A Framework for Evaluating Production Policies to Improve Customer Responsiveness

S. Kurnaz, A. Cohn, Y. Koren (1)

NSF Engineering Research Center for Reconfigurable Manufacturing Systems,

University of Michigan, Ann Arbor MI, USA

Abstract

Manufacturing systems are subject to both internal and external disruptions. As a result, production sequences which appear optimal during planning might be sub-optimal or even infeasible when implemented. We therefore consider how manufacturers can trade off anticipated completion time (makespan) against customer responsiveness when making sequencing decisions. We first define a policy as a way to control product sequencing in real time. We then present a framework for evaluating different policies to assess their impact on customer responsiveness. Finally, we demonstrate how this framework can be used to analyze the link between sequencing, lot size, and update frequency.

Keywords:

Manufacturing system, Operational flexibility, Responsiveness

1 INTRODUCTION

In this paper, we consider how disruptions (machine breakdowns, customer order changes, etc.) impact the quality of sequencing decisions in a *sequential manufacturing line* (e.g., *flow shops*) producing multiple products. In such a system, all product types move through the same sequence of operational stages, but may spend different amounts of time in each. If the products are not perfectly balanced in terms of these processing times, either across operational stages for a given product or between different products, then the order in which they are produced can have a considerable impact on primary performance metrics such as order completion time (*makespan*). Thus, making relatively simple, low cost operational changes (i.e. resequencing the products) can significantly alter system performance without any additional capital costs. In particular, sequencing decisions can impact *customer responsiveness* – loosely defined as how well we provide customers *what* they want *when* they want it – which is of particular importance in today's competitive and volatile marketplace.

In practice, material/manufacturing requirements planning tools (MRP/MRPII) are often used for production planning, including product sequencing. Such tools provide many important benefits, but are designed for deterministic environments with long lead times. When disruptions occur, changes must be manually entered into the system and then the user must develop a new product sequence. Therefore, these traditional tools do not fit production environments in which customers are allowed to make due date and quantity changes in their original orders; a new framework that considers such conditions is needed.

Although MRPII systems do not place primary emphasis on sequencing decisions, the academic community *has* devoted significant research to developing optimization tools for sequencing products so as to minimize makespan [e.g. 1, 2, 3]. However, this research again fails to address the issue of system disruptions. It is often the case that sequences which are optimal relative to the initial system inputs may not be optimal – or even feasible -- when implemented under actual conditions in which system disruptions such as machine failures and customer changes during order processing can occur (see, for example, [4, 5]).

In recent years, additional research has focused on addressing system variability in product sequencing [e.g. 6, 7]. Inherent in research efforts such as these, however, is the need to establish metrics for evaluating solution quality. Metrics typically considered include average makespan, holding and shortage costs, and excess inventory. But in today's competitive global economy, it is no longer sufficient to focus solely on these manufacturer-focused metrics -- customer responsiveness must be considered as well [8].

In environments where capacity is tightly constrained relative to customer demand, and where customer orders are typically filled through new production rather than from inventory, disruptions can often prevent manufacturers from fully satisfying customer demand at the original due date. In such cases, it is not immediately clear what the relationship is between a given production sequence and how customer responsive it is. Given that a customer's demand cannot be fully satisfied at the original due date, particularly if the order quantity has changed, what feasible alternatives are most desirable? How can this less easily quantifiable notion of customer responsiveness be incorporated in the decision making process?

A number of researchers have considered customer responsiveness in their work, either explicitly or implicitly. Matson and McFarlane define responsiveness as **'**…the ability of a production system to respond to disturbances …which impact upon production goals' [9]. Shafaei and Brunn define customer responsiveness simply as whether or not the customer demands are *fully satisfied* by the due date [10]. Wiendahl et al take a broader view of customer responsiveness across the whole supply chain [11].

As we see with these three simple examples, one unmet challenge in addressing customer responsiveness is that there is no universally valid measure of responsiveness – different industries and applications value different characteristics in their production outcomes. For example, in the semiconductor industry, suppliers are evaluated based not only on what percentage of the total demand can be made available at the due date (a common requirement is at least 95%), but also on how quickly the remaining demand can be provided. In some other situations, any demand not filled at the original due date will be *lost* and therefore this is the only relevant criterion.

The focus of our research is on using sequencing decisions not only to decrease makespan, but also to improve customer responsiveness. We seek to provide tools that support the development of automated *policies* which not only provide an initial product sequence but also advise the user on how to update this sequence as disruptions occur. In particular, we do not want to dictate how the system performance should be evaluated, but rather to allow users flexibility in assessing solution quality according to their individual situations. The goal of our research is thus not to develop optimal sequencing strategies for a single industrial situation but rather to develop a *framework* – both a vocabulary and a set of analytical tools – for analyzing a broad class of problems.

2. DEFINITION OF A POLICY

The existing state-of-the-art in production sequencing focuses either on makespan minimization or on resequencing when disruptions occur. We introduce the notion of a *policy* as a way to incorporate both of these goals and, more importantly, to provide manufacturers with planning tools that enable them to better achieve customer responsiveness, independently of how customer responsiveness is defined.

We define a policy as a set of rules for controlling the sequencing of products during production. Given a set of input data, a policy defines what the initial sequence should be and how this sequence should be modified as disruptions occur. A policy is made up of three components: *sequencing rules*, *update triggers*, and *strategies for anticipating demand changes*.

Sequencing rules: A sequencing rule takes as input a set of products and their corresponding demands and outputs a production sequence. For example, in the *fixed lot size* sequencing rule, an ordering of product types is given and a lot size is specified. Products are sequenced according to this ordering in batches of the specified lot size. For example, given a demand for 10 A's, 8 B's, and 4 C's, with a product ordering of {A, C, B} and a lot size of 3, the resulting sequence would be:

A A A C C C B B B A A A C B B B A A A B B A

Another possible sequencing rule is the *optimal makespan rule*, in which products are sequenced such that, in the absence of any disruptions, the makespan will be minimized. Note that there is a virtually infinite set of possible sequencing rules. We do not constrain the rules, so long as they provide a well defined mapping from a set of products to a production sequence.

Update triggers: Given the occurrence of disruptions, a sequence may no longer be optimal or even feasible. It may therefore be appropriate to update and resequence the remaining demand. It is not necessarily effective, however, to update the sequence every time there is a disruption. Update triggers specify when to update the sequence. *Event-driven update intervals* are triggered by the occurrence of an event which alters a parameter of production. Examples include demand changes of at least a specified percentage, machine breakdowns, or order cancellations. *Periodic update intervals* are triggered by a fixed time period, such as every 20 minutes or every 4 hours. *Hybrid update intervals* are triggered both by event and time. For example, resequencing may occur at fixed time intervals but also when significant events occur. Note that when an update is triggered, a new sequencing rule may be applied – it is not required that the same sequencing rule be applied at all updates.

Strategies for anticipating demand changes: Customers often increase, decrease, or cancel their orders after production has begun. Manufacturers can not only respond to such changes *reactively* by resequencing, but may also act *proactively*, using historical data to anticipate such changes. For example, a manufacturer may initially sequence production for only 80% of the initial demand for product B if it is predicted that orders for this product will decrease during production.

A policy is simply a combination of sequencing rules, update triggers, and strategies for anticipating demand changes. An example policy begins by sequencing 80% of the demand of customer A and 100% of all other customers' demand using a fixed lot sequencing rule. Whenever demand for any individual product changes by more than 20%, the sequence is updated, again using a fixed lot policy. Halfway through the production horizon, the sequence is again updated, now planning for the complete (updated) demand of customer A. Finally, at 75% of the time horizon, the sequence is updated according to the optimal makespan rule. Note that a policy clearly defines the production sequence, even in the presence of disruptions. Furthermore, a virtually infinite array of sequencing rules, update triggers, and strategies for anticipating demand changes makes the concept of a policy extremely general.

3. A FRAMEWORK FOR POLICY EVALUATION

It is unrealistic to attempt to develop a tool for generating optimal policies that apply to fully generic systems in which all disruptions are possible. The wide variety of system configurations and disruption distributions, along with the challenges associated with system variability, make this infeasible. Furthermore, even if such a tool could be developed, it is not clear what the objective should be, as not all users have the same values.

Thus, instead of trying to prescribe policies for a single given industrial setting and situation, we seek to develop a generalized, simulation-based *responsiveness frame-work* to help users assess a variety of sequencing policies and to analyze how these policies perform under real world conditions. Figure 1 depicts the components of the framework.

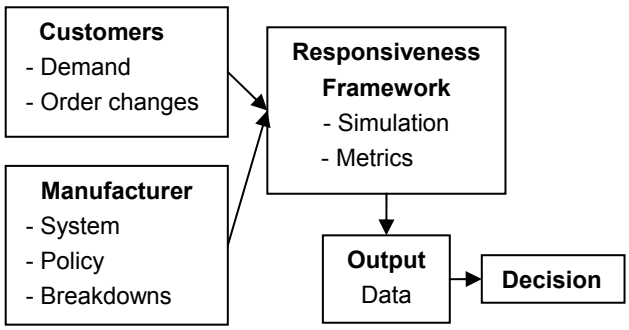

Figure 1: Responsiveness framework.

Inputs: The inputs of the framework start with the system configuration; this is currently restricted to pure serial lines. The user specifies the number of operational stages and, for each operational stage, a disruption distribution to describe the probability of machine failure (both frequency and duration). Next, the user must specify how much time each product spends in each operational stage. The changeover time between products must also be provided. Then, for each customer, the initial demand of each product is given, as well as the order due date. Each customer also has a probability distribution defining the likelihood and magnitude of order changes. In addition, an initial inventory can be specified for each product. Finally, the user specifies a policy, as defined in Section 2. Note that the input parameters are very general and thus this framework can be applied in many industrial circumstances.

Outputs: Different users will be concerned with different metrics in different circumstances. Therefore, instead of quantifying the policy's value with a single metric, we seek to provide extensive reporting capabilities on a variety of levels. Natural metrics include: makespan, time spent in machine failure, time spent blocked, overtime (i.e. difference between makespan and production horizon), shortage and holding costs, over-production by product, and final demand by product. Because the system performance is being simulated repeatedly, it is natural to report across all instances the average, variance, and minimum and maximum values of these metrics.

In addition to these, there are other characteristics that cannot be so concisely reported. Customer responsiveness falls into this category. Consider the case where two different policies result in two different outcomes. In the first policy, 99% of demand will be completed by the due date, but the remaining 1% will take an additional 24 hours. In the second policy, 95% of demand will be completed by the due date, with the remaining 5% completed within the next 30 minutes. Which policy is preferable? This depends on the particular situation – different users will have different preferences between these two policies. To help analyze the customer responsiveness of different policies, we have therefore developed a *customer responsiveness curve*. This curve shows, for every point in time from the due date until the final makespan the percent of demand still unmet. Figure 2 depicts how three different responsiveness curves for three different policies might look; note that policy A is always preferred to policies B and C, but neither policy B nor policy C is dominant with respect to the other.

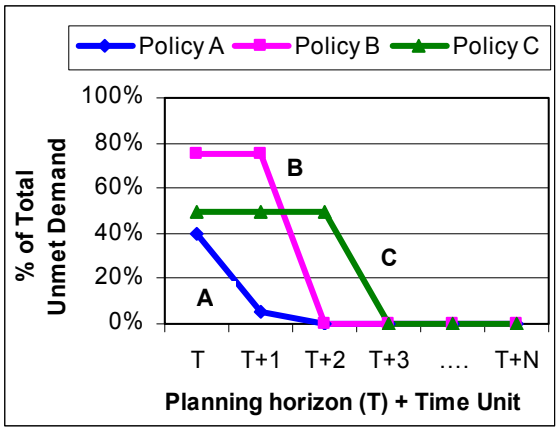

Figure 2: Customer responsiveness curves.

Computational Structure: We have completed implementation of the basic framework using C++, with many different sequencing rules and distribution functions already in place. 'Hooks' exist to extend these to additional rules and distribution functions. The flow of the software begins with the user inputting all parameters necessary to specify the system configuration, the customer data, and the policy to be evaluated. Production is then simulated a user-specified number of times. At each instance, an initial sequence is generated based on the user-specified policy. The products then begin production in this sequence. Random disruptions are generated (for both machines and customer orders), the sequence is updated when triggered, and production resumes. The process repeats until all demand is met. For each instance, several metrics are captured. After all instances have been simulated, cumulative and average metrics are computed as well.

4. SAMPLE ANALYSIS OF RESPONSIVENESS

The purpose of this section is not to make significant policy suggestions, but rather to provide a simple demonstration of the flexibility and analytical powers of the framework. To do so, we explore the link between lot size, update frequency, and customer responsiveness in a given manufacturing setting.

More specifically, we consider a flowshop with three products types and three operational stages. Each product is fairly balanced between operational stages, but is imbalanced with respect to the other products. Each product is associated with a single customer, who places an initial demand with a single due date and then during the production period randomly modifies this demand while maintaining this due date. We consider the following hypothesis: In a fixed lot size policy, it is preferable to have large lot sizes (which decrease production bottlenecks and therefore decrease makespan) and to update the production sequence frequently (so as to decrease the likelihood of overproduction).

We analyze this hypothesis in the following section. The framework enables us both to support some aspects of this hypothesis, and also to identify some less intuitive outcomes that were not initially expected.

4.1 Production Environment

We consider a flowshop with three products and three operational stages. System characteristics are given in Table 1. The planning horizon (that is, time from the beginning of production to the final due date) is 3000 time units; for the sake of simplicity, we assume instantaneous changeover.

Products	Stage 1 Time Units	Stage 2 Time Units	Stage 3 Time Units	Total Demand
A	10	9	11	100
B	2	3	4	110
C	8	7	7	95

Table 1: Processing times and initial demand

Two types of random disruptions are considered: customer changes in order quantity and machine failures.

Customer Order Changes: We assume three customers, each of whom has a due date of time 3000 time units. Customer A orders only product A and makes frequent, small changes in order quantity. Customer B orders only product B and makes frequent, medium changes. Customer C orders only product C and makes infrequent, large changes.

Customer order changes are generated from a discrete probability distribution at every 50 time units. The changes are cumulative over time. Table 2 shows these distributions. For example, at a given demand update, customer A will have no change with 0.97 probability; the order will increase by 2.5% with 0.015 probability; and the order will decrease by 2.5% with 0.015 probability.

Customer A		Customer B		Customer C	
P	**%**	**P**	**%**	**P**	**%**
0. 97	0	0. 97	0	0.99	0
0.015	- 2.5	0.01	+ 25	0.005	+ 25
0.015	+ 2.5	0.01	- 25	0.005	- 50
		0.005	+ 10		
		0.005	- 10		

Table 2: Probability distribution
P - Probability, % - Percent change

We assume that customer order changes cannot occur after the initial production horizon is complete, even if overtime is required to complete production.

Machine Failures: We consider three types of machine breakdowns. The machine in operational stage 1 has frequent, short breakdowns; in operational stage 2 has infrequent, long breakdowns; and in operational stage 3 has moderately frequent, very short breakdowns. These breakdowns are generated from uniform distributions, as depicted in Table 3.

Machine	MTTF Time Units	MTTR Time Units
1	U (70,110)	U (5,15)
2	U (150,300)	U (20,40)
3	U (125,175)	U (1,4)

Table 3: Mean time to failure (MTTF) and mean time to repair (MTTR)

Policy: Throughout the experiments, we assume the following policy. First, we do not incorporate any forecasting with regards to customer demand changes – we sequence products in the quantities given by the customer orders. Second, the initial sequence rule is fixed lot size, with given product sequence {A, B, C}. The lot size varies as part of the experiment. Third, sequence updates are triggered at fixed time intervals – every *n* units of time, the demand will be adjusted to take into account any order changes that have been made since the last update. This new demand will be resequenced according to the following rule. First, if the current product lot has not been completed, this lot will be continued, so long as it does not lead to overproduction of that product, given the newly updated demand. This is to maintain the consistency of the lot sizing approach. Then, the remaining products will be sequenced, again using the fixed lot size rule, but according to the newly updated product demand, starting with whichever product type would come next after the current product.

For example, suppose we initially wanted to produce 20 A's, 20 B's, and 20 C's. With a lot size of 10, we might begin by producing 10 A's, 10 B's, and then 7 C's before reaching the first update trigger. Suppose that during this time period, the demand for A had dropped to 18, for B had dropped to 8, and for C had dropped to 9. Then the updated sequence would have us next produce 2 C's, finishing the existing lot but dropping its lot size from 10 to 9 so as to not overproduce. The final step would be to produce a lot of size 8 A's to complete its demand; product B has been overproduced by 2 units.

This process of updating demand and resequencing repeats at every *n* time units, where the update interval also varies as part of the experiment.

4.2 Experiments and Analysis

We began with an experiment designed to highlight the relationship between product sequence and makespan. This experiment is fully deterministic – we did not permit customer order changes nor machine breakdowns. We considered lots from size one to 110 (i.e. the size at which each product type would be produced in a single lot). The framework was used to replicate production under this policy. Given that no random disruptions occur, a single iteration of the simulation produces the deterministic makespan for the production sequence.

Note in Figure 3 that, as expected, increases in lot size lead to decreases in makespan. This is because smaller lot sizes require more frequent switching between product types, which leads to more frequent bottlenecks because of imbalances from across products. Beyond a certain point, however, the decrease in makespan slows dramatically with respect to lot size. Furthermore, many lot sizes lead to virtually identical makespans, as illustrated in Figure 4. The reason for this can be illustrated with a simple example. Consider lot sizes of 70 and 71. In the absence of any customer demand changes, the sequences will be:

70 A 70 B 70 C 30 A 40 B 25 C,

and

71 A 71 A 71 C 29 A 39 B 24 C.

These two sequences have the same number of shifts from one product type to another and therefore encounter virtually the same bottlenecks.

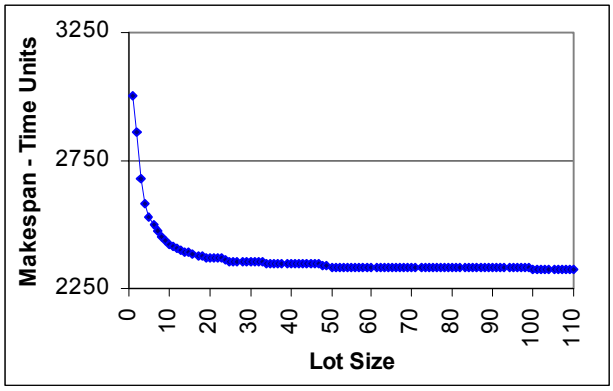

Figure 3: Makespan vs. lot size.

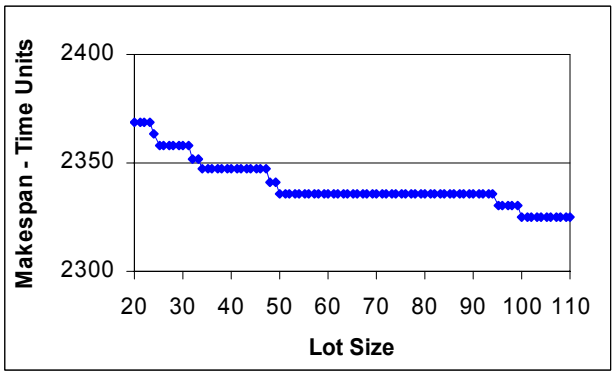

Figure 4: Makespan vs. lot size (20-110).

The results of this experiment suggest that large lot sizes are preferable. However, this basic analysis only considers the makespan assuming no disruptions occur. Furthermore, there is a limited difference in makespan once the lot size exceeds a fairly low threshold (approximately 20). Therefore, we next sought to analyze how variability affects average makespan.

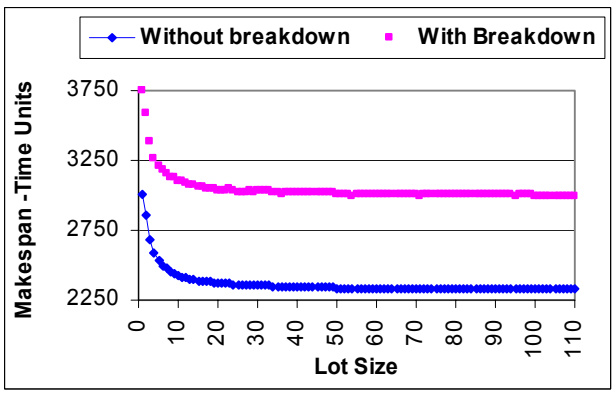

Figure 5: Average makespan with and without machine breakdowns.

We began by simulating the system 10000 times, now adding in the possibility of machine breakdowns. Figure 5 shows that, as expected, machine breakdowns delay production. There is a nearly constant difference between

the two average makespan curves for lot sizes greater than 20. Interestingly, we noted that for small lot sizes, the *absolute* change is higher but the *percentage* change is lower. The absolute change increases with decreases in lot size largely due to the fact that, because the overall makespan is much longer, there are typically more machine breakdowns occurring overall during this time period. On the other hand, the percentage increase can be attributed largely to the fact that a decrease in lot size increases bottleneck time and thus decreases the amount of time during which machines are operating. Thus, it also decreases the likelihood that a machine failure will occur at a time when the machine is operational (that is, not bottlenecked waiting for a new product) and so the machine failure will lead to additional delays.

We next began to incorporate changes in customer order quantities. First, we simulated the system 10000 times and computed for each of these instances the overall customer order changes.

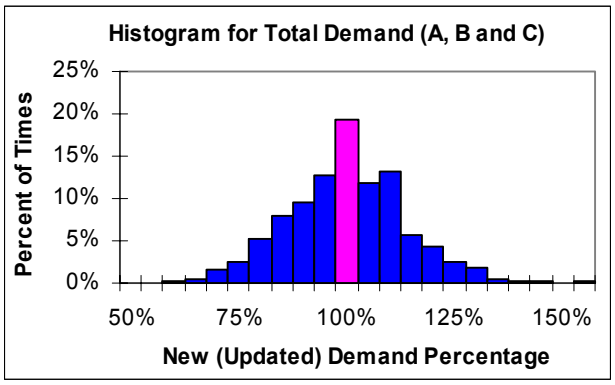

Figure 6: Variability in total demand.

Figure 6 shows that 80% of the time, total demand varied, with increases/decreases of up to 50%. These statistics understate the impact, however, because an increase in one product's demand may be balanced by a decrease in another product's demand. Figure 7 displays demand changes by individual product.

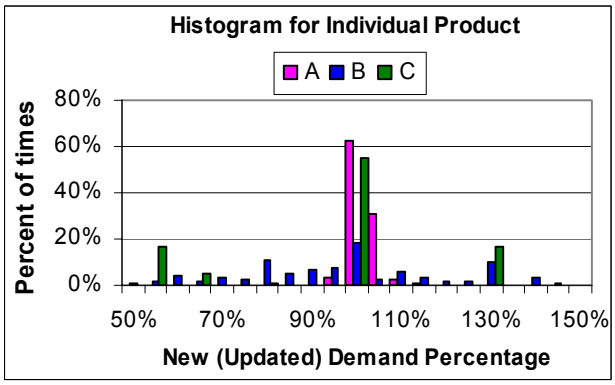

Figure 7: Variability in terms of product.

Given the significant impact that customer order disruptions and machine failures can have on the system, we next considered how adding sequence updates to our policy could help to absorb these disruptions. We considered lot sizes of 1, 25, 50, and 100, with sequence updates triggered every 50, 750, 1500, or 3000 time units. At each of these updates the demand changes were incorporated and the production sequence was adjusted to avoid overproduction.

Figure 8 shows the connection between update frequency and overproduction. As expected, the curves are increasing – as the time between sequence updates increases, so does the amount of overproduction. Note, however, that *larger lot sizes result in significantly more overproduction*, even with very frequent resequencing. This is because it is less likely that decreases in demand for the first products produced can be captured, as nearly all of these products will have been produced early in the production period.

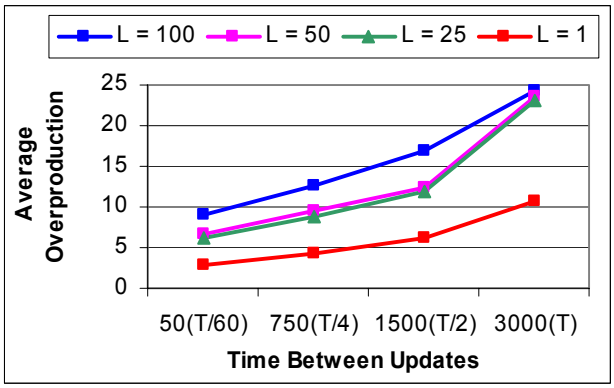

Figure 8: Average overproduction vs. frequency update. L - Lot size, T - Planning horizon.

Figure 9 demonstrates that both overproduction and makespan increase as the time between updates increases. Note that there is a significant increase in makespan with a lot size of one relative to the other lot sizes considered; even though fewer products are produced (i.e. less overproduction occurs). This is because a lot size of one results in a substantial amount of bottleneck. However, lot sizes of 25, 50, and 100 are virtually the same under all update intervals. Presumably, the relatively small improvements in makespan found by using larger lot sizes are being absorbed by the increase in overproduction as lot size increases.

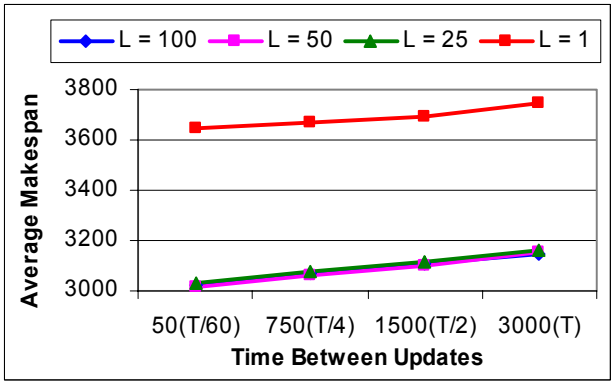

Figure 9: Average Makespan vs. frequency update.

The results thus far suggest that the benefit of very large lot sizes (with regards to expected makespan) can be achieved with smaller lot sizes (for example, 25 instead of 100), and that these smaller lot sizes result in less overproduction, independent of update frequency. The final question we set out to address was whether customer responsiveness was impacted by lot size and update frequency. Figures 10 and 11 show customer responsiveness curves for lots of size 1 and 100, with update frequencies of 50, 750, 1500, and 3000.

In both cases, we see that more frequent sequence updates are dominant for a given lot size. This is not surprising given the previous results. More frequent updates lead to less overproduction. Instead, time is used to produce desired products and thus at the end of the production horizon, a larger percent of demand has been met. Beyond this time point, no further customer order changes occur and thus the remaining production occurs in a similar fashion, independent of the original update

frequency. Notice that the curves are steeper for larger lot sizes, as production of the remaining demand occurs more quickly independent of the initial outstanding quantity.

In summary, we observe that – as expected – larger lot sizes can be used to decrease makespan and more frequent sequence updates can be used to avoid overproduction. However, less intuitively, we observe that larger lot sizes lead to more overproduction, and that the benefits of reduced makespan can be counter-acted by this increased production, both in terms of actual time to meet customer demand, and also in terms of the cost of excess inventory due to the overproduction. Furthermore, we have observed that the decrease in makespan as lot sizes increase is relatively small after a certain point, due to the fact that the number of changeovers from one product type to the next remains quite small. Thus, the benefits of decreased makespan can be achieved with more moderate lot sizes while still achieving the benefits of reduced overproduction through the use of frequent sequencing updates.

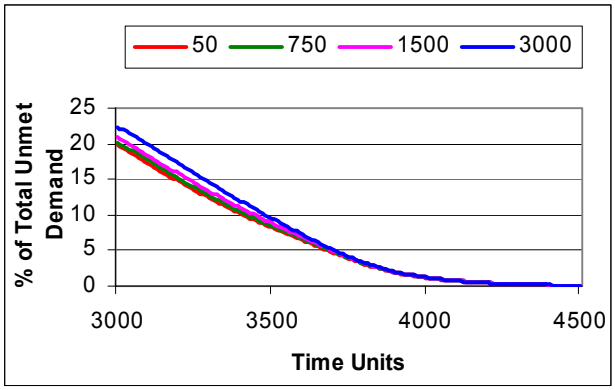

Figure 10: Customer responsiveness curve – lot size 1.

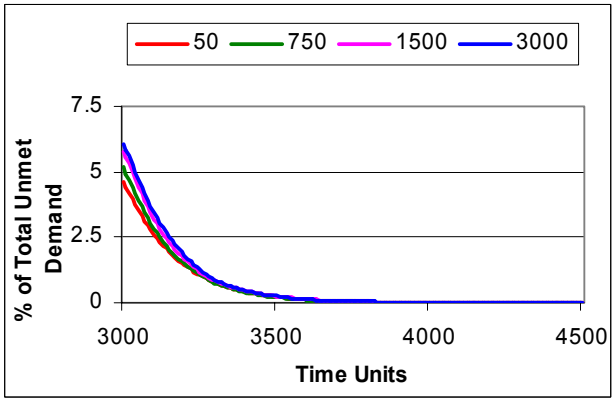

Figure 11: Customer responsiveness curve – lot size100.

5. SUMMARY AND CONCLUSIONS

Manufacturers are faced with frequent disruptions that impact the quality of their production plans. Resequencing is a relatively simple and low-cost way to address these disruptions so as to provide improved customer responsiveness. We have presented a framework to evaluate different policies for controlling sequencing decisions in production. A key strength of this framework is that it does not dictate how system performance should be evaluated. Rather, it provides flexibility to recognize that different users have different definitions of customer responsiveness and thus use different criteria for evaluating solution quality.

We provided a simple example to demonstrate how this framework can be used. Within the analysis, we both provided support for certain intuitive expectations and also identified areas where our intuition was incorrect. These results can be used to develop ideas for improved policies; the framework can then be used to assess these new policies.

Although the existing framework provides opportunity for many different types of analyses, additional extensions will be beneficial as well. These include penalties or delays for resequencing and product changeover; new sequencing rules, disruption distributions, and metrics; and more explicit incorporation of costing issues. More generally, we would like to permit a broader class of system configurations, such as serial parallel lines with crossover, so that the framework can be applied in an even greater number of circumstances.

6. ACKNOWLEDGMENTS

The authors are pleased to acknowledge the support by the National Science Foundation Engineering Research Center for Reconfigurable Manufacturing Systems under Grant No. NSF-Sub-EEC-9529125.

7. REFERENCES

[1] McCormick, S.-T., Pinedo, M.-L., Shenker, S., Wolf, B., 1989, Sequencing in an Assembly Line with Blocking to Minimize Cycle Time, Operations Research, 37/6:925-935.

[2] Abadi, K., Sriskandarajah, C., Hall, N., 2000, Minimizing Cycle Time in a Blocking Flowshop, Operations Research, 48/1:177-180.

[3] Ronconi, D.-P., 2004, A Note on Constructive Heuristics for the Flowshop Problem with Blocking, International Journal of Production Economics, 87:39-48.

[4] Dudek, R.A., Panwalkar, S.-S., Smith, M.L., 1992, The Lessons of Flowshop Scheduling Research, Operations Research, 40/1:7-13.

[5] Wiendahl, H.-H., Roth, N., Westkämper, E., 2002, Logistical Positioning in a Turbulent Environment Annals of the CIRP, 51/1:383-386.

[6] Church, L.-K., Uzsoy, R., 1992, Analysis of Periodic and Event-Driven Rescheduling Policies in Dynamic Shops, International Journal of Computer Integrated Manufacturing, 5/3:153-163.

[7] Sabuncuoglu, I., Bayiz, M., 2000, Analysis of Reactive Scheduling Problems in a Job Shop Environment, European Journal of Operational Research, 126:567–586.

[8] Wiendahl, H.-H., Lutz, S., 2002, Production in Networks, Annals of the CIRP, 51/2:573-586.

[9] Matson, J.-B., McFarlane, D., 1999, Assessing the Responsiveness of Existing Production Operations, International Journal of Operations and Production Management, 19/8:765-784.

[10] Shafaei, R., Brunn, P., 1999, Workshop Scheduling Using Practical (inaccurate) Data - Part 1: The Performance of Heuristic Scheduling Rules in a Dynamic Job Shop Environment Using a Rolling Time Horizon Approach, International Journal of Production Research, 37/17:3913-3925.

[11] Wiendahl, H.-H., Cieminski, G.-V., Begemann, C., 2003, A Systematic Approach for Ensuring the Logistic Process Reliability of Supply Chains, Annals of the CIRP, 52/1:375-380.