.2, and used each model to predict the average number of orders in the backlog. Figure 8 shows a plot of each model's output.

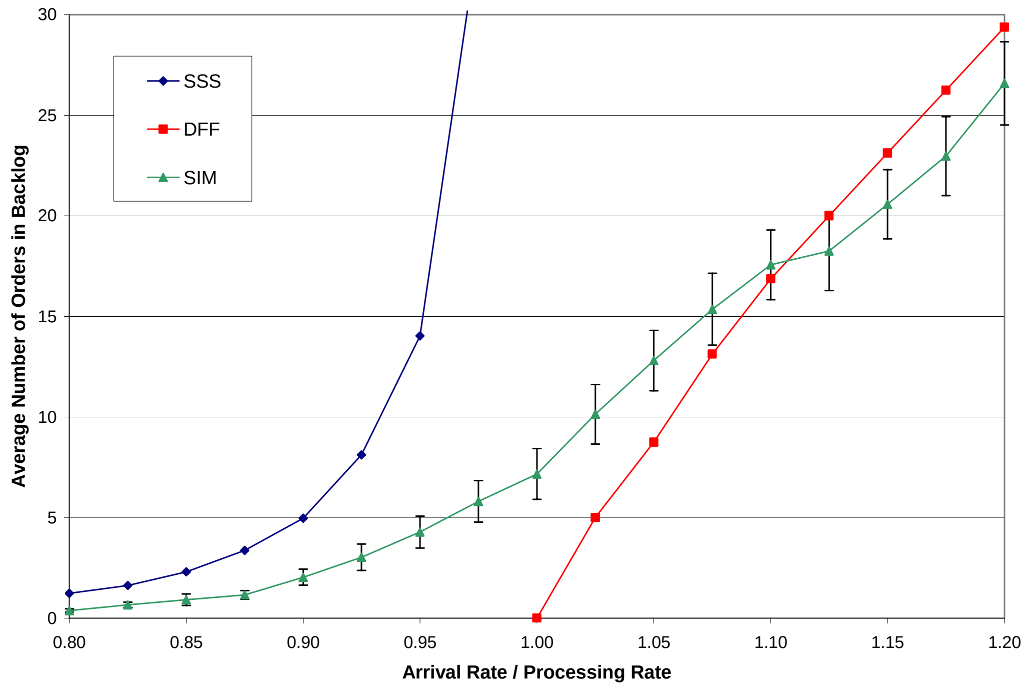


Figure 8. Average number of orders in backlog versus traffic intensity

We expected the simulation model to most closely approximate the behavior of the system, which we predicted would be a smooth, monotonically increasing curve as the arrival rate slowly overcame the processing capacity of the system. As expected, the SSS approximation followed the simulation results initially, but increased asymptotically to infinity as the arrivals approached capacity from the left. (The SSS approximation overestimates slightly because it assumes that the kanban card interarrival times are exponentially distributed, but they are actually constant.) The DFF reported backlogs well below the simulation model but caught up to and followed closely with it at higher arrival rates. If we assume that the simulation model is the closest approximation to the behavior of the system then the comparison is just as Hall predicted, with the SSS approximation most useful when the arrival rate is well below capacity (Case 1), the DFF approximation most useful at rates well above capacity (Case 2) and the simulation model best at rates near capacity (Case 3).

A key difference between the models is the computational effort that each one requires. The results of the SSS and DFF approximations can be obtained almost immediately from a spreadsheet model. The simulation model took nearly 2 minutes to process each data point. The flexibility of the simulation model has a high computational price that affects its usefulness for optimization of the transition.

3.4 Hybrid Model

The above results indicate that, for a system in which the decision variables are unconstrained, the optimal transition policy may be in the domain that we have identified as being best suited for simulation modeling. Thus, any automated optimization will likely spend a good deal of time searching the most computationally expensive region. In order to increase the speed of optimization, it would be useful to have an analytical solution for this region, even if it is merely an approximation. We derived linear approximations for the transition time and the total backlog for the case where the traffic intensity is between 0.9 and 1.5 (these values were selected after some experimentation to get the best fit). We can use these linear approximations to bridge the gap between the SSS and DFF model results and create a hybrid model.

Given a problem instance and values of the kanban arrival rate λ_k, the order deferral rate λ_d, and the number of additional resources r_+ such that $0.9 < \rho = (\lambda_k + \lambda_a - \lambda_d)/((r + r_+)\mu_r) < 1.5$, we will determine the transition time and average backlog for the two extreme values (by changing the traffic intensity ρ and determining the corresponding kanban arrival rate) and then interpolate to estimate the desired values.

$$\lambda_L = 0.9(r + r_+)\mu_r - \lambda_a + \lambda_d \tag{14.}$$

$$t_{1L} = t_0 + \frac{(n_k - 1)}{\lambda_L} \tag{15.}$$

For the moment, we assume that the order interarrival and service times are distributed exponentially, meaning $c_a = c_e = 1$.

$$CT_{qL} = \left(\frac{1^2 + 1^2}{2}\right)\left(\frac{0.9^{\sqrt{2(r+r_+ +1)}-1}}{(1-0.9)}\right)\frac{1}{(r + r_+)\mu_r} \tag{16.}$$

$$t_{2L} = t_{1L} + CT_{qL} + \frac{1}{\mu_r} \tag{17.}$$

$$Q(t_1)_L = CT_{qL}(\lambda_L + \lambda_a - \lambda_d) \tag{18.}$$

$$Q(t_2)_L = \left(\frac{\left(\frac{\lambda_a}{r\mu_r} \right)^{\sqrt{2(r+1)}-1}}{1-\left(\frac{\lambda_a}{r\mu_r} \right)} \right) \frac{\lambda_a}{r\mu_r} \tag{19.}$$

$$\int_{t_0}^{t_2} Q(t)dt_L = \frac{\left(Q(t_1)_L + Q(t_2)_L\right)}{2}\left(t_2 - t_0\right) \tag{20.}$$

$$\lambda_U = 1.5(r + r_+)\mu_r - \lambda_a + \lambda_d \tag{21.}$$

$$t_{2U} = \left(\frac{n_k - 1}{\lambda_U} \right)\left(1 + \frac{\lambda_U + \lambda_a - \lambda_d - (r + r_+)\mu_r}{-\lambda_a + \lambda_d + (r + r_+)\mu_r} \right) \tag{22.}$$

$$\int_{t_0}^{t_2} Q(t)dt_U = \left(\frac{(t_2 - t_1)}{2} \right)\left(\frac{n_k - 1}{\lambda_U} \right)\left(\lambda_U + \lambda_a - \lambda_d - (r + r_+)\mu_r\right) \tag{23.}$$

Given these reference points, we then interpolate as follows:

$$t_2 = t_{2L} + \frac{(\lambda_k - \lambda_L)}{(\lambda_U - \lambda_L)}\left(t_{2U} - t_{2L}\right) \tag{24.}$$

$$\int_{t_0}^{t_2} Q(t)dt = \int_{t_0}^{t_2} Q(t)dt_L + \frac{(\rho - 0.9)}{(1.5 - 0.9)}\left(\int_{t_0}^{t_2} Q(t)dt_U - \int_{t_0}^{t_2} Q(t)dt_L \right) \tag{25.}$$

We may now formulate a hybrid approximation for t_2 and the total backlog, using the SSS approximation at values of ρ below 0.9, the DFF approximation at values above 1.5, and the linear interpolation for values in between. To demonstrate, we developed a hybrid model of the example system introduced earlier and calculated values of t_2 and total backlog for a range of traffic intensities from 0.5 to 2.0. Figure 9 and Figure 10 illustrate how the values of t_2 and total backlog, respectively, vary as traffic intensity changes. The graphs include measurements taken from the simulation model along with the SSS, DFF and Hybrid approximations. The graphs also include a plot of error between the hybrid and simulation models.

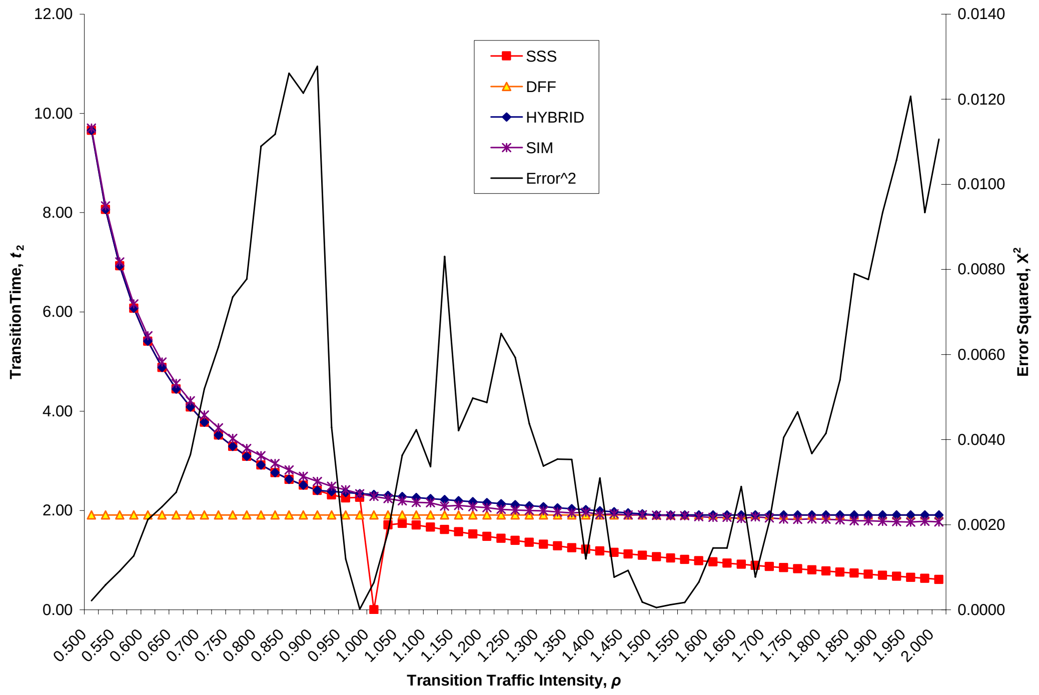


Figure 9. Transition time versus traffic intensity

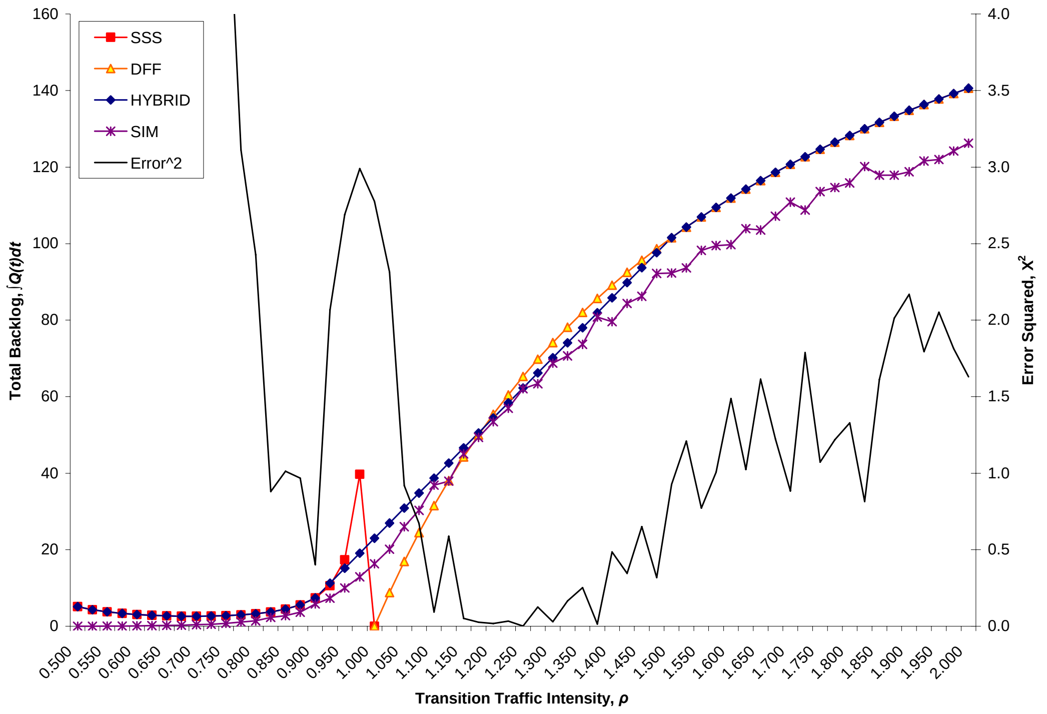


Figure 10. Total backlog versus traffic intensity

We see that the hybrid solution does indeed mimic the behavior of the simulation model. The error plots show that the hybrid model is not an exact match of the simulation, but it may still be useful for finding the optimum transition policy.

3.5 Sensitivity to Cost Rates

The cost rates affect the overall transition cost. To examine their effect, we substituted the hybrid models for t_2 and the total backlog into the objective function and measure the total cost of the example from Section 2.4 over the same range of traffic intensities as we used earlier. Figure 11 shows the how the overall transition cost changes as the transition traffic intensity varies. The graph shows the overall transition cost as well as the contributions of each component four component costs. (Note that the deferral cost in this example is zero.) For this illustration we set each of the four cost rates equal to one.

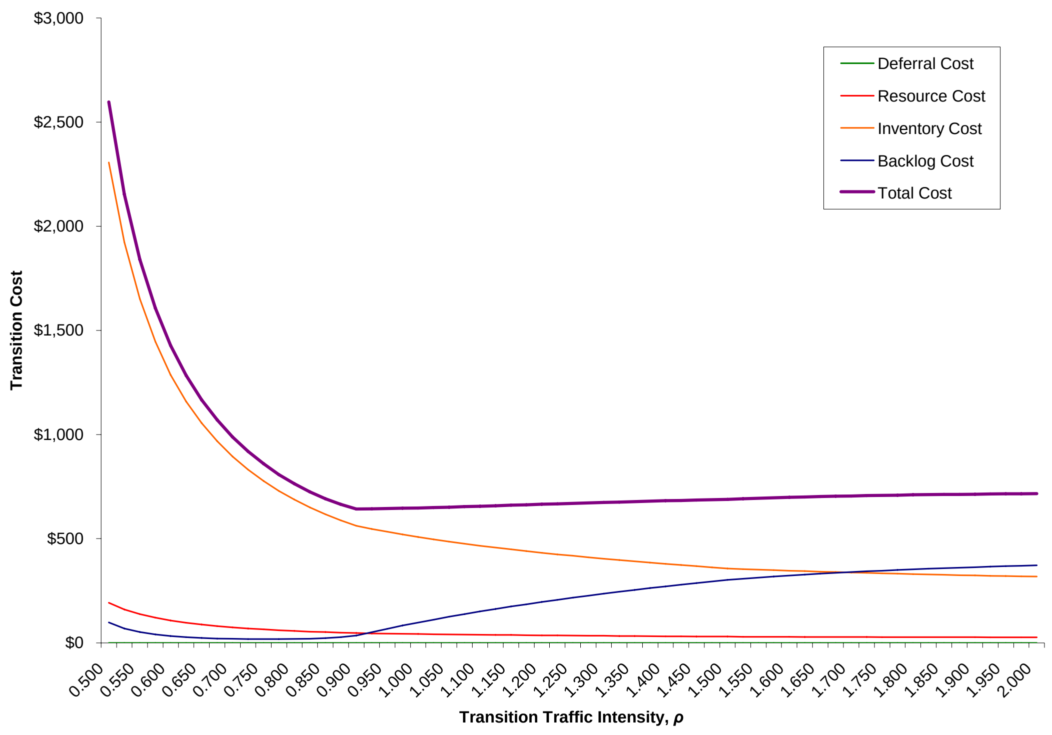


Figure 11. Transition cost versus traffic intensity

This graph closely mimics the results from the simulation model pictured in Figure 5. All but one of the cost components starts high at low traffic intensities and decreases monotonically as the intensity increases. The exception is the backlog cost, which decreases slightly to a minimum near where $\rho = 0.75$, then increases as the traffic intensity increases. The resulting overall cost function falls dramatically at first, then levels off and begins to climb slowly near where $\rho = 0.9$.

Since the total backlog is the only cost component that increases with traffic intensity, it follows that the magnitude of the backlog cost rate with respect to the other cost rates determines when, or if, the cost function reaches its minimum as the traffic intensity increases. Figure 12 shows only overall transition cost for different backlog cost rates. (Note that all of the other cost rates equal one in this example.)

From this we can derive an important rule of thumb for Lean transition. If one is not concerned with backlog cost ($c_r \approx 0$), it is best to introduce the kanban cards into the system all at once ($\lambda_d = \infty$). However, if backlog cost is a significant concern, more analysis is required. The hybrid approximation model makes this analysis straightforward.

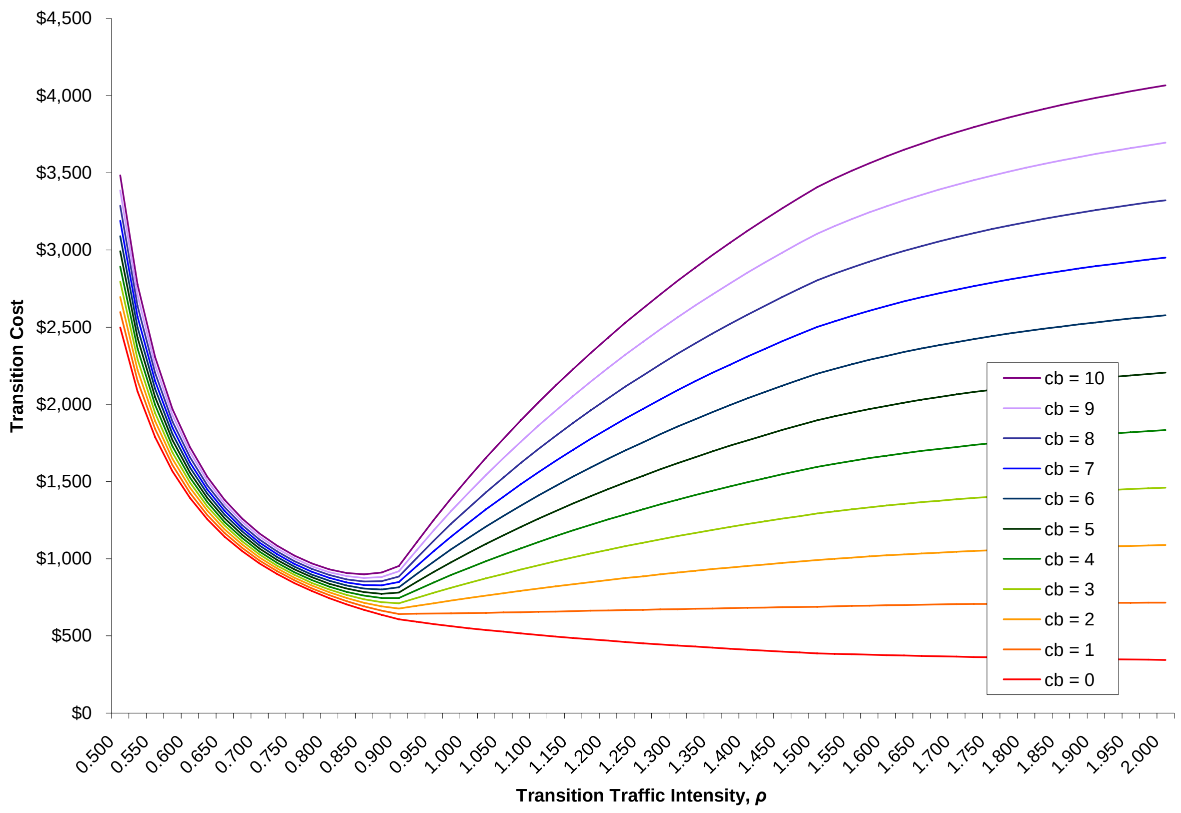


Figure 12. Transition cost versus traffic intensity at varying backlog cost rates

3.6 Simple Transition Policies

Using these models, it should be an easy thing to develop tools to find an optimal transition policy. However, I wish to determine if such high-order analysis is, in fact, necessary. Could a practitioner, using the limited advice from the literature, develop a simple transition policy that minimizes transition cost? In order to establish a baseline for optimization, I considered some simple policies inspired by the Lean references and some that we considered to be intuitive approaches. I considered the following policies (Table 2 shows the corresponding values of the decision variables for each policy).

- Case 1 – Instantaneous Kanban Card Introduction with no Mitigation
- Case 2 – Instantaneous Kanban Card Introduction with Complete Order Deferral
- Case 3 – Kanban Introduction Rate Calibrated to Make Traffic Intensity Equals 1
- Case 4 – Kanban Introduction Rate Matched by Deferral of Customer Orders to Maintain Traffic Intensity
    - Case 4A – 33% Deferral of Customer Orders
    - Case 4B – 66% Deferral of Customer Orders
    - Case 4C – 100% Deferral of Customer Orders
- Case 5 – Kanban Introduction Matched with Additional Resource to Maintain Traffic Intensity
    - Case 5A – 50% Additional Resources
    - Case 5B – 100% Additional Resources

Cases 1 and 2 are the simplest and most often cited in the literature. Case 3 proposes to fill any perceived excess capacity with kanban card deliveries. Case 4 considers operating the system in a pre-transition configuration, such that the system does not see any increase in order arrivals, but where some fraction of the previously customer generated load is replaced with kanban card arrivals. Case 5 addresses a system in which order deferral is not an option and the kanban card arrivals are balanced with additional resources such that the traffic intensity remains constant.

Using the simulation model developed in Section 2, I measured the cost of each of these policies. Because transition cost is sensitive to backlog cost, we measured the cost when the backlog cost rate, c_b, is \$1 as in the original experiments, and where it is \$2 as in the previous section. This is intended to demonstrate the potential impact of an incomplete understanding of transition cost.

Table 2. Transition costs for simple transition policies, measured using the hybrid approximation model

Example Case	Kanban Introduction Rate, λ_k	Additional Resources, r_+	Order Deferral Rate, λ_d	Transition Traffic Intensity, ρ	Transition Cost, C_{tot}	
					$c_b =$ \$1.00	$c_b =$ \$2.00
1	∞	0	0	∞	\$873.14	\$1405.98
2	∞	0	75	∞	\$1070.62	\$1600.78
3	25	0	0	1.0	\$1388.41	\$1579.58
4A	25	0	25	0.75	\$1627.46	\$1809.77
4B	50	0	50	0.75	\$1492.88	\$1923.23
4C	75	0	75	0.75	\$1461.30	\$1947.33
5A	37.5	5	0	0.75	\$829.57	\$832.34
5B	75	10	0	0.75	\$461.23	\$462.03

The most expensive policies were those that made use of order deferral to balance the traffic intensity. This is primarily due to the much longer transition times these policies incur. The most popular policies, Cases 1 and 2, fared slightly better, but demonstrated an acute sensitivity to backlog cost. In fact, although Case 1 had one of the lowest costs at c_b=\$1, its cost increased by 60% when c_b was changed to \$2. The lowest cost option, cases 5A and 5B proved to have the lowest overall cost and were also the least affected by changes in the backlog cost rate. However, the resource cost rate in this example is arguably the least realistic, meaning that the overall cost of these policies would likely be the most expensive in a real world application.

This exercise illustrates that there is a significant variation in the cost of even simple transition policies. Further, it shows that an incomplete understanding of transition costs, especially the effect of backlog cost, could lead to significantly higher overall transition cost. It is clear that there is a need for better decision support for transition policies. Using the model developed in the previous section and the on-board optimization engines in Excel, we can develop optimal transition policies.

3.7 Optimization

Using the hybrid approximation model from Section 4, we can employ powerful automated tools to find the optimal transition policy without expensive simulation runs. (For this problem, I used the Solver in Microsoft Excel.) We know that when the backlog cost rate is zero, the optimal transition traffic intensity is infinity. For other backlog cost rates, I found the optimal transition policy. Table 3 shows the results of this experiment.

Table 3. Optimal transition policies for varying backlog cost rates, obtained using the hybrid approximation model

Backlog Cost Rate, c_b	Optimal Transition Parameters				
	Kanban Introduction Rate, λ_k	Additional Resources, r_+	Order Deferral Rate, λ_d	Transition Traffic Intensity, ρ	Transition Cost, C_{tot}
\$0.00	9999	10	75	49.995	\$249.87
\$1.00	9999	10	75	49.995	\$420.46
\$1.50	9999	10	75	49.995	\$505.75
\$1.75	300.00	10	75	1.500	\$574.20
\$2.00	192.76	10	75	0.964	\$595.08
\$2.50	180.06	10	75	0.900	\$601.52

In this example, all three decision variables were allowed to change, subject to the following constraints:

$$0 < \lambda_k \leq 9999 \tag{26.}$$

$$0 < r_+ \leq r_0 \tag{27.}$$

$$0 < \lambda_d \leq \lambda_a \tag{28.}$$

The data validates the rule of thumb that when the backlog rate is very low, the optimal kanban introduction rate and transition traffic intensity tend to be as high as possible. As the backlog cost rate becomes more significant, the optimal kanban introduction rate decreases such that the traffic intensity decreases below 1.

The simulation model developed in Section 2 also features a built-in optimization engine.

I performed the same exercise using the simulation model. The results are shown in Table 4.

Table 4. Optimal transition policies for varying backlog cost rates, obtained using the simulation model

Backlog Cost Rate, c_b	**Optimal Transition Parameters**				
	Kanban Introduction Rate, λ_k	**Additional Resources, r_+**	**Order Deferral Rate, λ_d**	**Transition Traffic Intensity, ρ**	**Transition Cost, C_{tot}**
\$0.00	∞	10	75	∞	\$335.79
\$1.00	103	10	74.91	0.52	\$358.44
\$1.50	103	10	75	0.50	\$556.28
\$1.75	68	10	74.86	0.34	\$779.77
\$2.00	142	9	71	0.77	\$520.24
\$2.50	108	9	58	0.66	\$525.60

In this exercise, the optimization engine was allowed to search for 8 hours in order to limit the computational expense. However, this may have limited the effectiveness of the search, especially in the critical domain near where traffic intensity is 1. Despite the long search time, few of the analyses topped 100 trials. This may explain why the values neared, but did not settle on the parameter limits. Consider that the values in Table 3 took less than an hour to generate and it is clear why I seek an approximate analytical model.

In order to validate the hybrid model optimization, I used the values from the spreadsheet optimization in Table 3 and plugged them into the simulation model. The results of this substitution are shown in Table 5, below.

Table 5. Transition costs for hybrid approximation optimal policies, measured using the simulation model

Backlog Cost Rate, c_b	**Optimal Transition Parameters**				
	Kanban Introduction Rate, λ_k	**Additional Resources, r_+**	**Order Deferral Rate, λ_d**	**Transition Traffic Intensity, ρ**	**Transition Cost, C_{tot}**
\$0.00	9999	10	75	49.995	\$335.79
\$1.00	9999	10	75	49.995	\$533.88
\$1.50	9999	10	75	49.995	\$633.13
\$1.75	300.00	10	75	1.500	\$600.27
\$2.00	192.76	10	75	0.964	\$556.24
\$2.50	180.06	10	75	0.900	\$591.47

In general, the spreadsheet model costs are lower than those generated by the simulation model. We recognize that this is due to the error between the models, both in their estimates of total backlog and transition time, which tend to compound each others effect on total cost. From the data in Figure 9 and Figure 10, the region in which the error is smallest is where traffic intensity is between 1.2 and 1.5. We see that in the fourth trial, where the traffic intensity is 1.5, the difference in transition cost between the two models is minimized. The most notable result is that this error changes the location of the tipping point where the increasing backlog cost rate forces the optimal policy from instantaneous kanban card introduction to a more moderate rate. In the spreadsheet model, this point is between \$1.50 and \$1.75, whereas in the simulation model it is somewhere below \$1.50. We propose that despite the error in total cost, this error does not diminish the ability of the spreadsheet model to find a near-optimal transition policy very quickly.

3.8 Additional Examples

Both the example and the cost rates used in the previous sections were not very realistic. In order to better understand the significance of this technique and to learn more about the trade-offs of different transition policies, we can apply it to some additional examples.

First, consider a manual assembly cell undergoing Lean transition. The cell is manned by 4 workers. Each operator can process one order every 10 minutes. On average, an order for 1 item arrives every 3 minutes. Thus, the steady state parameters for this system are as shown in Table 6.

Table 6. Manual assembly cell example steady state parameters

Parameter	Value
Order arrival rate λ_a	20 orders per hour
Service rate μ_r	6 items per hour per worker
Number of resources r_0	4 workers
Traffic Intensity, ρ	0.83

from which we can calculate an initial traffic intensity of 0.83. The production planners wish to establish pull production control on this cell with kanban cards distributed twice per shift. Thus, they wish to build a kanban buffer equal to the 4-hour demand. That is, they wish to maintain an inventory of 80 items. They estimate their cost rates as

Table 7. Manual assembly cell example transition cost rates

Parameter	Value
Order deferral cost, c_d	\$1000 per order per hour
Resource cost, c_r	\$80 per worker per hour
Inventory cost, c_i	\$1.25 per order per hour
Backlog cost, c_b	\$1.00 per order per hour

Order deferral was an unattractive option for these planners, so they made the order deferral cost arbitrarily high. The resource cost was set to be the loaded cost of hiring temporary hourly workers. The inventory and backlog costs were determined based on holding cost rates and the earned value of the items.

Using these values, I configured the hybrid model and used the solver to determine the optimal transition policy and cost. Constraints on the optimization were kept the same as in Section 3.7. The model yielded the values in Table 8.

Table 8. Manual assembly cell example optimal transition policy, obtained using the hybrid approximation model

Parameter	Value
Order deferral rate, λ_d	0 orders per hour
Additional resources, r_+	4 workers
Kanban arrival rate, λ_k	9999 kanban cards per hour
Phase 1 transition time, $t_1 - t_0$	0.0079 hours = 28.44 seconds
Phase 2 transition time, $t_2 - t_1$	2.821 hours = 2 hours, 49 minutes
Transition traffic intensity, ρ	209
Total transition cost, C_{tot}	\$1361.29

This result is consistent with our rule of thumb. Since the backlog cost is relatively low, the optimal transition policy is to introduce the kanban cards as quickly as possible and if affordable, as it was in this case, purchase additional capacity to mitigate the backlog and overall transition time. By increasing the backlog cost and repeating the experiment (see Table 9), one can begin to see the tipping point at which the optimal transition policy switches from instant kanban introduction to something less is somewhere between \$4.00 and \$5.00 per order per hour in backlog.

Table 9. Manual assembly cell example optimal kanban arrival rates for varying backlog cost rates

Backlog Cost, c_r	**Optimal Kanban Arrival Rate, λ_k**
\$1.00	9999
\$2.00	9999
\$3.00	9999
\$4.00	9999
\$5.00	52.00
\$10.00	51.98

Next, we consider another system with a different set of constraints. In another part of the manufacturing system above is an automated machining cell. As part of the same value stream as the manual assembly cell, it experiences the same order arrival rate. However, it processes the orders with a single machine capable of producing 1 item about every 2.7 minutes, or 22 per hour. The machine is highly specialized and very expensive. The parameters for this stage of the system are then

Table 10. Automated machining cell example steady state parameters

Parameter	Value
Order arrival rate λ_a	20 orders per hour
Service rate μ_r	22 items per hour per machine
Number of resources r_0	1 machine
Traffic Intensity, ρ	0.91

Table 11. Automated machining cell example transition cost rates

Parameter	Value
Order deferral cost, c_d	\$1000 per order per hour
Resource cost, c_r	\$5000 per machine per hour
Inventory cost, c_i	\$1.25 per order per hour
Backlog cost, c_b	\$1.00 per order per hour

The planners are just as averse to order deferral in this case as they were in the previous example. In this case though, there is a much higher cost to augment the capacity of the system. For the sake of comparison, we leave the inventory and backlog costs the same as in the previous example. Using the hybrid model, the solver and the previously discussed constraints, we can find the optimal transition policy:

Table 12. Automated machining cell example optimal transition policy, obtained using the hybrid approximation model

Parameter	Value
Order deferral rate, λ_d	0 orders per hour
Additional resources, r_+	1 machine
Kanban arrival rate, λ_k	9999 kanban cards per hour
Phase 1 transition time, $t_1 - t_0$	0.024 hours = 1 minute, 26 seconds
Phase 2 transition time, $t_2 - t_1$	9.958 hours = 9 hours, 57 minutes
Transition traffic intensity, ρ	227
Total transition cost, C_{tot}	$63,259.07

Even at the much higher resource cost, the optimal policy calls for the use of an additional machine. Like the previous example, this solution is also consistent with the rule of thumb. The result poses the question: how high would the resource cost have to be in order to make order deferral an attractive option? By varying the resource cost, we can illustrate the region in which the increasing cost affects the nature of the optimal transition policy.

Table 13. Automated machining cell example optimal deferral rate and additional resources for varying resource cost rates

Resource Cost, c_r	Optimal Order Deferral Rate, λ_d	Optimal Number of Additional Resources, r_+
$21,000	0	1
$21,500	0	1
$22,000	5.49	1
$22,500	14.91	1
$23,000	20	0
$23,500	20	0

In the vicinity of $22,000 per machine per hour the resource cost and deferral cost are roughly in balance, resulting in a greater and greater deferral of customer orders and the continued use of an additional machine. Eventually, the resource cost becomes so great that the optimal policy is to simply stop taking customer orders while the machine, unaided, fills the kanban buffer.

These results bear out the utility of my proposed rule and shed some light on the effect of cost on the optimal Lean transition policy.

3.9 Summary

Converting a manufacturing system from push to pull and moving the inventory/order interface is an important, but poorly understood, part of Lean manufacturing. In order to understand the cost of transition, I developed a cost model for transition and described three distinct types of transition. I developed three models, a stochastic steady state approximation, a deterministic fluid flow approximation and a simulation model, that all approximate the behavior of a single stage undergoing Lean production control transition. I illustrated the differences in the models by applying them to a test case. To make optimization more efficient, I developed a hybrid model that used a linear approximation between the SSS and DFF models in lieu of the simulation model. The hybrid model made it quick and easy to estimate the optimal transition policy. I used several examples to demonstrate the utility of the new model and to illuminate interesting features of the Lean transition problem.

The lessons learned here provide a useful rule of thumb to guide practitioners. For systems in which backlog holding costs are much greater than the cost of adding resources, kanban cards should be introduced into the system all at once, and additional resource should be employed to reduce the overall length of the transition. In cases where backlog costs are significant, the hybrid model provides a method to find the optimal transition policy.

This single stage rule works well by itself, but it remains to be seen how it will hold up in the context of a multiple stage line transition. In coming chapters, I will expand the models to include multiple stages undergoing Lean transition and attempt to better understand how to optimize the many more decision variables such a system would present.

Chapter 4 Production Control Transition, Multi-Stage

One tenet of Lean Manufacturing is that one should not optimize one part of the system at the expense of the whole. Now that we have studied the transition of a single stage from push to pull production control, we must now consider how to transform an entire system. To begin, we will consider how to transform a two-stage manufacturing system from push to pull. Any lessons learned from this elementary system should be applicable to *N*-stage systems.

4.1 Problem Setup

In this section, I define the events of a multi-stage transition. I previously defined the events of a single-stage transition. A multi-stage transition consists of the same set of events occurring at each station in the system. The multi-stage transition policy determines the timing of the events at each individual stage with respect to those occurring at the other stages in the system.

Consider an *N*-stage serial manufacturing system. Each stage of the system experiences the arrival of orders. Based on where in the system each stage is located and dependent on the production control policy in effect, the orders may be customer orders that initiate production of a finished good, or they may be signals from an adjacent stage to produce some kind of sub-assembly. In practical terms, there is no difference within the stage where the order came from. The rate of order arrival was previously defined as λ_a, but we must now differentiate the order arrival rate at each stage in the system. For this parameter, and all parameters defined in the single stage case, we will denote the stage by adding another subscript, such that the order arrival rate at stage *i* is λ_{ai}. Table 14 lists the likely now-familiar steady state parameters and Table 15 lists the transition parameters for stage *i*, where $1 \leq i \leq N$. The tables include one new parameter, the material arrival rate, λ_{mi}. In the single stage case, we assumed that the system had an infinite supply of material with which to fill customer orders. This is still true for stage 1 of the multi-stage system, but stages 2 through *N* depend on the output of the stage directly upstream (*i*-1) to fill their material needs.

Table 14. Steady state performance parameters for stage *i* of an *N*-stage manufacturing system

Parameter Name	Symbol
Order arrival rate	λ_{ai}
Material arrival rate	λ_{mi}
Number of resources	r_{0i}
Processing rate per resource	μ_{0i}

Table 15. Transition performance parameters for stage *i* of an *N*-stage manufacturing system

Parameter Name	Symbol
Number of kanban cards	n_{ki}
Order deferral rate	λ_D
Kanban card arrival rate	λ_{ki}
Additional resources	r_{+i}

Recall from Chapter 3 that each stage experienced three distinct temporal events: the start of kanban card introduction, the end of kanban card introduction and the processing of the last kanban card. We will denote the first two events as t_{0i} and t_{1i} to indicate the stage to which each event refers. The third event, processing the last kanban card, is not a meaningful event in the multi-stage scenario. It will become clear why this is true in a moment.

We also define two new temporal events in the transition of each stage. As the system transitions from push to pull, each stage will first change from push control to being the push-pull interface. In push production control, customer orders are first directed to stage 1 of the system. All subsequent stages are triggered to process by the flow of material from the preceding stages. In hybrid production control, the flow of customer orders is directed at a stage in the middle of the system. This stage is the push-pull interface and it processes material only if it has a customer order. All upstream stages operate with pull production control, meaning they work only if there is room in a downstream queue, while all downstream stages operate as they always have under push control. In Lean transition, the movement of the push-pull interface is assumed to be unidirectional downstream, toward the customer. As a result, the production control of a given stage will always proceed from push to push-pull interface to pull. These two changes necessitate two new temporal events: the interface transition time, t_{xi}, and the pull transition time, t_{pi}. A complete list of the multi-stage temporal events is included in Table 16.

Table 16. Temporal event parameters for stage *i* of an *n*-stage manufacturing system

Parameter Name	Symbol
Arrival of first kanban card	t_{0i}
Arrival of last kanban card	t_{1i}
Change to push-pull interface	t_{xi}
Change to pull	t_{pi}

To develop a coherent transition strategy, we must identify the constraints on these events. First, we restrict the timing of the arrivals of the first and last kanban cards

$$t_{0i} < t_{1i} \tag{29.}$$

This reinforces common sense that we cannot finish kanban card introduction before it begins, but it goes a step further to say that kanban card introduction cannot be instantaneous. This constraint is a practical one to disallow infinite arrival rates.

Next, we say that we must introduce kanban cards stage-by-stage in order. That is, we cannot introduce kanban cards to stage 2 before stage 1, which we express as

$$t_{0i} \leq t_{0(i+1)} \tag{30.}$$

This constraint does not eliminate the possibility of starting the introduction of kanban cards to all of the stages simultaneously. We further restrict kanban card introduction by saying that it cannot be completed at any stage before it is complete at the preceding stage. We write this constraint as

$$t_{1i} \leq t_{1(i+1)} \tag{31.}$$

Again, this does not prohibit two or more stages, or all of them for that matter, from completing kanban card introduction at the same time. As discussed earlier, we can also restrict the order of the production control transition such that each stage transitions first from push production control to push-pull interface to pull by saying

$$t_{xi} \leq t_{pi} \tag{32.}$$

There is a push-pull interface in every system, even in one controlled with push. To ensure that this is true in a model, we specify that stage 1 is never operated in push production control, it can only be operated with pull or as the push-pull interface. So, we restrict stage 1 as follows:

$$t_{x1} << t_{01} \tag{33.}$$

We have not yet discussed t_2. In the single stage model, we defined t_2 as the time at which all of the kanban cards have completed processing within the stage and as the endpoint of the push-to-pull transition. The model implicitly assumed that no downstream stage was consuming the kanban inventory. Our definition of the multi-stage system makes no such assumption, so the definition of a t_2-like variable for each stage would necessarily be much more complex as kanban cards would circulate between stages throughout transition. As we look at different transition policies, we will explore new definitions of t_2.

Note that although these events are temporally constrained, I do not want to imply that the exact timing of these events might be known a priori. In fact, in this research we will consider them output rather than input parameters of a multi-stage transition. We will instead discuss and use system states as triggers for these events.

With our definition of the system complete once again, we can now discuss multi-stage transition cost.

4.2 Multi-stage Transition Cost

In the previous chapter, we identified two important queue-related transition costs, backlog and inventory, where the distinction was principally on which side of the system the queue was located. In a multi-stage system, things are no different. Each stage has an upstream queue for orders or other demand signals which we will refer to as the backlog queue, a downstream queue for completed orders, which we'll refer to as the inventory queue and a resource queue where orders that have been matched with material wait to be served. In the single stage model we assumed an infinite supply of raw materials with which to fill customer orders. This is still true of the first stage of the multi-stage system. All of the other stages, stages 2 through N however, must have an order in their backlog queue and material in the previous stage's inventory queue before they can process. In this way, the stages overlap and are inter-connected.

Recall from Section 3.2.3 that we defined single-stage inventory cost, C_i, as

$$C_i = (n_k/2)\,(t_2 - t_0)\,c_i \tag{34.}$$

This was based on a convenient simplification of the average volume of the inventory queue, whose volume increased linearly throughout the well-defined time interval. As we discussed in the previous section, neither the behavior of the queue volume nor the bounds of the per-stage transition time interval are so neat in the multi-stage case. It makes sense then to restate this definition as

$$C_I = \sum_{i=1}^{N} c_{Ii} \int_T \mathbf{Q}_{Ii}(t)\,dt \tag{35.}$$

In which we have renamed the total system inventory cost C_I, using the capital I sub-script for inventory to prevent confusion with lower case *i*, which we now use to indicate stage number. Since we no longer have an unambiguous definition of the end event, we replace t_1 and t_2 in the equation with *T*, the transition interval. We also introduce $Q_{Ii}(t)$, the number of orders in the inventory queue in stage *i* at time *t*. Note the similarity of this equation to our previous definition of backlog cost, which we modify here for the multi-stage case.

$$C_B = \sum_{i=1}^{N} c_{Bi} \int_T \mathbf{Q}_{Bi}(t)dt \qquad (36.)$$

Resource cost can then be defined as

$$C_R = \sum_{i=1}^{N} c_{Ri} \int_T r_{+i}(t)dt \qquad (37.)$$

Again we adopt the capitalized subscript to indicate total system cost versus the per-stage costs we used in the previous chapters.

Backlog cost and order deferral cost, as defined in our initial model, seemed somewhat unrelated, but in a multi-stage model, it is clear to see that they share a strong common relationship to the incoming stream of customer orders. Customer orders represent demand for finished goods, not subassemblies, so the flow of customer orders cannot reasonably be deferred on a per-stage basis. As a result, we define the order deferral cost as

$$C_D = c_D \int_T \lambda_d(t)dt \qquad (38.)$$

which is absent the summation-by-stage evident in the previous cost terms. Similarly, backlog cost is related to the number of customer orders waiting to begin processing.

Note that these cost equations permit both of the mitigation mechanisms, the number of additional resources and the order deferral rate, to be non-stationary during the transition. The reason for this will become clear later, when we discuss transition policies.

Summing up the cost terms, we can now define transition cost for the multi-stage model as

$$C_T = C_I + C_B + C_R + C_D \tag{39.}$$

or

$$C_T = \sum_{i=1}^{N} c_{Ii} \int_T \mathbf{Q}_{Ii}(t)dt + \sum_{i=1}^{N} c_{Bi} \int_T \mathbf{Q}_{Bi}(t)dt + \sum_{i=1}^{N} c_{Ri} \int_T r_{+i}(t)dt + c_D \int_T \lambda_d(t)dt \tag{40.}$$

Now that we once again have a transition objective to minimize, let us now discuss transition strategies.

4.3 Multi-stage Transition Strategies

Using the constraints developed in the previous sections, we can develop an infinite number of transition strategies for even very simple systems, but we will begin by looking at two useful, unmitigated special cases, All-at-Once and One-by-One.

4.3.1 All At Once (AA1) Transition Strategy

What little advice the Lean literature has to offer on the subject of production control transition includes the very general guideline that it should be done as quickly as possible. The previous chapter supports this theory in that the least expensive so-called simple transition policy was that in which all of the kanban cards were introduced into the system instantaneously. It might follow then that introducing all of the kanban cards at every stage in a multi-stage system might be an inexpensive, simple transition control strategy. I call this policy the All At Once strategy, or AA1 strategy, and define it in terms of our new parameters as follows:

$$t_{0\mathrm{i}} = 0 \quad \text{for } 1 \le i \le N \tag{41.}$$

$$t_{1\mathrm{i}} \to 0 \quad \text{for } 1 \le i \le N \tag{42.}$$

which indicates that all of the kanban cards are introduced at each stage, nearly instantaneously. Then, we define the transition timing as

$$t_{\mathrm{x}1} << 0 \tag{43.}$$

This means that the push-pull interface must begin within the system of interest. According to the constraints, stage 1 must be the first stage to become the push-pull interface, so we say its transition to push-pull interface event happens before the Lean transition begins.

$$\mathbf{Q}_{I1}\left(t_{p1}\right) = n \tag{44.}$$

During the transition, the inventory queues will fill, starting at stage N and working back to stage 1, so the inventory queue at stage 1 is where the transition event is triggered. The push pull interface moves to stage 3 when all of the kanban cards have been processed and are sitting in the stage 1 inventory queue.

$$t_{p1} = t_{pi} = t_{xN} = t_{xi} \text{ for } 2 \leq i \leq N\text{-}1 \qquad (45.)$$

So, true to its name, in the AA1 transition strategy, the kanban cards are introduced to the system all at once, and when they are all matched with inventory, all of the transition events occur at the same time.

4.3.2 One By One (1B1) Transition Strategy

We may instead choose to transform each stage of the system in turn, essentially doing exactly what we did in the single stage study, moving the push-pull interface downstream as each kanban queue fills completely. I call this transition strategy One By One, or 1B1.

The previous chapter showed that careful selection of mitigation techniques can minimize the transition cost of a single stage. It also demonstrated that different types of manufacturing processes are subject to different transition constraints which call for different transition policies. In a multi-stage system it is reasonable to assume there will be a variety of processes and that a unilateral transition policy like AA1 may not achieve the lowest cost. Instead, let us consider transforming each stage of the system in turn, optimizing the transition policy of each transformation and completing each stage before moving to the next.

For this policy, we will re-use the end-of-transition variable from the single stage case, but add a subscript to indicate to which stage it refers. We define t_{2i} as the time at which the transition of stage i is complete, when all of the kanban cards have been processed and are waiting in inventory. For timing, we define this policy like this

$$t_{01} = 0 \tag{46.}$$

$$t_{0(i+1)} = t_{2i} \quad \text{for } 1 \leq i \leq N\text{-}1 \tag{47.}$$

This equations says that the transition begins at each stage when the transition at the previous stage is complete.

$$\mathbf{Q}_i(t_{2i}) = n_\mathrm{i} \text{ for } 1 \leq i \leq N \tag{48.}$$

This means that the transition of each stage is complete when the inventory queue at the stage contains all of the kanban cards.

$$t_{\mathrm{x}i} = t_{0i} \quad \text{for } 1 \leq i \leq N \tag{49.}$$

$$t_{\mathrm{p}i} = t_{2i} \text{ for } 1 \leq i \leq N \tag{50.}$$

These two equations say that each stage becomes the push-pull interface at the same time that the transition begins, and that it changes to pull production control when the transition is complete.

This means that kanban cards are initially introduced to stage 1 and it is allowed to process all of them. When all of the kanban cards are completed at stage 1, kanban card introduction begins in stage 2. At the same time, the push-pull interface moves from stage 1 to stage 2 and stage 1 begins to operate under pull production control.

To compare the effectiveness of these two policies, we need to apply them to a simple example system undergoing Lean transition.

4.4 Two Stage System Example

For a simple example system on which to test transition policies we revisit the single stage model first introduced in Section 3.2.4. We can easily imagine a two stage system in which each of the two stages is identical to this example. Figure 13 illustrates this example system.

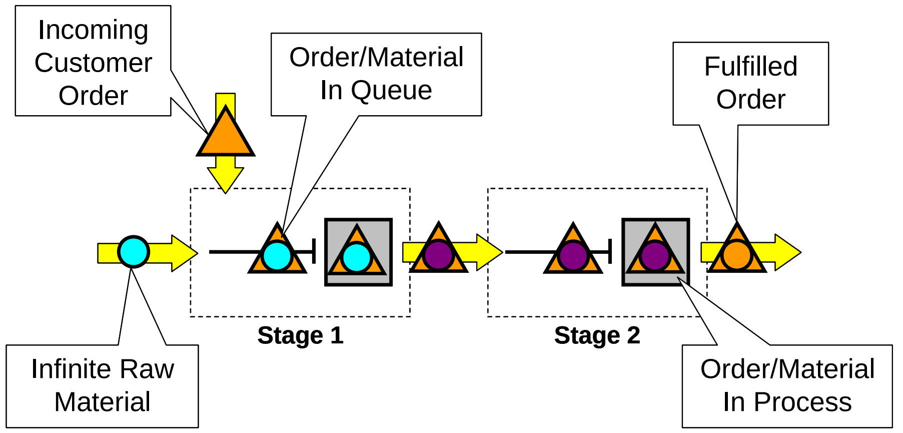


Figure 13. A two stage system operating under push production control. In this figure, incoming customer orders (orange triangles) are matched with raw materials (blue circles), which are processed in stage 1 to become intermediate assemblies (purple circles), which are then processed in stage 2 to become finished goods (orange circles). The customer order follows the material through the system because this system is under push production control.

The example system consists of two workstations. Stage 1 receives customer orders, which are matched with raw material from an infinite supply. The material/order then waits in queue for service at stage 1. When service is complete, the material/order moves to stage 2, where it again waits for service. In this system the material and order stay together throughout. When service is complete at stage 2, the now-fulfilled customer order leaves the system. As in Chapter 3, the system parameters are as shown in Table 17.

Table 17. **Two stage system parameters**

Parameter	**Value**
Number of resources *r*	10 per stage
Service rate μ_r	10 items per unit time per resource
Order arrival rate λ_a	75 items per unit time

The system as shown is under push production control. Each stage is allowed to serve all of the material/orders in its queue without requiring any other trigger. We wish to transform this system to pull production control, where the final system configuration is shown in Figure 14, with 240 kanban cards in each of the two stages.

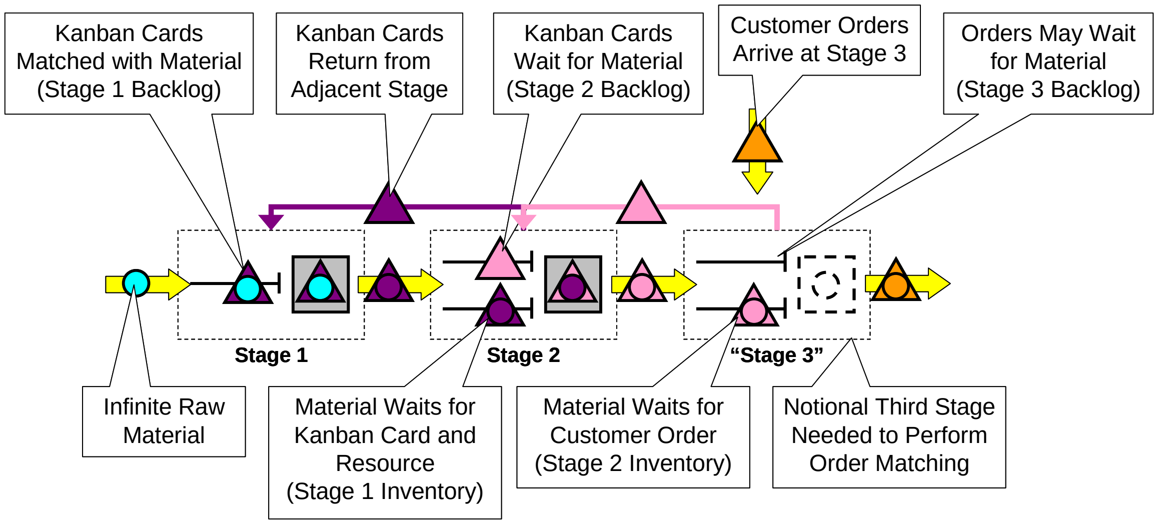

Kanban Cards Matched with Material (Stage 1 Backlog)
Kanban Cards Return from Adjacent Stage
Kanban Cards Wait for Material (Stage 2 Backlog)
Customer Orders Arrive at Stage 3
Orders May Wait for Material (Stage 3 Backlog)
Stage 1
Stage 2
"Stage 3"
Infinite Raw Material
Material Waits for Kanban Card and Resource (Stage 1 Inventory)
Material Waits for Customer Order (Stage 2 Inventory)
Notional Third Stage Needed to Perform Order Matching

Figure 14. A two stage system operating under pull production control. In this figure, incoming customer orders (orange triangles) are matched with finished goods in inventory (pink circle with triangle), which causes a finished goods kanban card (pink triangle) to travel back to stage 2 and signal the production of a finished good from an intermediate assembly in inventory (purple circle with purple triangle). This causes an intermediate assembly kanban card (purple triangle) to travel back to stage 1 and signal the production of an intermediate assembly from raw materials (blue circles).

In its final, pull, configuration, the system requires a notional third stage. The extra station matches incoming orders, which now enter the system there, with material in the stage 2 inventory and thereby “pull” production in stage 2. Both of the original two stages operate by matching material with a kanban card before starting service. The material inventories in stages 2 and 3 are bundled with kanban cards, which are separated when the material is used and sent back to the originating stage.

In this chapter we apply the two transition strategies proposed in the previous sections, examine their behavior and determine which one results in lower transition cost. We use a simulation model of the system.

4.4.1 Modeling a Two Stage System

A simulation model of the two-stage example was created using Arena. Figure 15 shows the Arena user view of the model. This model, like those used in Chapter 3 was built to provide plots of system performance over time in order to better see the dynamics of transition. The plots in the following sections are average values of 10 replications, sampled every 0.01 time units.

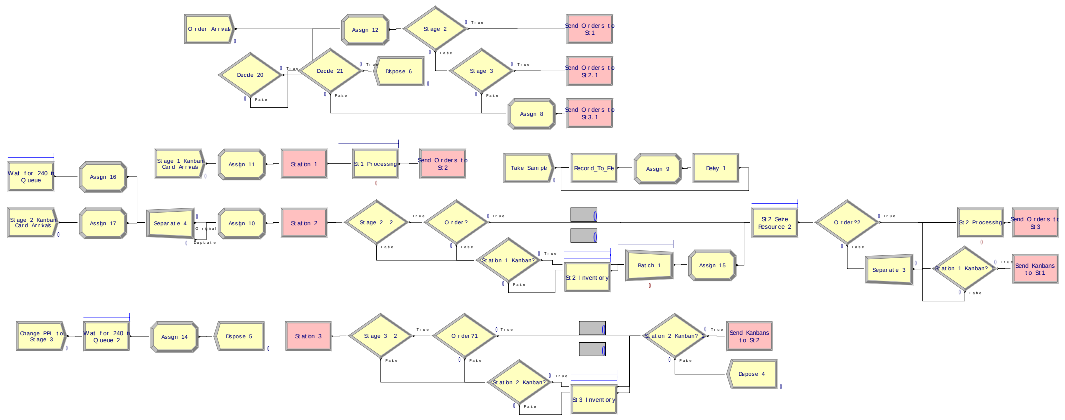


Figure 15. A model of a two-stage system, built using Arena

4.4.2 Modeling an AA1 Transition

The model output for a simulation of an AA1 transition is shown in Figure 16. The plot tracks the number of orders or material in each queue in the system versus time.

At *t*=2, the kanban cards are introduced to both stages, resulting in predictably large spikes in both backlog queues. As we saw in the single stage example in Chapter 3, the stage 2 backlog queue decreases quickly as stage 2 processes material that accumulates as stage 2 inventory. However, unlike the single stage model, the stage 1 backlog queue continues to fill. For each kanban order that stage 1 processes, stage 2 immediately consumes the material and sends the stage 1 kanban card back to the stage 1 backlog queue. Since the two stages process at the same rate, stage 1 keeps pace with stage 2, but the return of kanban cards and the continued arrival of new customer orders increases the stage 1 backlog queue well beyond the initial spike. When stage 2 completes processing its kanban cards and stops withdrawing material from the stage 1 inventory, then that inventory begins to grow. When all of the stage 1 kanban cards have become stage 1 inventory, the push-pull interface moves to stage 3 because all of the stage 2 kanban cards have become stage 2 inventory. However, more than 50 customer orders are still in stage 1 backlog queue. . After the push-pull interfaces moves, at about $t = 10$, the new orders quickly consume the stage 2 inventory, which sends stage 2 kanban cards back to the stage 2 backlog queue, so stage 2 begins to consume the stage 1 inventory. Both inventories are again completely consumed somewhere between $t = 10$ and $t = 15$, and customer orders start to build up in the stage 3 backlog queue. The stage 2 backlog queue was receiving a steady flow of kanban cards returning from the stage 2 inventory as well as the approximately 50 customer orders that were at stage 1 when the push-pull interface moved. At about the same time that the stage 2 inventory is depleted, the flow of customer orders from stage 1 also ceases, and stage 2 begins working through its now-sizable backlog queue. Stage 1 also begins to catch up, and both stages decrease their backlog queues. They complete the waiting customer orders, and their kanban cards accumulate as inventory. By about $t = 55$, the system has reached a new steady state.

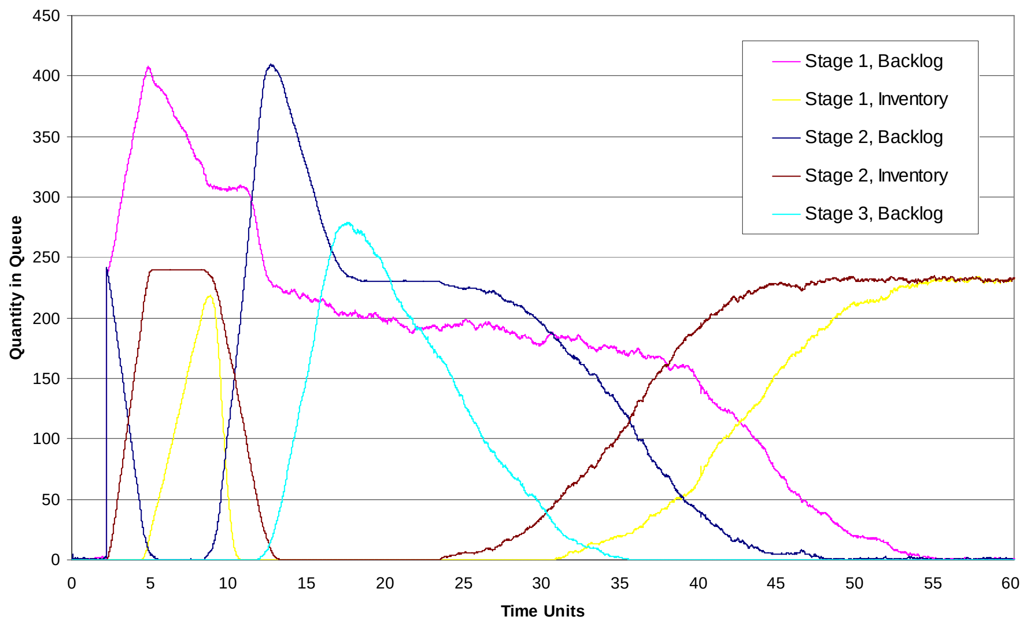


Figure 16. A plot of quantities in queue versus time for a two-stage, AA1 transition

4.4.3 Modeling a 1B1 Transition

Figure 17 shows the equivalent plot for the 1B1 strategy. As in the previous example, the system begins in steady state, under push production control. At *t* = 2, kanban cards are introduced to stage 1 only, which results in a spike of orders in the stage 1 backlog queue. Just as in the single-stage model, the kanban cards are quickly converted into stage 1 inventory. In fact, the plot closely resembles the single stage model until stage 1 finishes processing its kanban cards. In the B1B strategy, this triggers the push-pull interface to shift to stage 2, at which point all of its kanban cards are then introduced. Since stage 1 and stage 2 have the same number of kanban cards, the stage 2 kanban cards are all immediately matched with the stage 1 inventory, emptying that queue. As stage 2 processes the matched orders, the stage 1 kanban cards begin returning to the stage 1 backlog queue, where the influx temporarily halts its progress reducing the customer orders there. As in the AA1 strategy, stage 2 is once again subject to an existing backorder while receiving both new customer orders and older orders now filtering through the stage 1 backlog, resulting in a peak of backlog of roughly the same magnitude as that seen in AA1. Since the kanban cards were the first to arrive in the stage 2 backlog queue, they are processed first and quickly accumulate as stage 2 inventory, triggering the push-pull interface to move again to stage 3. Just as it did when the push pull interface was moved to stage 2, the new customer orders quickly consume the stage 2 inventory at the push pull interface, creating a stream of kanban cards that return to and greatly increase the stage 2 backlog queue, and the incoming customer orders begin to build up at the stage 3 backlog queue. In this case, both preceding stages have largely worked through their backlogs of old customer orders and can now focus on processing kanban card demand. For 10 or 15 time units, the two stages work in near-parallel to fulfill the demand originating downstream. Slowly, the backlog queues are emptied and the stage 1 and stage 2 inventories accumulate. At about $t = 50$ the system settles into a new steady state.

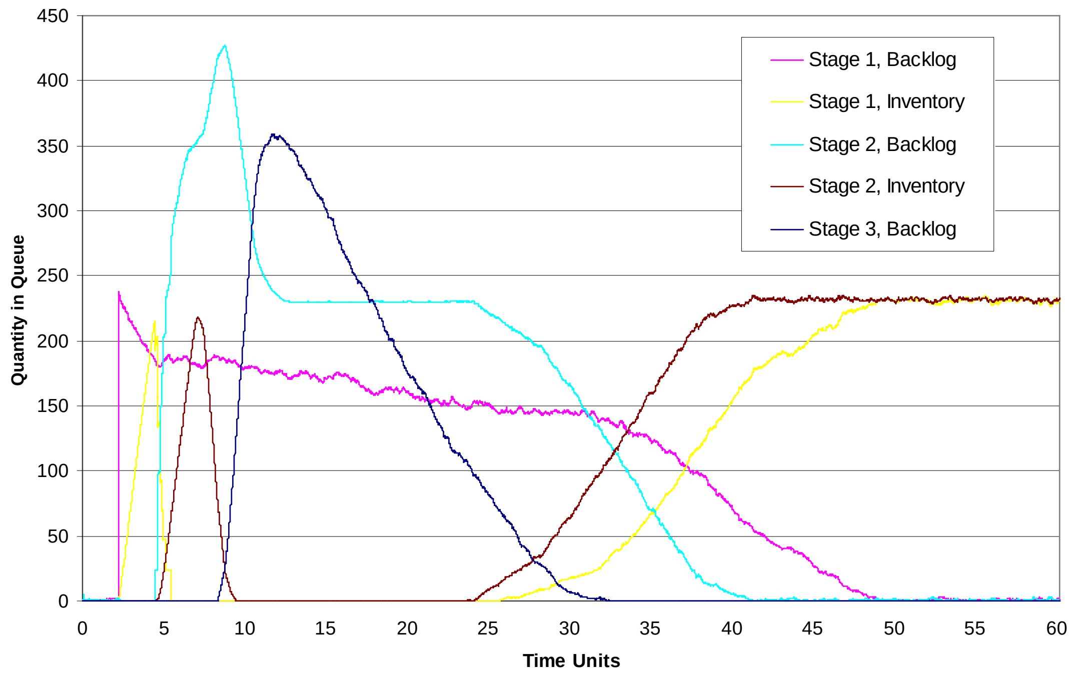


Figure 17. A plot of quantities in queue versus time for a two-stage, 1B1 transition

4.4.4 Two Stage Transition Cost

Using the data collected from the trials above, one can easily evaluate the cost of transition. In the previous chapter, measuring the transition cost was relatively straightforward because we were using a well-defined event to bound the transition time. In the multi-stage case, as we discussed earlier, the per stage and overall transition times are harder to define. Setting aside this dilemma for the moment, we can look at the cumulative costs of the two strategies we modeled. Table 18 lists the cumulative costs, where both the backlog and inventory holding costs are $1 per order per time unit.

Table 18. Cumulative costs for AA1 and 1B1 strategies

Strategy	Cumulative Transition Cost		
	@ $t = 20$	@ $t = 40$	@ $t = 60$
All at Once, AA1	\$12,153	\$22,008	\$31,239
One By One, 1B1	\$11,304	\$20,181	\$29,456
Difference	\$849	\$1827	\$1783

The 1B1 strategy is superior throughout the transition, but builds most of its advantage over the AA1 strategy in the first half of the transition. Looking back at Figure 16 and Figure 17, the period between $t = 0$ and $t = 20$ is marked by large buildups of backlog as the system absorbs the surge of kanban cards. The backlogs in stages 2 and 3 share similar trajectories in both strategies. Stage 2 comes to a sharp, tall peak that descends quickly, levels out, and then continues to descend more moderately in both scenarios. Stage 3 similarly peaks quickly, and then follows a more moderate path downward. The principle difference between the strategies is seen in the trajectory of the stage 1 backlog. In the AA1 strategy, the stage 1 backlog climbs steadily, following a similar trajectory to that of stage 2, peaking early, then falling, leveling out, then falling more slowly over the rest of the transition. In the 1B1 strategy, however, the stage 1 backlog begins with the surge of kanban cards, but falls throughout the transition. This marked difference in behavior explains the difference in cost.

4.4.5 Mitigating Transition Cost; Deferral and Resources

In Chapter 3 we experimented with techniques to mitigate the transition cost for a single stage. To demonstrate their effectiveness in the multi-stage case, we re-run the strategies we just discussed, but with mitigation as prescribed by the optimization of the last chapter. In Section 3.7 it was shown that the optimal transition policy for our stage single stage was full deferral of customer orders and doubling the resource capacity from 10 units per stage to 20. To better understand the effects of each of these measures, we can re-run the trials of both strategies, but with mitigation as described in Table 19.

Table 19. Mitigation techniques applied to two stage transition

Mitigation Trial	Order Deferral, λ_d	Additional Resources, r_+
Deferral	75	0
Resources	0	10
Deferral and Resources	75	10

Figures 18 and 19 are plots of the deferral mitigation technique applied to the two-stage system for the AA1 and 1B1 strategies, respectively.

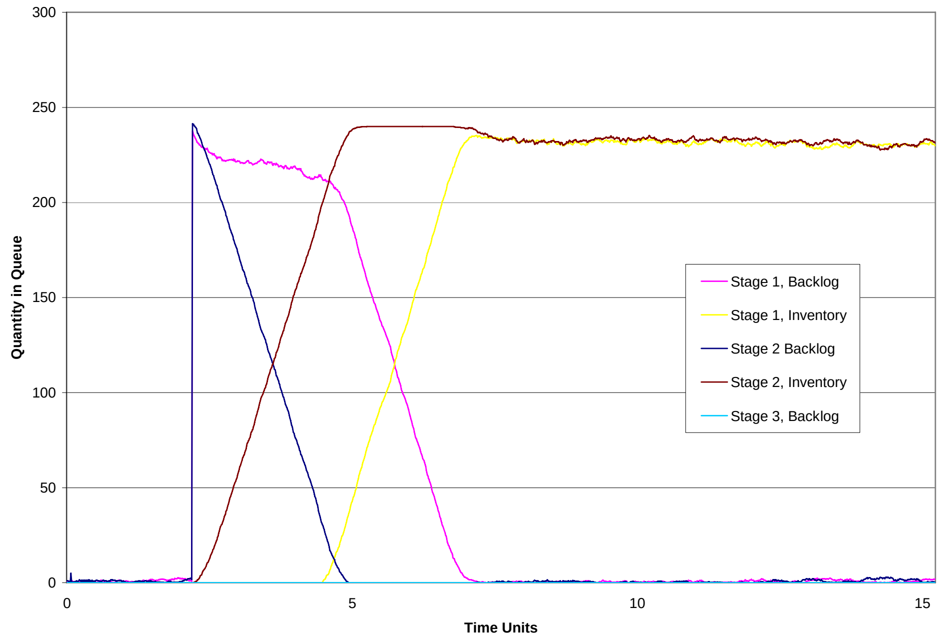


Figure 18. A plot of quantities in queue versus time for a two-stage, AA1 transition, with order deferral mitigation, $\lambda_d = 75$

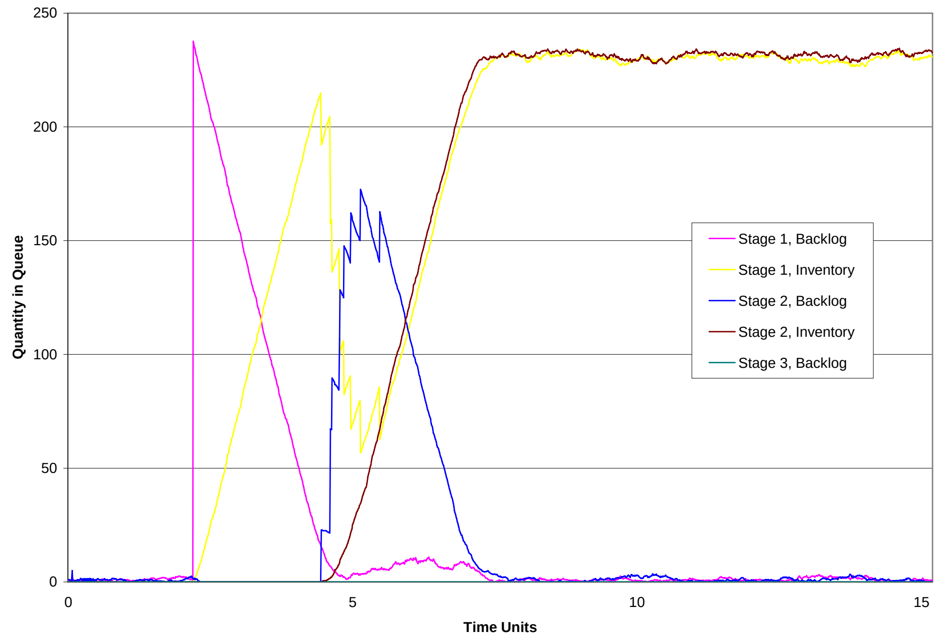


Figure 19. A plot of quantities in queue versus time for a two-stage, 1B1 transition, with order deferral mitigation, $\lambda_d = 75$

These plots show that the deferral-mitigated strategies exhibit much different behavior from their unmitigated counterparts. Although we still see the initial spikes of backlog in both stages, they dissipate rather quickly and never exceed the quantity of the initial surge of kanban cards. In general, these strategies take much less time to reach the new steady state.

Figures 20 and 21 are plots of the resources mitigation technique applied to the two-stage system for the AA1 and 1B1 strategies, respectively.

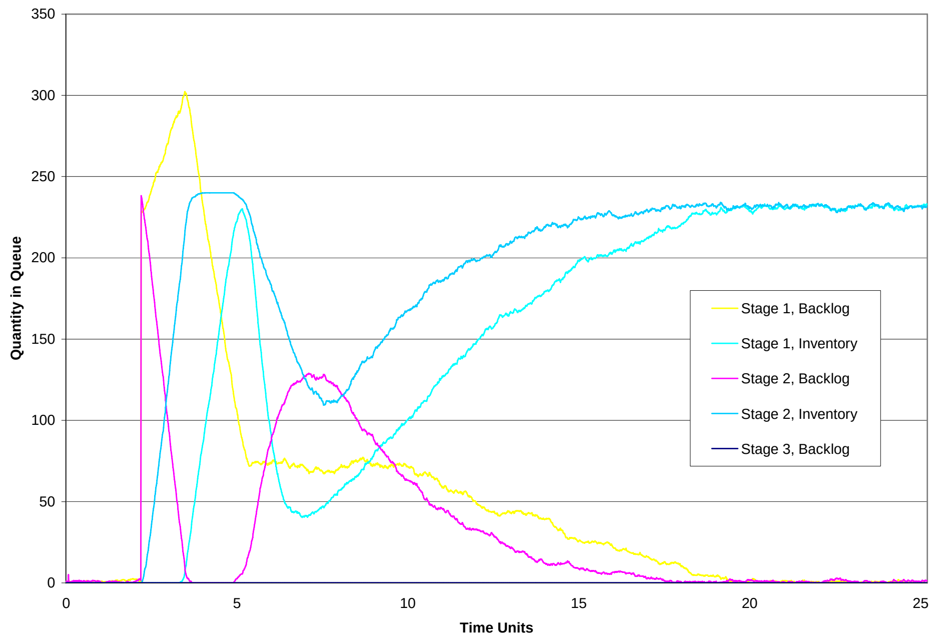


Figure 20. A plot of quantities in queue versus time for a two-stage, AA1 transition, with resource mitigation, $r_+ = 10$

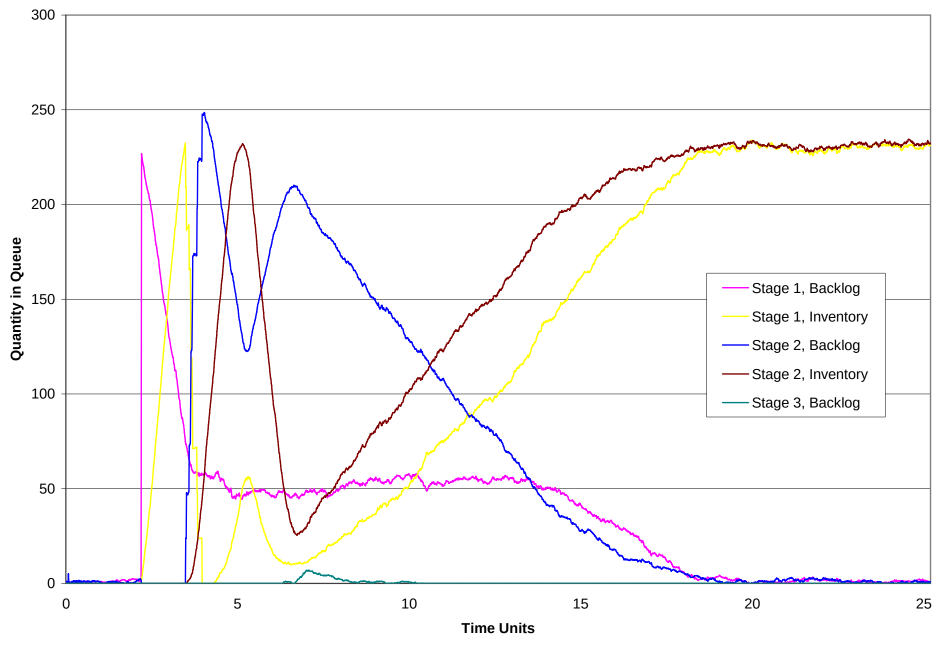


Figure 21. A plot of quantities in queue versus time for a two-stage, 1B1 transition, with resource mitigation, $r_+ = 10$

The resource-mitigated strategies behavior is more similar to the unmitigated trials than the deferral-mitigated strategies were. Both exhibit the same complex interactions between the stages, but the effects are moderated by the additional resources, which lower the traffic intensities somewhat. The backlog peaks are not as pronounced and the inventories almost never empty completely.

Figures 22 and 23 are plots of the resources and deferral mitigation techniques applied simultaneously to the two-stage system for the AA1 and 1B1 strategies, respectively.

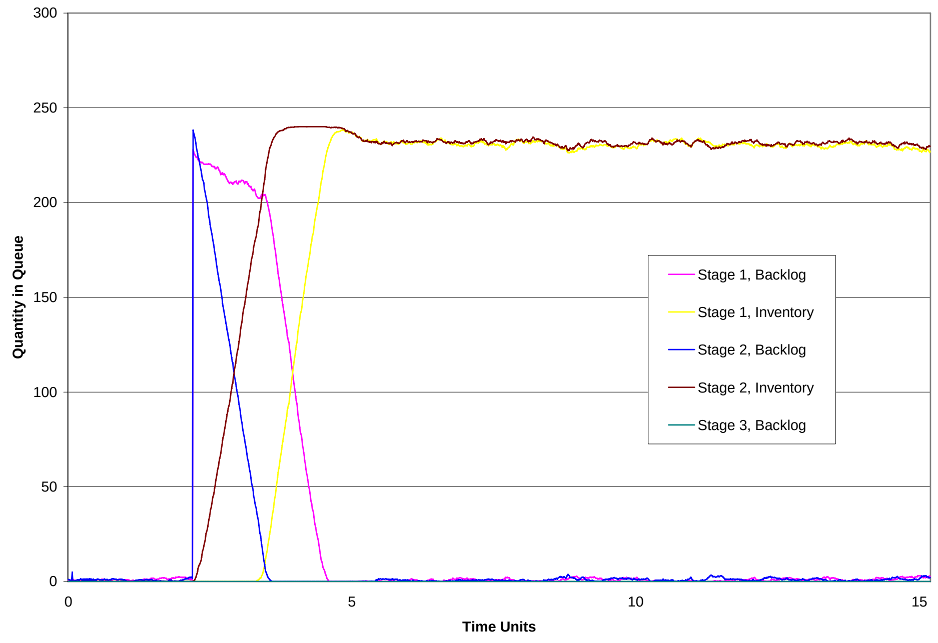


Figure 22. A plot of quantities in queue versus time for a two-stage, AA1 transition, with deferral and resource mitigation, $\lambda_d = 75$ and $r_+ = 10$

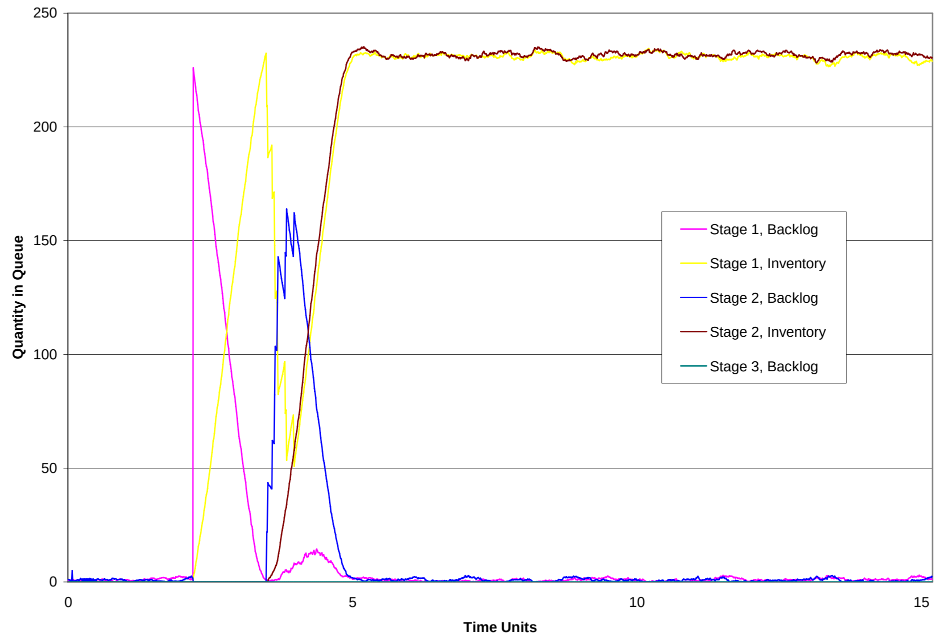


Figure 23. A plot of quantities in queue versus time for a two-stage, 1B1 transition, with deferral and resource mitigation, $\lambda_d = 75$ and $r_+ = 10$

The behavior of the strategies mitigated with both order deferral and additional resources are very similar to those of the deferral-mitigated trials, but the transitions conclude much more quickly due to the extra capacity.

Without explicitly defining a transition completion event, we estimate the transition cost based on the point at which each system reaches its new steady state. We again use unit rates for all of the cost factors. Table 20 shows the transition costs for the three mitigation techniques in each of the two transition strategies.

Table 20. Transition costs of deferral and resource mitigated AA1 and 1B1 transition strategies

Mitigation Trial	All at Once, AA1	One By One, 1B1
None	\$28,917 (@ $t = 55.0$)	\$24,814 (@ $t = 50.0$)
Deferral	\$26,827 (@ $t = 7.0$)	\$17,661 (@ $t = 7.2$)
Resources	\$80,941 (@ $t = 20.0$)	\$69,252 (@ $t = 20.0$)
Deferral and Resources	\$16,175 (@ $t = 5.0$)	\$11,511 (@ $t = 5.0$)

These results show that order deferral has the most dramatic impact on both transition time and cost. The resource-mitigated trials required more than twice as much time to reach their new steady state, accumulating proportionally higher cost as a result. The 1B1 strategy proved once again to be the superior alternative in each of the three mitigation trials.

4.4.6 Summary

In this chapter, we studied the behavior of multi-stage Lean transitions by considering a specific two stage system. A multi-stage transition was defined in terms of events and their relative timing. Based on this framework, two transition strategies were proposed; the All At Once (AA1), in which all of the stages begin the transition at the same time, and the One By One (1B1), in which the each stage begins its transition as the preceding stage finishes. It was shown that the behaviors of the backlog and inventory queues, which are important components of transition cost, are very complex. In general, the 1B1 strategy minimized the transition cost because the AA1 strategy allowed large initial spikes of orders in the backlog queues while the 1B1 strategy suppressed them. Two mitigation techniques, the deferral of customer orders and the addition of resource capacity during the transition, were evaluated to see their effect on multi-stage transition cost. Both techniques reduced overall cost, but order deferral proved to be more effective. Order deferral, combined with the 1B1 transition strategy, proved to reduce overall transition cost significantly in this case.

The results provide clear evidence of the complexity of the transition from push to pull in a multi-stage system and the potential impact of different strategies and mitigation techniques. More work is needed to understand this complex behavior more fully.

In this investigation, we did not vary the kanban card introduction rate, a cost reduction technique demonstrated when we considered the single stage system in Chapter 3. To continue down this path of exploration, this option should be exercised. We used an easily measured, unambiguous event to trigger the start of each stage transition in the

1B1 strategy, but there are many other criteria that one could use as a trigger. These other triggers should be examined to determine if this strategy can be further improved. The system considered in this chapter had two identical stages. More work is needed to evaluate the performance of the transition strategies in other types of manufacturing systems. Little attention was paid to a concluding event or criteria to mark the end of transition. This topic deserves further investigation to more fairly compare different transition policies. Finally, the optimization tools used successfully in the single stage case should be employed to optimize the cost of multi-stage transition.

Chapter 5 Simulation Modeling of Production Control

The previous chapters demonstrated that the cost of Lean transition can be mitigated if the transition is carefully controlled. It was shown that simulation modeling can be a powerful tool to find the optimal transition policy, especially when the on-board optimization utilities are brought to bear. However, the systems used as examples were relatively simple. In contrast, the simulation models of those same systems were necessarily very complex, due to the fact that off-the-shelf simulation software does not easily support pull production control. Since the goal of this research is to develop robust transition control policies for complex systems, a new class of simulation software objects is required in order to fully model hybrid production controls. It must enable parametric description of production control so that the built-in optimization utilities can be used to find the optimal transition policy. This chapter describes a new logical model of a manufacturing process, the Multi-Flow Modeling Paradigm (MFMP). It also introduces a hierarchical set of new simulation modeling objects, the Production Control Framework (PCF), based on this paradigm. The PCF makes it possible to model production control parametrically and therefore facilitate automated search of the production control domain.

5.1 Modeling the Hidden Factory

Although manufacturing systems are designed primarily to process raw materials into finished goods, the modern manufacturing system increasingly processes information as well. Vollmann, Berry, and Whybark (1997) describe this dualism as the "hidden factory." They describe a manufacturing firm as being comprised of two factories, one that processes parts, while the other "hidden factory" processes transactions on paper and through computers. Most of these transactions are signals for a process to "go". These signals, which I will refer to as "demand" are an important element missing from previous models of manufacturing systems.

A manufacturing system contains multiple manufacturing processes. A manufacturing process requires materials, resources, and demand (and time, but we will ignore this element for now). Materials are the physical would-be products of a manufacturing system, and they follow a unidirectional path through the manufacturing system. Resources are the finite capacity components of a manufacturing system. Resources enter the system, but do not leave. They follow a cyclic path through a series of system states. Demand is also necessary to carry out a manufacturing process. Demand may follow a unidirectional or cyclic path, depending on the production control policy of the system.

5.2 Multi-Flow Modeling Paradigm

Traditionally, a manufacturing process was modeled as a single-server queueing model. In such a model, customers arrive at the server and, if the server is idle, they are served for some processing time. If the server is busy, the customers wait in a queue for service. The system variables are few and very simple; server state is either idle or busy, and the queue contains 0 to ∞ customers. Figure 24 shows a simple single-server queueing model.

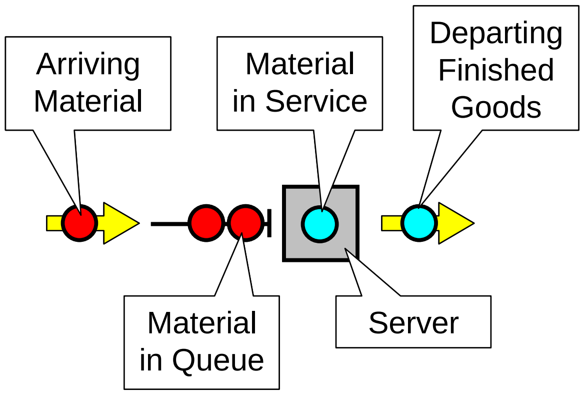


Figure 24. A single server queueing system (Law and Kelton, 2000). In this figure, material (red circles) arrives at the server where they waits in a queue for service. The server processes the material, converting it finished goods (blue circles), which depart the system.

While this model is simple, it can be combined with other such models to represent very complex systems (Law and Kelton, 2000). However, it cannot easily model pull production control because it cannot accommodate demand. A new abstraction is required to model demand-driven manufacturing systems.

To answer this need, I developed the Multi-Flow Modeling Paradigm (MFMP), an extension of the single server queueing model, designed specifically for the manufacturing domain. It is based on the idea that each process is divided into two stages, storage and service. Looking again at the single server queueing model (see Figure 25) we can identify the two stages.

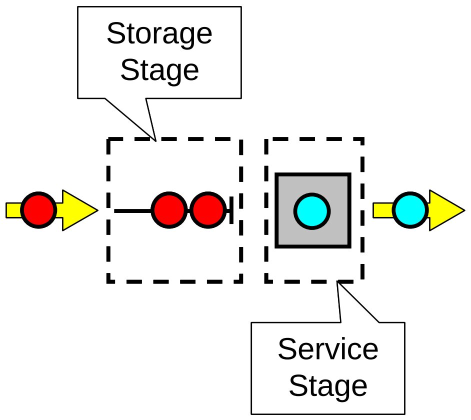


Figure 25. A single server queueing system with storage and service stages identified.

Through these stages, three types of components flow: material, resource and demand. The three types are distinguished by how their flows converge and diverge before and after each stage of the process. A manufacturing system can be characterized by the flow of these components through manufacturing processes. Material generally flows through a system in a unidirectional manner. Individual material flows may converge in an assembly process. Resource flows in repetitive trajectories within the system as they cycle between busy and idle states. Demand can arrive from external sources and from within the system. Demand follows both unidirectional and cyclic trajectories, based on the production control policy.

We begin to develop the MFMP by looking at a useful variant of the single server queueing model, an assembly station model. In such a model, two or more queues store different types of components until a certain quantity arrives and a server is available. For example, a process in which a stool is assembled might require a seat, three legs and an assembly operator before service could begin. Figure 26 shows a model of an assembly station configured for stool assembly.

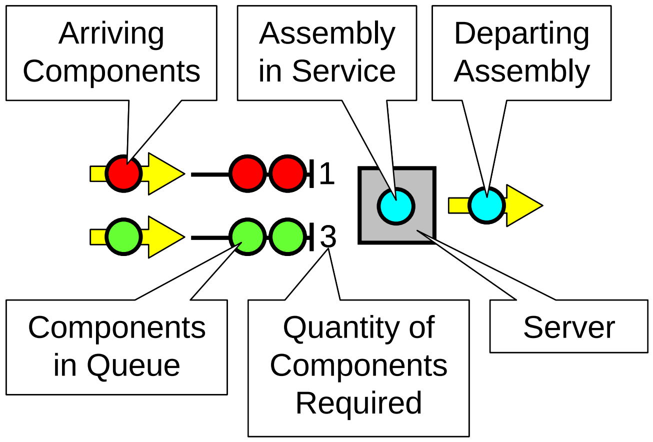


Figure 26. A two queue assembly station model. In this figure, the components of a stool, legs (green circles) and seats (red circles) arrive at the station and wait in queue to be assembled. When there is a seat, three legs and an available server, the components are processed by the server into a stool (blue circle), which departs the system.

The assembly station model can be further abstracted and simplified when one considers that the server itself is a component of the process and that service time is actually spent in yet another queue. However, the difference between the material components and the server, or resource component is in where it goes after service is complete. As in the previous model the material component, the assembly, departs the system, but the resource component, the server, circles back around to its original queue. Figure 27 illustrates such a model. In the figure, three server/resource components are pictured. This is mostly for the sake of illustration, but it could represent a station with three processing resources. This type of model makes the number of servers easy to change by adding or removing resource components from the model, but retains much of the assembly station's original operating logic. It actually simplifies the logic somewhat.

Where before it had to check the state of two different data objects (queues and servers), in this new model, the queues contain the entire system state.

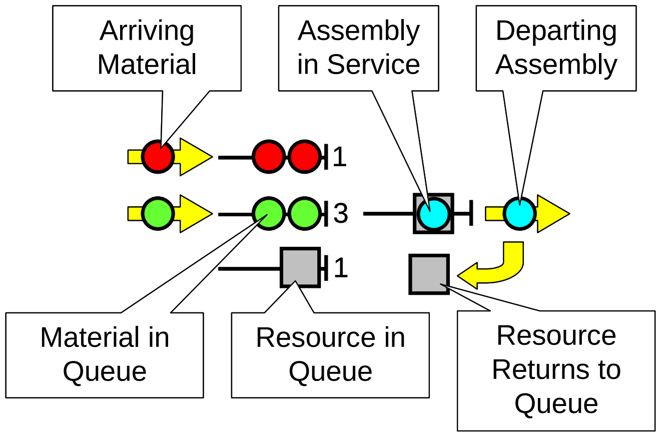


Figure 27. A three queue assembly station model with resource components and service queue. In this figure, the components of a stool, legs (green circles) and seats (red circles) arrive at the station and wait in queue to be assembled. An additional queue contains resource components which represent an available server. When there is a seat, three legs and a resource component available in the queues, the components are processed by the server into a stool (blue circle), represented by a time period in which they are stored together in a service queue. The stool departs the system and the resource component returns to its queue.

We then consider that demand is also a component of the process. Extending this model just a bit more yields a four server assembly station model in which the demand components have their own queue, and for which there is a required quantity to initiate service. Figure 28 illustrates this model. Like the resource component, the path of the demand component is not as simple as that of the material. In fact, it has three possible paths, corresponding to the three production control policies to which this process might be subject.

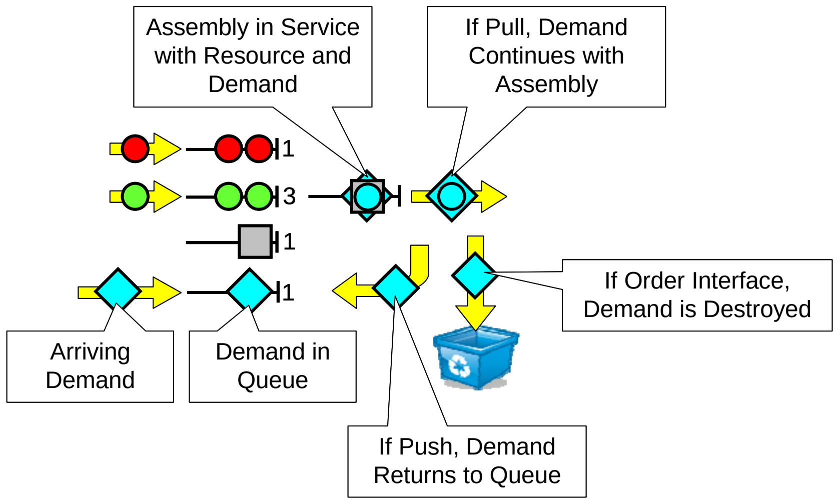

Assembly in Service with Resource and Demand
If Pull, Demand Continues with Assembly
1
3
1
1
If Order Interface, Demand is Destroyed
Arriving Demand
Demand in Queue
If Push, Demand Returns to Queue

Figure 28. A four queue assembly station with resource and demand queues, showing demand paths. In the figure, the components of a stool, legs (green circles) and seats (red circles) arrive at the station and wait in queue to be assembled. An additional queue contains resource components which represent an available server. Another queue contains demand components (kanban cards, perhaps). When there is a seat, three legs, a resource component and a demand component available in the queues, the components are processed by the server into a stool (blue circle), represented by a time period in which they are stored together in a service queue. The stool departs the system, the resource component returns to its queue and the demand component follows one of three paths, depending on the production control policy in effect.

If the production control policy is Pull, then the demand component continues to the next process with the resulting material component. If the policy is Push, the demand component, like the resource component, circles around and rejoins its original queue. If this process is the Customer Order interface, then all of the downstream processes are Push, meaning that those stations already have their own circling demand components. As a result, the demand component from this process is not needed and leaves the system entirely.

There are other manufacturing-specific behaviors we can model within this evolved paradigm. In an assembly process, many “constituent” material components may be combined into a single “assembly” material component. It is easy to rationalize that the components become a permanent “batch”, but in an explicit model like this, where do the extra constituents go? Further, as in the figure above, where pull production control results in a temporary batch of material and demand, when and where does that batch separate and what happens to the components? In addition, the behavior of resources is not always as simple as our model presently indicates. Some resources operate this way, cycling between idle and busy states, staying within the same process. However, in a modern manufacturing system, some resources travel along with material through some or all of the system. Examples of such a resource are fixtures, jigs and other tooling that hold a product together or otherwise facilitate transport between processes. Figure 29 illustrates the fully realized MFMP model, which accommodates all of these domain-specific flows.

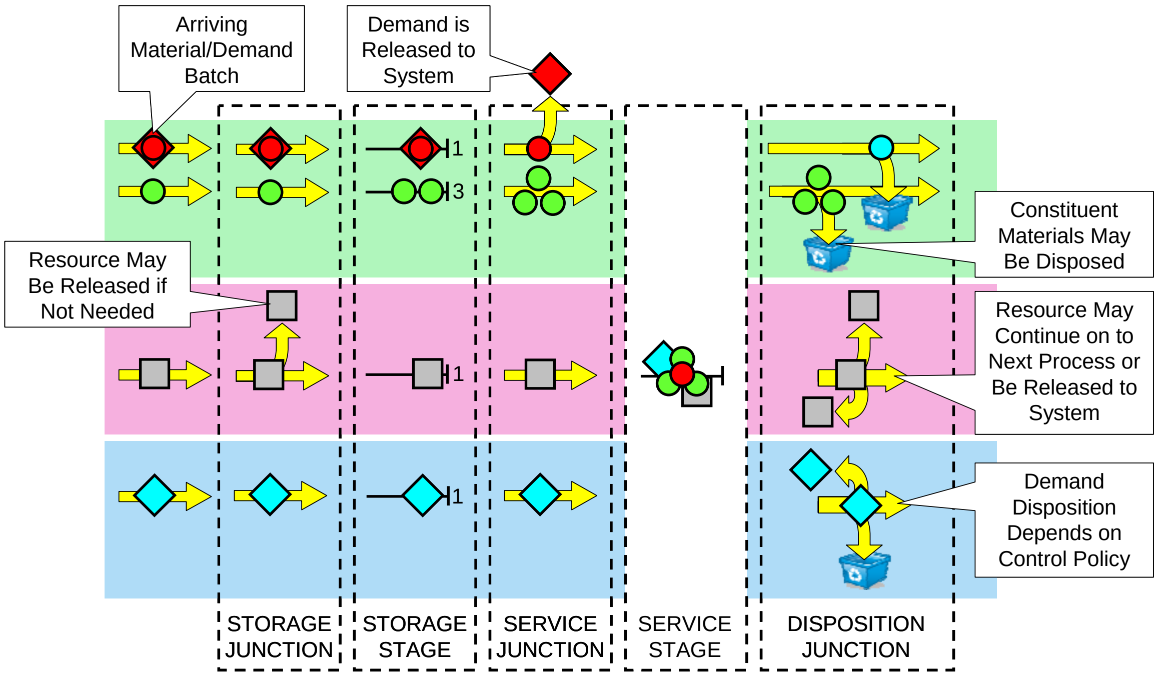

Arriving Material/Demand Batch
Demand is Released to System
Resource May Be Released if Not Needed
Constituent Materials May Be Disposed
Resource May Continue on to Next Process or Be Released to System
Demand Disposition Depends on Control Policy
1
3
1
1
STORAGE JUNCTION
STORAGE STAGE
SERVICE JUNCTION
SERVICE STAGE
DISPOSITION JUNCTION

Figure 29. A multi-flow modeling paradigm (MFMP) model of an assembly process. In this figure, the green bar contains the material component paths, the pink bar contains the resource component paths and the blue bar contains the demand component paths through the junctions and stages of the MFMP.

In the figure, the material, resource and demand component flow paths are denoted by the green, red and blue colored bars, respectively. The flows pass through the storage and service stages, just as the single server queueing model did, but these stages are bounded and separated by three junctions where the flows may be diverted out of the main process flow.

In the first junction, the storage junction, resource components may be diverted back to the manufacturing system. Resources so diverted might represent a forklift, a delivery person or some other system resource that is needed for transport, but is not required for this process.

In the second junction, the service junction, batched material and demand is separated and the demand is released back to the system. In this figure we see, for the first time, this batched material/demand component. It is common in manufacturing systems where pull production control is used to affix a card to some material. The card is later detached when the material is consumed and becomes a signal to produce more of that material. This "batched" component is analogous to such a real-world implementation of pull.

At the final junction, the disposition junction, each of the three flows may diverge. Material components may either continue on to the next process, or they may be disposed from the system. Here the constituent components of an assembly are disposed to leave only one "assembly" component remaining. Resource components may return to the storage stage if they are resources that are used solely in this process. They may also be released back to the system if they are resources that travel with the material through two or more processes. If this process is the first of a series of processes in which such a traveling resource originates, it may continue on to the next process with the resulting material component. In the disposition junction, we see again that the path of the demand component is a function of the production control policy in place.

Contrast the MFMP model of a manufacturing process with the single-server model. Like the MFMP model, the server has two stages: storage and service. However, it accommodates only one flow, that of material. In the server model, demand is implicit and resources are modeled as a separate binary data type, sometimes called a semaphore. The MFMP model can emulate a single-server model, but has greater flexibility. In the MFMP model, the state of the system takes the form of a single data type – the number of components in the storage stage. This greatly reduces the complexity of the control logic required to manage the system and adds flexibility to manage many types of manufacturing processes within the same basic framework. The result is a robust foundation upon which to build a complete simulation modeling framework.

5.3 Production Control Framework

The production control framework described in this chapter 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, in accordance with the MFMP. The proposed framework is shown in Figure 30.

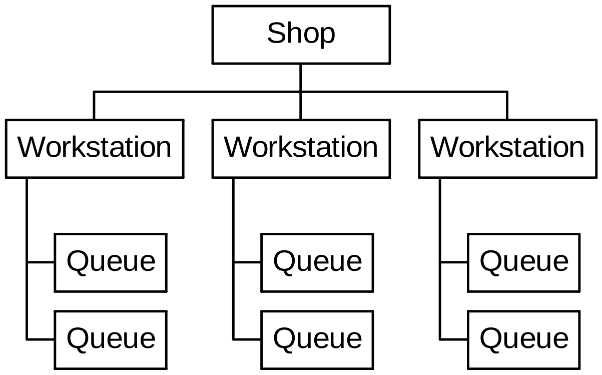


Figure 30. Production control framework

The lowest level in the framework is the queue. Queues store and order the components to be processed in the storage stage. 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 assigned to 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. A single workstation can control many processes from its queues. The workstation is essentially an MFMP model, executing all of the junction traffic control and housing the service stage, but with the storage stage queues modeled separately.

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 components throughout the system.

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.

5.3.1 Components

Traditionally, the term component referred only to the material elements of a product, but in the proposed framework material, resources and even demand are considered components, just as in the MFMP. I 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, Demand Components**

  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 Permission Components**

  Resource permissions, like demands, are signals transmitted through the system, but unlike demands, 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, machines, tools and any other capacity-limited system resource.

- **Type 4, Batch Components**

  Batch components are administrative groupings of types 1, 2 and 3 components that are to be processed as a single unit and travel together between the different elements of the PCF.

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 here 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:

$$\mathbf{c} = [c_1, c_2, \ldots, c_{nc}]^T \tag{51.}$$

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 21, below.

Table 21. PCF component types and numbers 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 22, below.

Table 22. PCF 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 remainders 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. Table 23, Table 24, Table 25 and Table 26 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 23. PCF type 1 (material) component 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 23 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 24. PCF type 2 (demand) component attributes

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

In Table 24, the only temporal attribute is due date. It is primarily used to measure the response time of a system. Each demand 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 25, 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 25. PCF type 3 (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 26. PCF type 4 (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, which is the type 1 component with the lowest component class attribute. Attribute 11 is taken from the primary demand 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.

5.3.2 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 two parameters:

$$\mathbf{q} = [q_1, q_2] \qquad (52.)$$

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 wide variety of operating policies 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 27 lists the parametric descriptions of the eight queue disciplines and their common names.

Table 27. Material queue discipline parameters

Queue Discipline	q_1	q_2
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 28 lists the parametric descriptions of five resource sequence rules and their common names.

Table 28. Resource sequence rule parameters

Dispatching Rule	q_1	q_2
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 here. 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.

5.3.3 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 workstation controller **w** has four components:

$$\mathbf{w} = [\mathbf{C}, \mathbf{Q}, \mathbf{X}, \mathbf{D}] \tag{53.}$$

C is a set of *nq* queue controller constituents. **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.

$$\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{nq}]^T \tag{54.}$$

$$\mathbf{c}_i = [\mathbf{c}_{i1}, \mathbf{c}_{i2}] \tag{55.}$$

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

$$\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{nq}]^T \tag{56.}$$

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

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{nx}] \tag{57.}$$

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{i,nq}, t_{is}, t_{ip}]^T \tag{58.}$$

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.

$$\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_{nx}] \tag{59.}$$

$$\mathbf{d}_i = [d_{i1}, d_{i2}, \ldots, d_{i,nq}]^T \tag{60.}$$

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 disposed. 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 disposed, but if they are not, they must be included in the output batch. Material components are functionally disposed 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 components from one control level to another. All communication is either up or down the hierarchy. A component must pass through the shop controller on its way from one workstation 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.

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 are 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.

5.3.4 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 **s** has four components:

$$\mathbf{s} = [\mathbf{W}, \mathbf{R}, \mathbf{P}, \mathbf{B}] \qquad (61.)$$

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 *nc* process plans.

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{nw}] \qquad (62.)$$

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

$$\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{nr}]^T \qquad (63.)$$

$$\mathbf{r}_i = \left[r_{i1}, r_{i2}, r_{i3}, r_{i4}\right] \tag{64.}$$

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.

$$\mathbf{P} = \left[p_1, p_2, \ldots, p_{np}\right]^T \tag{65.}$$

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 order interface, or control point, for the product.

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 order interface, 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.

$$\mathbf{B} = \left[b_1, b_2, \ldots, b_{np}\right]^T \tag{66.}$$

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. Thereafter 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 order interface workstation for product *i*. In some systems and under some production control policies, there may be more than one order interface workstation. If this were the case, bill of material information would be required to determine the number and destinations of duplicate demand 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.

5.4 Application

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 31 illustrates the manufacturing system.

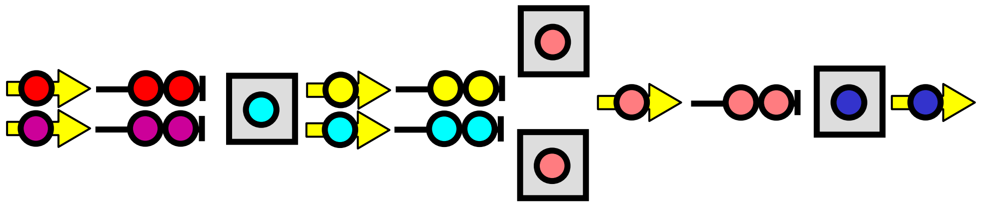


Figure 31. A three stage flow shop example. In this figure, two types of raw material components (red and purple circles) arrive at the first stage of the system, where they a single server processes them into sub-assemblies (yellow and blue circles, respectively). In the second stage of the system, the sub-assemblies are combined by one of two servers into an intermediate assembly (pink circle), which is then processed by a single server in the third stage of the system into a finished assembly (blue circle), which departs the system.

5.4.1 System Definition

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

$$\mathbf{s} = \left[\mathbf{W} \mid \mathbf{R} \mid \mathbf{P} \mid \mathbf{B}\right] \tag{67.}$$

The shop controller for this case is defined as

$$\mathbf{s} = \left[\begin{array}{c|cccc|c|c} \mathbf{w}_1 & 2 & 3 & 1 & 1 & 0 & 1 \\ \mathbf{w}_2 & 2 & 4 & 1 & 1 & 0 & 1 \\ \mathbf{w}_3 & 2 & 5 & 2 & 2 & 0 & 2 \\ & 2 & 6 & 3 & 1 & 0 & 2 \\ & 3 & 1 & 1 & 1 & 0 & 3 \\ & 3 & 2 & 2 & 2 & 0 & 4 \\ & 3 & 3 & 3 & 1 & & \end{array}\right] \tag{68.}$$

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 demand (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 three 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 are 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 count and destroy these components. 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.

$$\mathbf{w} = \left[\mathbf{C} \mid \mathbf{Q} \mid \mathbf{X} \mid \mathbf{D}\right] \tag{69.}$$

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

$$\mathbf{w}_1 = \left[\begin{array}{cc|c|cc|cc} 1 & 1 & \mathbf{q}_{11} & 1 & 0 & 0 & 0 \\ 1 & 2 & \mathbf{q}_{12} & 0 & 1 & 0 & 0 \\ 2 & 3 & \mathbf{q}_{13} & 1 & 0 & 0 & 0 \\ 2 & 4 & \mathbf{q}_{14} & 0 & 1 & 0 & 0 \\ 3 & 1 & \mathbf{q}_{15} & 1 & 1 & 1 & 1 \\ & & & 5 & 5 & & \\ & & & 5 & 5 & & \end{array}\right] \tag{70.}$$

For workstation 1, there are five queues; one for each of the two material classes processed there, one for each of the demands for the processes and one for the machines in the workstation.

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 3 subassembly. The second column defines the requirements to produce a class 4 subassembly.

Since there are two process combinations in the workstation, 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

$$\mathbf{w}_2 = \left[\begin{array}{cc|c|c|c} 1 & 3 & \mathbf{q}_{21} & 1 & 0 \\ 1 & 4 & \mathbf{q}_{22} & 1 & 2 \\ 2 & 5 & \mathbf{q}_{23} & 1 & 0 \\ 3 & 2 & \mathbf{q}_{24} & 1 & 1 \\ & & & 5 & \\ & & & 5 & \end{array}\right] \tag{71.}$$

$$\mathbf{w}_3 = \left[\begin{array}{cc|c|c|c} 1 & 5 & \mathbf{q}_{31} & 1 & 0 \\ 2 & 6 & \mathbf{q}_{32} & 1 & 0 \\ 3 & 3 & \mathbf{q}_{33} & 1 & 1 \\ & & & 5 & \\ & & & 5 & \end{array}\right] \tag{72.}$$

The function and behavior of queues is well understood. In the interest of brevity, assume all of the queue controllers are defined as FCFS or

$$\mathbf{q}_{ij} = [5 \quad 0] \ \forall \ i,j \tag{73.}$$

5.4.2 System Modification

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

$$\mathbf{P} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{74.}$$

Material component class 1 is now the order interface. Although the shop is still driven by push production control, the control point is now within the model and more progressive changes can begin to be made. For instance, the push pull interface may be moved downstream one station at a time until it is located at the last workstation and the shop is completely controlled by pull production control. The sequence of changes to **P** needed to implement this transition plan might be:

$$\mathbf{P} = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{75.}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 2 & 0 & 0 & 0 \end{bmatrix}^T \tag{76.}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 2 & 0 & 0 \end{bmatrix}^T \tag{77.}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 0 \end{bmatrix}^T \tag{78.}$$

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 2 \end{bmatrix}^T \tag{79.}$$

In the last configuration, the shop has been transformed to pull production control. This simple flow shop model illustrates the effectiveness of the proposed production control framework to model production control policy parametrically.

5.5 Summary

This chapter 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. The next step is to realize this framework in software and to test its ability to model production control.

Chapter 6 Simulation-based Optimization of Production Control

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 chapter introduces a technique that optimizes production control of single product flow shops under hybrid production control by using the Production Control Framework (PCF). This simulation modeling template is designed specifically to explore the production control domain. The chapter 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 kanban cards at each workstation. The objectives include improving customer service and reducing work-in-process inventory.

6.1 Hybrid Production Control Domain

In this chapter, we explore the hybrid production control domain, a sub-set of the full PCF domain. Using PCF nomenclature, we 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

$$\mathbf{P} = \left[p_1, p_2, \ldots, p_x, \ldots, p_{np}\right]^T \tag{80}$$

I describe the hybrid production control domain as

$$p_1 = 1 \tag{81}$$

$$p_x = 2 \tag{82}$$

$$p_{np} \in \{1,2\} \tag{83}$$

$$p_i = 1 \text{ for } 1 < i < x - 1 \tag{84}$$

$$p_i = 0 \text{ for } x + 1 < i < np \tag{85}$$

Thus, for a system with *np* material classes, there are *np-1* variations of hybrid production control. As Hopp and Spearman (2000) 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, $x = np - 1$.

6.2 Problem Setting

This chapter studies 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 32.

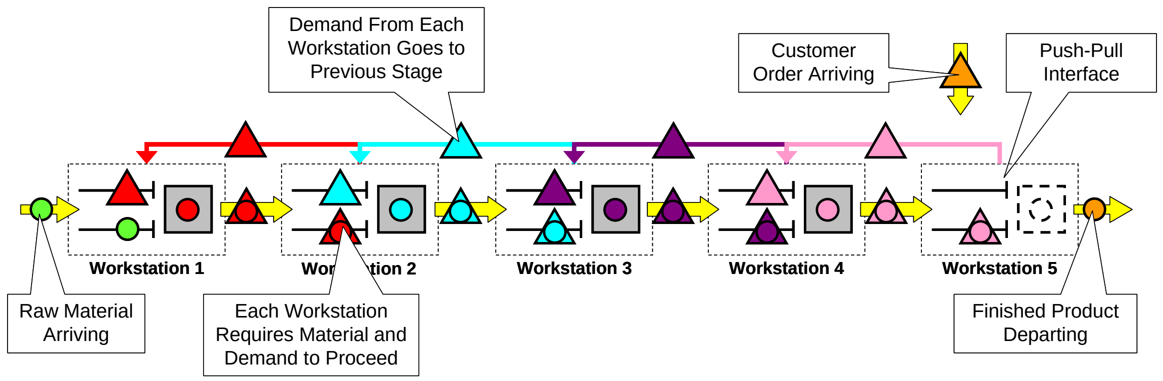

Demand From Each Workstation Goes to Previous Stage
Customer Order Arriving
Push-Pull Interface
Workstation 1
Wor tion 2
Workstation 3
Workstation 4
Workstation 5
Raw Material Arriving
Each Workstation Requires Material and Demand to Proceed
Finished Product Departing

Figure 32. A four stage flow line, shown configured for pull production control. In this example, incoming customer orders (orange triangles) are matched with finished goods in inventory (pink circle with triangle), which causes a finished goods kanban card (pink triangle) to travel back to stage 4 and signal the production of a finished good from an intermediate assembly in assembly (purple circle with purple triangle). This causes similar kanban card transactions triggering production of other intermediate assemblies (blue and red circles with triangles), from other assemblies (red circles with triangles) and raw materials (green circles), respectively. In this system, the push-pull interface is in workstation 5.

Using the PCF, five workstations are required to model a four stage system. One workstation is required to model each stage and its upstream buffers. A fifth workstation is needed to provide a downstream buffer for the fourth stage. The process in this fifth workstation is defined as requiring zero time. Using the PCF, the example system is modeled as

$$s_4 = \left[\begin{array}{c|cccc|c|c} w_1 & 2 & 2 & 1 & 5 & 0 & 1 \\ w_2 & 2 & 3 & 2 & 5 & 0 & 2 \\ w_3 & 2 & 4 & 3 & 5 & 0 & 3 \\ w_4 & 2 & 5 & 4 & 5 & 0 & 4 \\ w_5 & 2 & 6 & 5 & 5 & 2 & 5 \\ & 3 & 1 & 1 & 1 & & \\ & 3 & 2 & 2 & 1 & & \\ & 3 & 3 & 3 & 1 & & \\ & 3 & 4 & 4 & 1 & & \end{array}\right] \tag{86}$$

where

$$w_i = \left[\begin{array}{cc|c|c|c} 1 & i & q_{i1} & 1 & 0 \\ 2 & i+1 & q_{i2} & 1 & 0 \\ 3 & i & q_{i3} & 1 & 1 \\ & & & 0 & \\ & & & Logn(1.0,0.1) & \end{array}\right] \tag{87}$$

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

$$w_5 = \left[\begin{array}{cc|c|c|c} 1 & 5 & q_{51} & 1 & 0 \\ 2 & 6 & q_{52} & 1 & 0 \\ & & & 0 & \\ & & & 0 & \end{array}\right] \tag{88}$$

and

$$\mathbf{q}_{ij} = [5 \quad 0] \ \forall\ i,j \tag{89}$$

As a default, the WIP of each product in the system is set to five. Of course, this applies only to components controlled with pull. 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. For this configuration, the production control vector **P** is

$$\mathbf{P} = [0 \quad 0 \quad 0 \quad 0 \quad 2]^T \tag{90}$$

Since this system has five workstations, it can be used to model five different hybrid production control policies. Figure 33 illustrates the system with the order interface at the fourth stage, where **P** is

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & 2 & 1 \end{bmatrix}^T \tag{91}$$

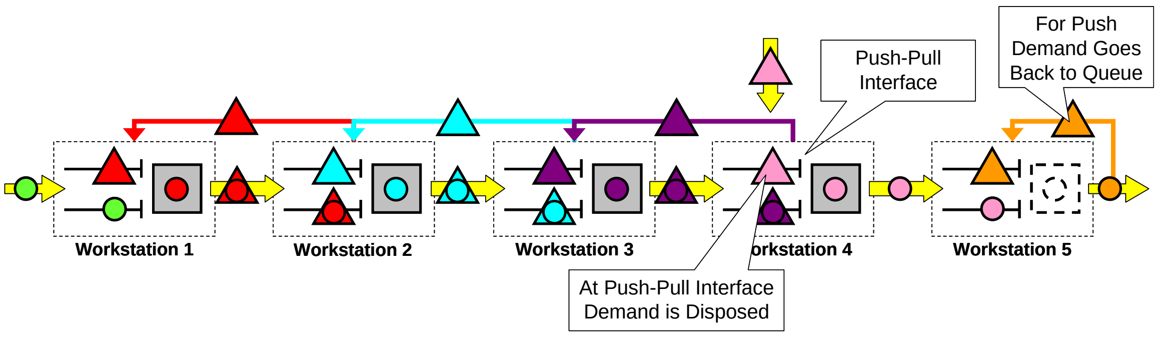


Figure 33. A four stage flow line with push-pull interface at fourth stage

Figure 34 illustrates the system with the push-pull interface at the third stage, where **P** is

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 2 & 1 & 1 \end{bmatrix}^T \tag{92}$$

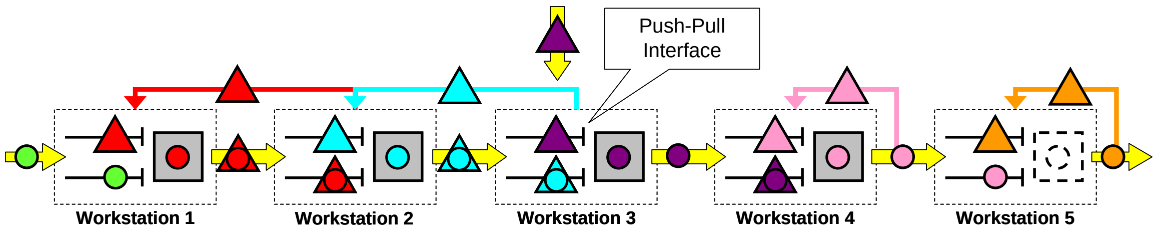


Figure 34. A four stage flow line with push-pull interface at third stage

Figure 35 illustrates the system with the order interface at the second stage, where **P** is

$$\mathbf{P} = \begin{bmatrix} 0 & 2 & 1 & 1 & 1 \end{bmatrix}^T \tag{93}$$

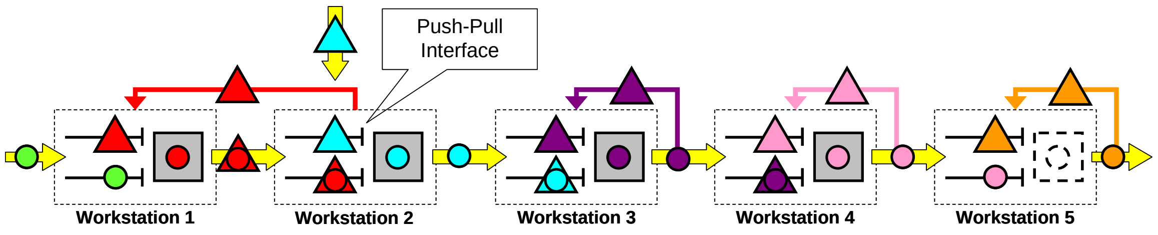


Figure 35. A four stage flow line with push-pull interface at second stage

Figure 36 illustrates the system in its final configuration, with the order interface at the first stage, in push production control. Here **P** is

$$\mathbf{P} = \begin{bmatrix} 2 & 1 & 1 & 1 & 1 \end{bmatrix}^T \tag{94}$$

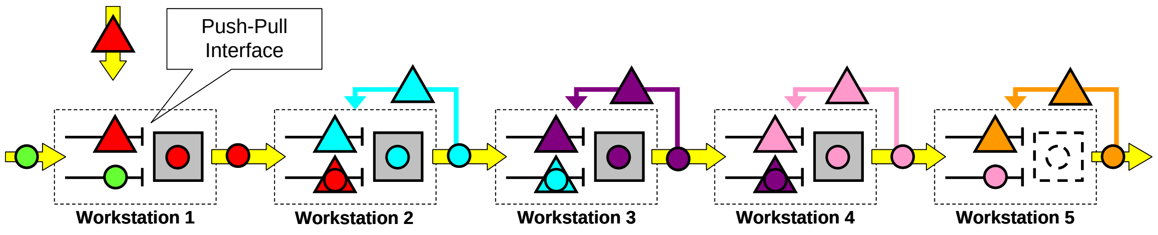


Figure 36. A four stage flow line with push-pull interface at first stage – push production control

Using the PCF, changing the production control policy is a simple parametric change to **P**. Models used in previous chapters, without the PCF, were much more difficult to change. Aside from the human effort benefits of using the PCF, changing production control from a structural model element to a parametric one makes it possible to use automated simulation-based optimization techniques to find the optimal production control policy for any PCF-modeled system, as we will soon demonstrate.

6.3 Model

I modeled the example system using a PCF-based template developed using Arena

(Kelton, Sadowski, and Sturrock, 2004). Figure 37 shows the user view of the example model in Arena.

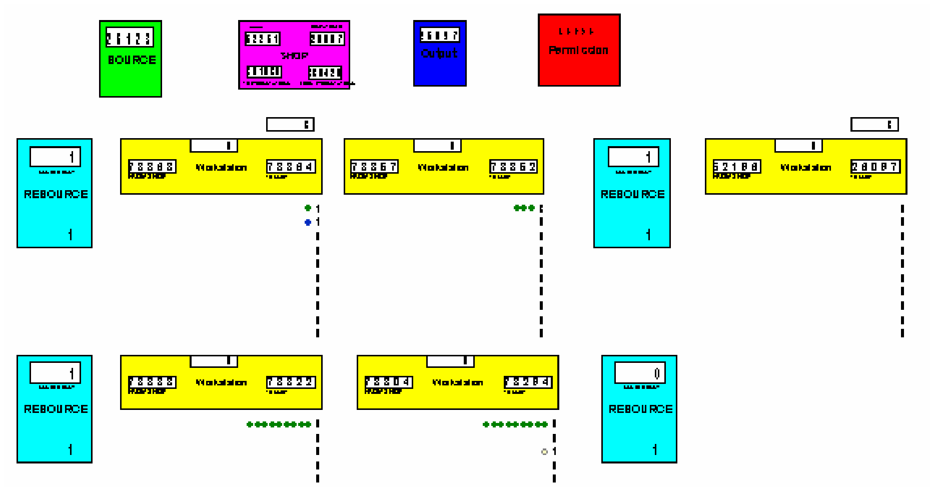


Figure 37. An Arena model of the example system, built using the PCF template

The PCF elements, as defined, are not sufficient to build a functioning model. Additional objects are necessary to create and dispose of the components that flow through the system. The PCF template provides modules for the creation of material, permission and resource components. It also provides a module to dispose of components that have completed processing.

By using the Arena platform, the built-in optimization engine, Optquest, can be applied to PCF models to find optimal production control policy and WIP levels.

6.4 Experiments

I chose to repeat a subset of the experiments Gaury (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 chapter is greatly reduced by my definition of the hybrid production control domain in which each station communicates only with the station immediately upstream.

6.5 Experimental Factors

Four experimental design factors were chosen to study the performance of the system. Gaury defined a set of 12 process, demand, and performance factors. My implementation of the PCF template is somewhat more limited in what it can express, so my experiments consider the following four factors (summarized in Table 29):

- **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

  $$\text{DI} = \max\{\text{TWC/N}-\min(\text{PTi}); \max(\text{PTi})-\text{TWC/N}\} \quad (95)$$

  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, the optimal WIP for each case was found in order to illustrate the difference this factor has on performance.

Table 29. Flow shop optimization experiment 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

6.6 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 30 details the experimental design.

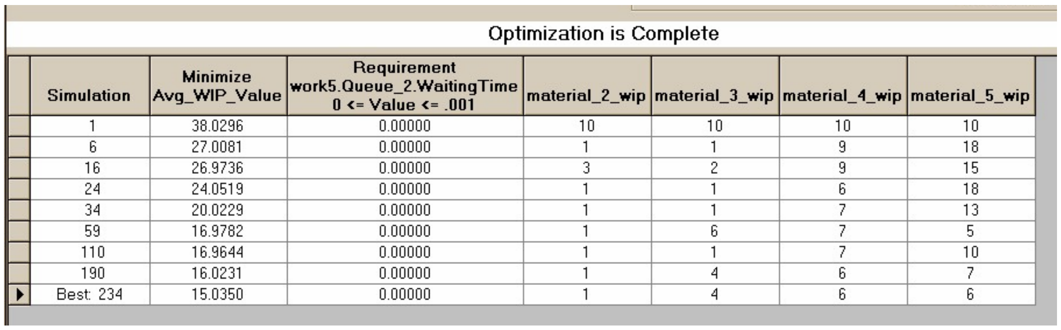
Each experiment was conducted with the customer order interface in each of the five possible configurations for a total of 60 experiments. Each trial was run for a single replication of 24,000 time units with a warm-up period of 1,000 units.

Table 30. Design of experiment, flow show optimization

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	-	-	-	-

6.7 Optimization Setup

Each trial configuration was optimized using Optquest, an optimization package that comes bundled 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 kanban cards that authorize production at that station. Otherwise, the permissions are not used. These controls were limited to integer values from 1 to 50, with 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 38 shows the Optquest user interface. The parameter values used in the experiments are included in Appendix A.


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 38. Optquest user interface

6.8 Results

The experimental results show, consistent with my 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 kanban cards 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 needed to further understand how changing the customer service requirements affect the performance of different production control policies.

6.9 Summary

This chapter explored the potential of the PCF to facilitate the use of simulation-based 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 re-configure 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 automated 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 in this direction would consider larger generic flow lines with non-monotonic processing time imbalances. It would pursue the incorporation of more intrinsic objective measurement. It would also address the limitations of the simulation optimization tool constraints.

Chapter 7 Conclusion

7.1 Summary

This dissertation addresses pull production control, an important component of Lean manufacturing, and explores the poorly understood process of implementing Lean principles and changing the production control policy. There is a host of Lean literature, no doubt due to its popularity as a buzzword in industry circles, but the rigor of most of this work is questionable. It is often based more on anecdotes than first principles, and, as a result, offers little to the would-be practitioner in terms of useful guidance. The serious analytical work in the production control realm of Lean study is focused on steady state performance of Lean systems. However, even the anecdotal articles tell the reader that the period of transition from “fat” to Lean is the most critical for ultimate success. This dissertation provides a greater understanding of this transient period and develops useful guidelines and tools for Lean production control design and transition with the real-world objective of reducing cost.

The transformation of a single stage of a manufacturing process is the fundamental element of production control transition. We developed a model of the transformation process, defining the events of a push-to-pull transition and developing a cost-based objective function. We based the objective function on the cost of holding both inventory and unfulfilled orders. We introduced three factors that we can use to control the speed of transition and thereby mitigate cost: controlling the speed at which kanban cards are introduced to the system, deferring the flow of new customer orders into the system and adding extra capacity to deal with the temporary surge in traffic. These factors added their own costs to the objective function. To test the effects of mitigation on cost, we developed both a simulation model of the system and an approximate analytical model. In order to find the right balance of mitigation techniques, we employed simulation based optimization. We demonstrated that the optimal transition policies we developed outperformed simple policies derived from the Lean literature and so-called common sense. We also established a useful guideline to be used in the absence of thorough analysis: accelerate transition as much as possible except where backlog is expensive.

Multi-stage transition is the reality of Lean production control implementation. Most manufacturing systems have many, often very diverse, stages. Using the single stage model as a basis, we developed a framework with which to describe the events of a multi-stage transition. In the multi-stage scenario, the events of each individual stages transformation are no different than the single stage model. The fundamental question of the multi-stage system then is when to initiate each individual transformation. Using a simulation model of the system, we studied two special cases of multi-stage transition: all of the stages transforming at the same time in the All At Once strategy and the stages transforming one at a time in sequence in the One By One strategy. It was shown that the One By One strategy was less costly by avoiding some of the more radical interactions of adjacent stages transforming together. Mitigation techniques that were applied to the single stage were tested on the multi-stage model, and they were shown to be just as effective in lowering the time and cost required for transition in this case.

In order to model and optimize the transition of complex systems like those that one would find in a real manufacturing system, a new modeling paradigm was required. The queue server model has long been used to represent push production control, but pull has no such convenient analog. By extending the queue server model and modifying it in a manufacturing domain specific manner, we developed the Multi Flow Model of a manufacturing stage as a basis on which to build more complex models. The Multi Flow Model treated all of the ingredients of a manufacturing process, which we defined as material, resources and demand, as components all necessary to trigger the stage to begin service. By instantiating resources, and demand we provided the signals necessary to coordinate pull production control. However, the signals required a network across which to travel. To provide that network, the Production Control Framework was defined. The Production Control Framework is a hierarchical model, based on the Multi Flow Model, which can be used to emulate a rich variety of manufacturing systems. Its three tiers, the Shop, the Workstation and the Queue can be pieced together and configured in different ways to express a wide variety of production control and dispatching rules. One novel aspect of the Production Control Framework is the ability to express the production control of the system in parametric form and thereby enable automated manipulation and optimization of a model.

The power and flexibility of the Production Control Framework were demonstrated when we implemented it in a set of software objects and used it to find the optimal production control policy for a five stage manufacturing system. The models created with the Production Control Framework software were very easy to reconfigure to emulate different production control policies. Its compatibility with simulation-based optimization was shown when we employed Optquest, a common optimization engine, to find the optimal production control policy for the system under different operating conditions.

7.2 Contribution

This dissertation makes contributions to the study of production control, especially in transient conditions, and to the simulation modeling of manufacturing systems.

Production control is normally studied in steady state. Lean manufacturing, a transformational doctrine, demands attention to the transient behavior of systems undergoing changes in production control. To do this, I created new analytical models of systems experiencing non-stationary traffic intensity. I created simulation models capable of studying transient behavior – phenomena usually carefully eliminated from simulation studies. I used these tools to explore the transient behavior of single stage and the surprisingly complex behavior of multi-stage systems undergoing production control transition. To control these effects, I developed the first well-defined transition strategies.

Simulation modeling of pull production control is a technical challenge. Traditional simulation modeling software is designed to model push production control. In order to simulate complex systems under a variety of production control policies, I developed a new modeling paradigm, built by extending the logical basis of contemporary simulation models. Using this new paradigm, I created a new, hierarchical framework built specifically to model a wide variety of manufacturing system production control policies and to express production control as a parameter of a model, rather than part of its structure. I implemented this framework in a new class of re-usable software objects that make it possible to use automated tools to find optimal production control policies.

Together, these contributions lay the groundwork for the exploration of the transient behavior of truly complex systems undergoing Lean transition and for optimization of those transitions.

7.3 Future Work

Although this dissertation sheds light on some previously unexplored phenomenon of Lean transformation, it leaves many questions yet unanswered and poses still more. It provides new tools to answer some of those questions, but they require more refinement to realize their full potential.

My models of the single stage transformation deserve continued attention. There remains some inaccuracy in the predictions of the analytical model versus the simulation model results. Continued study should focus on understanding that inaccuracy. One possible source of inaccuracy that should be examined is the selection of the simulation model run termination event. The time plots indicate that the system may take some time after the termination event to reach its new steady state. Meerkov and Zhang (2008) present an alternative termination criterion that should be pursued in this context. Further, our choice of first-in-first-out as the queue discipline used in the model may affect the results. One might argue that no real-world implementation of Lean would choose to give priority to orders destined for stock ahead of customer orders, no matter when they arrived at the stage. The effect of different queue disciplines should be evaluated.

Much more work is required on the subject of multi-stage transition. This dissertation barely scratched the surface of the fascinating behavior of such systems. Just as in the single stage case, the effect of varying the queue discipline should be investigated. Giving priority to customer orders over kanban cards might moderate the more extreme behavior the systems exhibited. Only two of the three mitigation techniques demonstrated with the single stage system were used to mitigate the multi-stage system. Transition strategies that make use of this technique should be developed and similarly tested. In general, more transition strategies should be developed and categorized. Some possible variations include different types of transition triggers, varying versus fixed mitigation levels, and shared versus non-shared resource pools. Systems with different line balances and varying performance should be examined to determine which strategies work in which kinds of systems. Automated optimization should be employed to optimize multi-stage transition. Production Control Framework based modeling objects now make this possible.

For all of the transition modeling and scenario testing, realistic cost rates should be further investigated. In this dissertation, I used unit rates in almost all of the tests. Although this does not affect my conclusions, it may render some of them moot from a practical standpoint.

The PCF software template is powerful and flexible, but it is not user-friendly. Describing a complex system using the PCF is not easy. The software often makes it more complex than necessary due to the limitations of the user interface. Developing a wizard or some other design aid would add great value to these tools. The software objects themselves would benefit from enhanced outputs, both visual and data. Further, there is no cost model like those used in Chapters 3 and 4 built into the software. This would be a useful addition.

The optimization of PCF models in Chapter 6 put the limitations of the optimization tool in stark relief. In Chapter 4 I developed simple constraints to define a multi-stage transition. However, the optimization tool I used did not allow me to input these constraints since they related one input variable to another. As a result, I was forced to do a full-factorial, fully manual evaluation. Either a new optimization tool must be identified that can accommodate the constraints I developed, or some kind of shell software must be devised to manage the constraints and run the experiments through the optimization software.

These few additions could add even more value to the work already completed in this dissertation. I look forward to pursuing them.

Appendices

APPENDIX A: EXPERIMENTAL INPUTS

Table 31 details the input parameters for the 12 configurations in which I optimized the production control policies for the example four-stage flow line.

Table 31. Experiment Input Parameters

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

Table 32 shows the optimal number of kanban cards at each of the four stages of the example flow line for each of the 12 configurations shown in Table 31.

Table 32. Optimal WIP Level for Experiment Input Parameters

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 kanban cards	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 kanban cards	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 kanban cards	4	4	18	6	15	17	16	4	18	27	13	11

Glossary

Acronyms

1B1:	One By One Transition Strategy
AA1:	All At Once Transition Strategy
CONWIP:	Constant Work In Process
DFF:	Deterministic Fluid Flow Model
EDD:	Earliest Due Date
FIFO:	First In First Out
MFMP:	Multi Flow Modeling Paradigm
MRP:	Material Requirements Planning
PCF:	Production Control Framework
PPI:	Push-Pull Interface
SPT:	Shortest Processing Time
SSS:	Steady State Stochastic Model
WIP:	Work-in-Process Inventory

Terminology

All At Once Transition Strategy (AA1): A strategy for converting a *multi-stage* manufacturing system from *push* to *pull* production control in which *kanban* cards are introduced into all stages simultaneously and the *push pull interface* moves from the start of the system to the end when the stage 1 *inventory* contains all of its *kanban* cards.

Backlog: A queue of unfilled customer orders and/or *kanban* cards awaiting material and/or service.

Constant Work In Process (CONWIP): A production control technique in which the amount of *WIP* in the system is held constant by matching order releases into the system with departing orders.

Deterministic Fluid Flow Model (DFF): A model of a queuing system in which the service time is deterministic and variability is introduced in the form of a non-stationary, but also deterministic, customer arrival rate.

Earliest Due Date (EDD): A queue discipline in which orders are arranged in ascending due date order. This queue discipline achieves low average order lateness at the expense of throughput.

First In First Out (FIFO): A queue discipline in which orders are arranged in ascending order of queue entry time.

Hybrid Production Control: A production control policy governing a multi-stage manufacturing system in which some of the stages are controlled with *push* and others with *pull*.

Inventory/Order Interface: See *Push-Pull Interface*.

Kanban: Japanese for "shop sign", and first adopted in the Toyota Production System, this term refers to paper or plastic cards that manufacturing systems under *pull* control often use to instantiate demand signals. In practice, *kanban* cards are affixed to products or components in *inventory*. When that product or component is needed, the card is detached and returned to its original process, which now has explicit permission to refill the *inventory* of the part specified on the card.

Lean: The western name for the Toyota Production System, a manufacturing management philosophy focused on the reduction of waste.

Material Requirements Planning (MRP): A production control system that attempts to fill external demand (customer orders) by scheduling the creation of purchase and production orders. For a given end item, MRP contains a bill of material, which it explodes to determine gross requirements for components and sub-assemblies. Using fixed lead times and assuming infinite capacity, it calculates creation dates for each order by back-scheduling from the due date of the external demand order (for the finished goods) or from the calculated creation date of the next higher assembly order (for subassemblies and components).

Multi-Flow Modeling Paradigm (MFMP): A domain-specific abstraction of a manufacturing process in which the flow of materials, resources and demand are considered, enabling emulation of a wide variety of production control policies, including both *push* and *pull*.

Multi-stage: A queuing system in which customers wait in a prescribed series of two or more queues (stages) for service by one or more servers. The variability of customer arrivals and service times at each stage dictates the arrival rate at the next.

One By One Transition Strategy (1B1): A strategy for converting a *multi-stage* manufacturing system from *push* to *pull* production control in which *kanban* cards are introduced into one stage at a time, beginning at the upstream end. As a given stage begins the transition, it becomes the *push-pull interface*. When all of the *kanban* cards have been processed and are waiting in *inventory*, the transition of the next stage downstream begins.

Production Control Framework (PCF): A three-tiered hierarchical description of a manufacturing system based on the *multi-flow modeling paradigm*. Starting from the bottom, the three levels of the *PCF* are: queue, workstation and shop.

Production Control Policy: The operating policy of a workstation in a manufacturing system that specifies whether the presence of material implies permission to process or if additional explicit permission is required. See *pull* and *push*.

Pull: A *production control policy* in which a workstation requires both material and explicit permission to proceed before processing. Permission to proceed is often granted in the form of a *kanban* card. Permission is generated by a customer, either external of internal to the system when they consume the product of a given workstation. Thus, production is governed by consumption and the number of *kanban* cards in a system dictates the level of *work-in-process*.

Push: A *production control policy* in which a workstation may process material as long as it is available. Permission to proceed is implicit with the presence of material.

Push-Pull Interface (PPI): The stage in a *multi-stage* manufacturing system under *hybrid production control* at which customer orders are introduced. All of the stages upstream of the *PPI* are controlled with *pull* and all of the stages downstream of the *PPI* are controlled with *push*. Also called *inventory/order interface*.

Shortest Processing Time (SPT): A queue discipline in which orders are arranged in order of increasing processing time. This queue discipline achieves high throughput at the expense of average due date performance.

Single-stage: A queuing system in which customers wait in a single queue to be served by a single server. When service is complete, they exit the system.

Steady State Stochastic Model (SSS): A model of a queuing system in which both the interarrival and service times are stochastic, but stationary.

Work-in-Process Inventory (WIP): A queue of non-finished-goods material awaiting orders/kanban cards and/or service.

Bibliography

Alrefaei, M.H. and S. Andradottir. 1995. “A new search algorithm for discrete stochastic optimization”. Proceedings of the 1995 Winter Simulation Conference, ed.C. Alexopoulos, K. Kang,W.R. Lilegdon, and D.Goldman, pp.236-241.

Alrefaei, M.H. and S.Andradottir. 1999. “Temperature for discrete stochastic optimization”. Management Science, pp.748.

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

Andradottir, S. 2002. “A method for discrete stochastic optimization”. Management Science, Volume 41, pp.1946-1961.

Andradottir, S. 1996. “A global search method for discrete stochastic optimization”. SIAM Journal on Optimization, Volume 6, pp. 513-530.

Askin, R. G. and J. B. Goldberg. 2002. *Design and Analysis of Lean Production Systems*. John Wiley & Sons, New York.

Banks, J. 1995. “Semantics of simulation software”. OR/MS Today, December.

Banks, J. Ed. 1998. *Handbook of Simulation*. Wiley Interscience, New York.

Banks, J. and R. Gibson. 1998. “Simulation evolution”. IIE Solutions, November, pp. 26-29.

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

Bispo, C. F., and S. Tayur. 2001. “Managing simple re-entrant flow lines: theoretical foundation and experimental results”. IIE Transactions, Volume 33, pp. 609-623.

Bonvik, A. M., C. E. Couch and S. B. Gershwin. 1997. “A comparison of production-line control mechanisms”. International Journal of Production Research, 35(3), pp. 789-804.

Buzacott, J. A., and G. J. Shanthikumar. 1993. *Stochastic Models of Manufacturing Systems*. Prentice Hall, Englewood Cliffs, New Jersey.

Carrascosa, M., S. D. Eppinger, and D. E. Whitney. 1998. "Using the design structure matrix to estimate product development time," Paper DETC98/DAC-6013, Proceedings of DETC98, 1998 ASME Design Engineering Technical Conference, Atlanta, Georgia, September 13-16.

Cassandras, C. G., D. Liyi and P. Christos. 1998. "Ordinal Optimization for a Class of Deterministic and Stochastic Discrete Resource Allocation Problems," IEEE Transactions on Automatic Control, Volume 43, Number 7, pp. 881-900.

Chase, C., and Ramadge, P. J. 1992. "On real-time scheduling policies for flexible manufacturing systems". IEEE Transactions on Automatic Control, Volume 37, Number 4, pp. 491-496.

Chu, C.-H., and W.-L. Shih. 1992. "Simulation studies in JIT". International Journal of Production Research, Volume 30, Number 11, pp. 2573-2586.

Church, L. K., and Uzsoy, R. 1992. "Analysis of periodic and event-driven rescheduling policies in dynamic shops". International Journal of Computer Integrated Manufacturing, Volume 5, pp. 153-163.

Fang, J., and Xi, Y. 1997. "A rolling horizon job shop rescheduling strategy in the dynamic environment". The International Journal of Advanced Manufacturing Technology, Volume 13, pp. 227-232.

Fu, M. C. 1994. "Optimization via simulation : a review". Annals of Operations Research, Volume 53, pp.199-247.

Gahagan, S. M., and J. W. Herrmann. 2001. "Improving simulation model adaptability with a production control framework". In Proceedings of the 2001 Winter Simulation Conference, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer. pp. 937-945. Piscataway, New Jersey: Institute for Electrical and Electronics Engineers. Available via <http://www.informs-sim.org/wsc01papers/124.PDF> [accessed August 15, 2005].

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.

Gerencser L., S.D.Hill and Z.Vago. 1999. "Optimization over discrete sets via SPSA," Proceedings of the 1999 Winter Simulation Conference, pp.466-470.

Gershwin, S. B. 1994. *Manufacturing systems engineering*. Prentice Hall, Englewood Cliffs, New Jersey.

Green, G.I., and L.B. Appel. 1981. “An empirical analysis of job shop dispatch rule selection”. Journal of Operations Management, Volume 1, pp. 197-203.

Hall, R. 1991. *Queueing Methods for Services and Manufacturing*. Prentice Hall, New Jersey.

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

Hopp, W. J. and M. L. Spearman. 2000. *Factory physics, 2nd edition*. Irwin/McGraw-Hill, Boston, Massachusetts.

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

Kelton, W. D., R. P. Sadowski, and D. T. Sturrock. 2004. *Simulation with Arena, 3rd ed*. McGraw-Hill, Boston, Massachusetts.

Kiefer, J. and J.Wolfowitz. 1952. “Stochastic estimation of the maximum of a regression function”. Annals of Mathematical Statistics, Volume 23, pp. 462-466.

Kumar, P.R. 1994. “Scheduling manufacturing systems of re-entrant lines”, *Stochastic Modeling and Analysis of Manufacturing Systems*. Yao, D. D. ed., pp. 325-360, Springer-Verlag, New York.

Law, A. M. and W. D. Kelton. 2000. *Simulation Modeling and Analysis, 3rd edition*. McGraw-Hill, New York.

Li, R-K., Y.-T. Shyu and S. Adiga. 1993. “A heuristic rescheduling algorithm for computer-based production scheduling systems”. International Journal of Production Research, Volume 31, pp. 1815-1826.

Liker, J. 2004. *The Toyota Way*. McGraw-Hill, New York.

Liker, J.K., editor. 1997. *Becoming Lean: Inside Stories of U.S. Manufacturers*, Productivity Press, Portland, Oregon.

Markowitz, D. M. and L. M. Wein. 2001. “Heavy traffic analysis of dynamic cyclic policies: a unified treatment of the single machine scheduling problem,” Operations Research, Volume 49, Number 2, pp. 246-270.

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

Meerkov, S. M. and L. Zhang. 2008. “Transient behavior of serial production lines with Bernoulli machines”, IIE Transactions, Volume 40, pp. 297-312.

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

Olumolade, M. O. and D. H. Norrie. 1996. “Reactive scheduling system for cellular manufacturing with failure-prone machines”. International Journal of Computer Integrated Manufacturing, Volume 9, Number 2, pp. 131-144.

Panwalker, S. S., and W. Iskander. 1977. “A survey of scheduling rules”. Operations Research, Volume 25, Number 1, pp. 45-62.

Perkins, J. R. and P.R. Kumar. 1989. “Stable, distributed, real-time scheduling of flexible manufacturing/assembly/disassembly systems”. IEEE Transactions on Automatic Control, Volume 34, Number 2, pp. 139-148.

Pflug, G. 1996. *Optimization of Stochastic Models: The Interface between Simulation and Optimization*. Kluwer Academic Publishers, Norwell, MA.

Price, W., M. Gravel and L. Nsakanda. 1994. “A review of optimisation models of kanban-based production systems”. European Journal of Operational Research, 75, pp. 1-12.

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.

Sabuncuoglu, I. and S. Karabuk. 1999. “Rescheduling frequency in an FMS with uncertain processing times and unreliable machines”. Journal of Manufacturing Systems, Volume 18, Number 4, pp. 268-283.

Schonberger, R. J. 1996. *World class manufacturing: The next decade: Building power, strength and value*. The Free Press, New York.

Shi, L. and S. Olafsson. 2000. “Nested partitions method for global optimization,” Operations Research, Volume 48, Number 3, pp. 390-407.

Shingo, S. 1989. *A Study of the Toyota Production System from an Industrial Engineering Viewpoint*. Cambridge, Massachusetts: Productivity Press.

Slack, N. (Ed.). 1997. *The Blackwell Encyclopedia Dictionary of Operations Management*. Oxford, United Kingdom: Blackwell Publishers Ltd.

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

Spall, J.C. 1995. "Implementation of the simultaneous perturbation algorithm for stochastic optimization," IEEE Transactions on Aerospace and Electronic Systems, Vol. 34, No. 3, pp.817-823, 1998.Thomson, Norman. Simulation in manufacturing. Research Studies Press, Taunton, Somerset, UK.

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 ed. 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.

Yan D. and H.Mukai. 1992. "Stochastic discrete optimization," SIAM Journal on Control and Optimization, Vol. 30, pp.594-612.