The Pennsylvania State University

The Graduate School

College of Engineering

SIMULATION BASED COMPARISON OF PUSH VERSUS PULL PRODUCTION IN A HARDWOOD DIMENSION MILL

A Thesis in

Industrial Engineering

by

Satinder Singh Gill

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2009

The thesis of Satinder Singh Gill was reviewed and approved* by the following:

Deborah J. Medeiros
Associate Professor
Department of Industrial and Manufacturing Engineering
Thesis Co-Advisor

Charles D. Ray
Assistant Professor of Wood Operations
Thesis Co-Advisor

M. Jeya Chandra
Professor in Charge of Academic Programs & Graduate Program Coordinator

*Signatures are on file in the Graduate School

ABSTRACT

This research focuses on two different operational strategies in a typical hardwood dimension mill - pull and push order scheduling. The use of simulation modeling to test and clarify lean production strategies in dimension mill operations is a powerful technique. A discrete event system simulation model of a hardwood dimension mill comprising different stations like planing station, ripping station, chop station, and molding station was developed for this purpose.

The objective of this research is to compare the relative impact of push and pull scheduling systems on final product inventory levels. For the mill and specific push and pull systems simulated, the push system resulted in higher inventory accumulations, throughput, and yield. The lower inventory accumulations associated with the pull configuration that was implemented did not benefit the system enough to allow greater throughput. The pull system resulted in lower utilization of machines and lower lead times.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 3

CHAPTER 4

LIST OF TABLES

CHAPTER 3

CHAPTER 4

ACKNOWLEDGEMENTS

No task, by any person, has ever been accomplished without the help and guidance of the people around him/her, and I am no exception.

To begin with, I am thankful to Dr. Deborah J. Medeiros and Dr. Charles D. Ray for their continued support and guidance throughout my Master's degree. Their combined guidance and expertise in operation research, simulation has been invaluable to me throughout the course of this research. I would also thankful to Dr. M. Jeya Chandra for reading my thesis and approving it.

I would like to thank Keith Atherholt at Lewis Lumber Products for allowing us to study their manufacturing operations. I am grateful to Janice K. Wiedenbeck at USFS, Princeton, WV for supporting our research project and providing guidance. I would like to thank all my friends, especially Ramesh and Vanathi for all their help. I am also overwhelmed with gratitude for the help given by Dr. Surinder Chopra and his wife Neena Chopra.

I express my gratitude to my uncle Jaswant Gill and aunt Ravinderjit Gill, without whose support I wouldn't have started (and finished) my MS. I thank my dad Sukhwant Singh Gill and mom Simerjit Kaur, my brother Jatinder, my sister Rinkie and all other family members for their never-ending love and support. Finally I would like to thank my wife Rupinder for bearing with me during this whole period and providing continuous support and encouragement.

I dedicate this thesis to my parents and our daughter Jessie.

Chapter 1

Introduction

1.1 Motivation

Due to intense global competition, U.S. wood products companies that remain in business have realized the importance of implementing more efficient management strategies and manufacturing tactics. A term has arisen to frame these strategies within a corporate context: The Lean Enterprise. The lean enterprise encompasses the entire supply chain including product design and engineering as well as relationships with customers (Levinson and Rerick 2002).

However, lean efforts in the hardwood industries have not yet achieved full implementation along the supply chain. Different operational components of the chain operate somewhat independently from each other. Lean production successes are typically found in cabinet or furniture assembly operations, but the dimension mill operations that feed these assembly plants are more complex and seem to be more difficult environments in which to apply traditional lean program components. Further complicating the concept is the traditional lumber supply infrastructure, in which sawmills typically manage production to optimize physical or economic yield from its log resource, rather than producing lumber to meet specific dimension producers' requirements just-in-time. This requires the dimension operators to build lead times and raw material inventory into their production schedule. Furthermore, dimension mill yield can be, and often is, manipulated by using cutting bills to optimize trade-offs between customer order files and mill yields.

Thus, within the hardwood products supply chain, the dimension operation is a key link. Because of its important role in the supply chain, this research focuses on the different operational strategies, specifically, pull (consumer demand based) and push (also called traditional manufacturing, supplier production based) flow scheduling, in a typical dimension manufacturing operation, and how those strategies may impact the overall leanness of the vertically-integrated hardwood "lean enterprise."

As will be demonstrated in the literature review (Chapter 2), the evolution of simulation modeling and analysis has seen significant application in research aimed at improving dimension mill operations. However, research in this area is far from complete considering how quickly the dimension industry is changing in response to competitive pressures. Of particular interest today is the use of simulation modeling to test and clarify lean production strategies in dimension mills – an area in which combined yield and process flow (or discrete – event) simulation should prove to be a powerful evaluation tool.

1.2 Hardwood Dimension Mill Simulation

The simulations required for this work involved Process simulation and Flow simulation.

Process simulation is the part where we configure various mill parameters related to the cutting of the lumber (like grade of the lumber, rip or saw first, optimization of cutting saws for cutting the lumber, etc). Process simulation is used to create the components (loads) that go into the Flow simulation. The tool used for Process simulation is ROMI (developed by USDA).

Flow simulation is the part where the parameters like processing times of machines, number of operators used on a machine; downtimes of machines, transfer batch sizes, batch transfer times, etc are set up. The output files generated by ROMI are fed to the flow simulation. For flow simulation, AutoMod (developed by Brooks Automation, now acquired by Applied Materials, Inc.) has been used.

1.3 Thesis Outline

Chapter 2 is a detailed description of wood products operation with a focus on problems and complexities of simulation of wood processes operations, and direction of research in such simulation. Literature about benefits of information sharing and application of simulation in supply chain is reviewed.

Chapter 3 presents the description of the model of a hardwood dimension mill. It includes details on development, methodology and logic of simulation model.

Chapter 4 discuses experimental setup and the results obtained by running the simulation model, along with a discussion on the reasons driving these results. An outline for future work that may be done in this field of study has also been discussed in Chapter 5.

Chapter 2

Literature Review

2.1 Introduction

A simulation is the imitation of the operation of a real-world process or system over time. It is a powerful tool which can be used to describe and analyze the behavior of a system. It is specifically useful in answering "What if?" questions about the real systems (Banks 2004). Both existing and conceptual systems could be modeled with simulation.

The simulations required for this work could be divided into two parts: Process simulation and Flow simulation. Process simulation is used to determine the best way to manufacture rough dimension parts from lumber. Process simulation enables the accurate modeling of a rough mill's processing situation, like equipment setup, cutting bill, lumber grade, and cutting methodology (Stiess, 1997). Process simulation is best suited to answer questions like: What yield and processing requirements could be expected from a particular cutting bill? Is the cutting bill better suited to rip-first or crosscut-first processing? What if two or more cutting bills are processed together? etc. In this research, process simulation is used to create the components (loads) that go into the flow simulation. The simulation software used for Process simulation is ROMI, developed by USDA (Weiss & Thomas, 2005).

Flow simulation allows the user to try out different plant layouts scenarios, number of machines to use, number of operators required, transfer batch sizes, transfer

rates (frequency and time), effect of machine breakdowns etc. For flow simulation, AutoMod (Banks, 2004), developed by Brooks Automation, has been used.

In the following sections of this chapter, a summary of past research relevant to this research effort is provided. The chapter is divided into three main parts. In the first part, past research in flow simulation related to dimension mills is reviewed. The second part is focused on process simulation. In the third part, literature related to analyses of push and pull systems is reviewed.

2.2 Flow Simulation Related to Dimension Mills

Many researchers have recognized the critical role of the dimension operation and utilized computer simulation to evaluate processing improvement opportunities.

Araman (1977) demonstrated how complex processing questions, like the effects of different grades of lumber on production rate and equipment utilization, could be answered by computer modeling. He used GPSS (General Purpose Simulation System), program available from IBM at that time, to simulate a rough mill for the production of interior furniture parts and experiment with different layouts of the mill.

Anderson (1983) used simulation to evaluate a crosscut-first furniture rough mill using a combined continuous/discrete model, developed with the FORTRAN-based General Activities Simulation Program IV (GASP IV). The developed simulation allowed determination of total processing costs for individual parts produced in the rough mill. Anderson described input lumber by assuming normal distributions for board length, width, and thickness. He later demonstrated that modern simulation languages

could be used to accurately simulate the workings of a typical dimension mill (Anderson 1985).

Using the SIMAN/CINEMA (FORTRAN based) simulation language, Kline et al. (1992) expanded the mill simulation concept, demonstrating detailed mill analysis capability, such as cost, component yield, throughput, and waste production over time. They used simulation to help make decisions such as when to add equipment, when to renovate, or how to improve throughput subject to time-varying demands in hardwood dimension mills. They used animation to reduce the time for model development and for communication purposes such as illustrating "how" and "why" a given solution can be effective, or in demonstrating queuing problems to management.

Wiedenbeck (1992) utilized simulation specifically to study dimension mill processing alternatives. She ran simulation experiments to determine the economic consequences of processing short, medium and long lumber in the crosscut-first rough mill. The simulation model generated average values for rough part production volume, part value and various machine utilization rates. These production figures formed the basis of an economic analysis of the feasibility of producing rough dimension parts from short lumber. Her simulation showed that the production of rough furniture parts from short lumber in a crosscut-first rough mill can be economically feasible with careful matching of cutting bills and short lumber input schedules.

Lin et al. (1994) explored the potential of processing dimension parts directly from logs, by eliminating the intermediate steps of lumber manufacturing, grading and trading. Their simulation estimated the potential cutting yield of dimension parts and potential value recovery obtainable from Factory Grade 2 and 3 red oak sawlogs under

various processing configurations and cutting bills. The results indicated that Grade 2 logs produced higher dimension yield than Grade 3 logs. However, Grade 2 logs resulted in much less value recovery per dollar log input than Grade 3 logs because of the notably higher price of Grade 2 logs. Their results indicated that the combination of longer cuttings and shorter cuttings in a cutting bill can offer higher value recovery than only having shorter or longer cuttings in a cutting bill, suggesting that shorter cuttings and longer cuttings should be combined in a production run, whenever possible, to recover the maximum value from given logs. To prove the potential feasibility of a direct-processing system, Lin et al. (1995) used simulation to design and evaluate four different direct-processing mills. They found that, depending on the mill designs and cutting bill, processing Grade 2 logs resulted in 30 – 64 percent more rough dimension production rate compared to processing Grade 3 logs. A cutting bill that included many shorter lengths increased the load on the rough mill components of designs studied.

Gazo and Steele (1995) introduced a Rough Mill Analysis Model (RAM), a unique simulation that combined the lumber yield simulation CORY (Brunner et al. 1989) with a process flow discrete event simulator based on the computer simulation language SLAM (Pritsker 1986). They utilized this combination tool to determine the impact of pre-sorting the lumber entering the operation on dimension mill yield (Steele and Gazo 1995).

Recognizing the need to better elucidate discrete-event simulation modeling methodology as it relates to wood processing, Wiedenbeck and Kline (1994) provided an in-depth characterization of the computer simulation model development life cycle. They used actual case studies in modeling furniture rough mill to assist in illustrating the

simulation modeling life cycle. These early studies highlighted that more realistic dimension mill simulations were generated when part yields were based on data from real lumber. During their data collection phase for developing the model, they identified the operations which were most affected by lumber length and piece counts. The cutting bills used in the mill studies were selected by the rough mill supervisors. They made an attempt to choose a cutting bill that would match up well with the short length lumber input. The piece rates and cutting length distributions associated with these cutting bills were key input variables for their simulation models. Additional time data were also collected during the mill studies. These data, gathered at each of the primary rough mill cutup operations, consisted of single operator timings, by lumber length and grade.

They (Wiedenbeck and Kline,1994) classified their simulation model as self-driven (Kobayashi 1978). Variable values in self-driven models are based on probabilistic distributions specified by the modeler. Occasionally, a highly specific simulation model will be trace-driven (Kobayashi 1978). Trace-driven models are driven by a series of numeric inputs of actual data from the real system. They figured that rather than sampling from a distribution, actual board length and width data could have been entered into the simulation model. For their simulation model to be trace-driven, very large body of data needed to be available. This observation started research towards generating part yields based on data from real lumber.

2.3 Process Simulation Related to Dimension Mills

To address this requirement (of generating part yields based on data from real lumber), Gatchell et al. (1993) produced a databank of red oak lumber with the digital description of the clear area and defect characteristics of hundreds of boards in each lumber grade. Their databank contained 10,718 board feet in 1,578 boards. They used NHLA's (National Hardwood Lumber Association) Special Kiln Dried Rule to grade the lumber in the databank because it counts all defects and treats each board as if it were air-dried (NHLA 1990). This rule states that each kiln-dried board will be graded as if it were air-dried and that it will be graded with all defects counted. With the advent of the Gatchell database, dimension mill simulations could realistically simulate part yields obtained from the heterogeneous lumber raw material.

Thomas, working to develop an industry-standard lumber cut-up simulator, produced ROMI-CROSS (Thomas 1996), to simulate product yields from Gatchell's database in a crosscut-first dimension mill; ROMI-RIP (Thomas 1998) to do the same in a rip-first environment; and then ROMI 3.0 (Weiss and Thomas 2005) to combine both system capabilities into one tool.

Thomas and Buehlmann (2002, 2003) validated and tested the performance of the ROMI-RIP tool by comparing gang-rip, chopsaw, and overall yields to those obtained in an actual state-of-the-art rough mill. They digitized a 930-board-foot lumber sample by recording all defect locations, and types as well as board size and grade, allowing the rough mill and ROMI-RIP to process identical lumber samples. They considered the following performance criteria for the validation:

Overall yield: the total yield resulting from strip-cutting and crosscutting strips to part lengths, *Strip yield:* the yield obtained by the two systems when converting boards to strips, and *Crosscut yield:* the yield obtained by the two systems when chopping strips to part lengths. The validation was defined as successful when no statistically significant yield differences could be detected at the 95 percent significance level They concluded that the tool accurately simulated actual dimension mill results as long as the simulation settings were accurately specified to represent the mill being modeled.

Given the lumber databank and the ROMI yield (cut-up) simulation programs, scientists and practitioners have been able to explore a multitude of operational questions. Machine productivity ratios as generated through yield simulation studies using an unpublished software simulation named "Cut-Sim" conducted by Steele et al. (1999) proved the importance of considering cutting bill characteristics when making decisions about lumber grade mix for the dimension mill.

Most of the computer simulations modeled the parts production process as batch processing, by which all parts are produced by one saw of each type (Steele and Aguirre, 2004). A rip-first rough mill in which a single gang ripsaw is followed by a single optimizing crosscut saw is an example of such batch processing. However, in rough mills employing multiple manually operated crosscut or straight-line ripsaws, the total cutting order is segmented into smaller cutting orders that are assigned to multiple machines. They (Steele and Aguirre, 2004) termed this segmentation of the cutting orders as segmented processing. Using the same Cut-Sim simulation tool, they proved that segmented cutting orders, common to mills employing multiple manually operated crosscut or straight-line ripsaws, result in lower mill yields than running the entire cutting

order in a single batch. This confirmed the independent findings of Thomas and Brown (2003) of the effects of utilizing more sorting capacity (i.e., the number of lengths and widths cut simultaneously) on yield for various lumber grade mixes and typical industrial cutting bill combinations. Using an analysis of variance, they (Thomas and Brown, 2003) observed significant yield increases as a result of increasing sorting capacities. However, they observed that a plateau was reached around 18 to 20 part sizes, where additional sorting capacity increases result in negligible yield gains.

2.4 Comparative Analyses of Push and Pull Systems

The push (supply-based) and pull (demand-based) operational scheduling systems are descriptive terms distinguishing two basic production control strategies for discrete manufacturing processes (Spearman et al. 1990). A push system "**schedules** the release of work **based on demand**", while a pull system "**authorizes** the release of work **based on system status**" (Hopp and Spearman 2000).

The push system, widely employed in both material requirements planning (MRP) systems and manufacturing resources planning (MRP II) systems, is considered the traditional material management approach, associated in most cases with "mass production." It has been effectively applied to various production lines, reducing inventory and improving customer service (Spearman et al. 1990).

The pull system is characteristically associated with lean production, exhibiting distinct and demonstrable advantages with respect to material flow and lead time, stability of inventory, and reduced carrying cost (Allen et al. 2001). They stated that pull systems help establish exceptional flow, which in turn leads to lower inventories, better

quality, less floor space, better communications, quicker responses to problems, and faster throughput.

However, due to its two fundamental conditions - the minimum inventory requirement of each product at every machine and the steady demand of repetitive products - the pull system may not be applicable to all production lines (Suri 1998). He stated that pull system is best applied to products with stable demand and high volume. The applicability of pull systems to different types of wood products operations remains to be determined.

A useful concept for understanding the impact on system performance of push and pull systems is the *push-pull interface*, perhaps most easily explained as the place of inventory accumulation in the process. Acknowledging that most real-world production systems exhibit elements of both push and pull systems, Hopp and Spearman (2000) provide an excellent discussion of the importance of the push-pull interface and its role in manipulating the trade-off between flexibility and shorter lead times. They effectively demonstrate that by moving the push-pull interface closer to the customer, lead times can be reduced, but at the expense of decreased flexibility in manufacturing. Interestingly, they use the example of a wood product process (plywood) to illustrate a condition when the low number of different finished products would allow for the push-pull interface to be set at finished goods, resulting in short lead times. In contrast, they mention a PC assembly plant (roughly comparable to a wood cabinet assembly operation) as a case where the large number of finished product combinations requires the push-pull interface to be moved upstream.

However, in order to study the mechanisms of each system, research typically has compared and contrasted pure push or pure pull systems (Chen et al. 1997; Kimura and Terada 1981). We take this approach as well.

A few investigations have compared the performance of pull and push systems in non-wood products operations. Hopp and Spearman (2000) evaluated the differences and similarities between the two systems in a descriptive manner. Kimura and Terada (1981) assessed relative system performance with respect to fluctuation in product demand, concluding that the pull system responded more effectively to the changed demand than the push system.

Considering factors such as forecast error, vendor influence, buffer mechanisms, product structures, facility design, scrap loss, equipment failures, worker flexibility, inventory accuracy and lot sizing rules, Krajewski et al. (1987) compared percentages of past due product and total inventories for the two systems. Their research indicated that the particular parameters of the manufacturing environment greatly affect the performance of the pull system. This conclusion was supported by Monden (1983), who pointed out that, in general, the pull system is not feasible when production runs are short, demand fluctuations are unpredictable, production setup time is long, or the cost of scrap is high.

Spearman and Zazanis (1992) developed mathematical models to compare the pull and push systems theoretically, demonstrating through their work the overall superior performance of the pull system. Their models indicated that the pull system exhibits better controllability and experiences less block-out of production flow.

On the contrary, Krishnamurthy et al. (2000) concluded that a push system is more feasible in an operation that produces frequently changed products and processes unstable customer orders (both characteristics of the common hardwood dimension mill operation). Their simulation study revealed a lower work-in-process (WIP) inventory level in the push system and demonstrated the pull system's negative performance characteristics due to high variability of the products. Conflicting findings in the literature such as these continue to fuel the debate on the appropriate application of the two systems.

2.5 Summary

In the previous sections of this chapter, a summary of past research relevant to this research effort has been provided. Sections 2.2 and 2.3 illustrate the extensive amount of work devoted to hardwood dimension mill modeling, differentiate between flow simulation work and process simulation work, and explain how the first area of work led to the second. Section 2.2 also establishes shortcomings in past dimension mill flow modeling, and lays the groundwork for the potential to use further development of the technology for more targeted investigation of modern manufacturing issues relevant in the production of hardwood dimension mill parts. More specifically, Section 2.4 establishes a foundation for further investigation of the issue of push versus pull manufacturing so that its study in the dimension part production is seen as both relevant and timely.

Chapter 3

Hardwood Dimension Mill Simulation Model

3.1 Introduction

This research focused on the different operational strategies, specifically, push and pull flow scheduling, in a typical hardwood dimension mill manufacturing operation. The use of simulation modeling to test and clarify lean production strategies in dimension mill operations appears to have tremendous potential. The objective of this research was to compare, through simulation, the relative performance of push and pull systems with respect to the yield, work-in-process, production throughput, lead time and machine utilization.

This work uses two types of simulation software to analyze a typical rip-first hardwood dimension mill. ROMI (developed by USDA) is used for process simulation. AutoMod™ is used for flow simulation.

Process Simulation simulates yield from the available lumber based on the available digitized data for the particular lumber grade. Flow simulation simulates the product flow between operating stations and within the factory to determine work-in-process inventory levels, system bottlenecks, conformance to schedule etc.

3.1.1 Dimension Mill Set-up

The set-up of the dimension mill for flow simulation is based on an actual hardwood dimension mill in Pennsylvania. The mill has a Planer, a Rip machine for Primary Operations, a salvage Rip machine, a Chop machine and three molders. For our study, we have assumed that all the boards first go through the Planer, then they are

Ripped and sent for Chop, and after chopping, the pieces go through the Molders. Figure 3.1 demonstrates how a raw board is converted into dimension components while flowing through the mill.

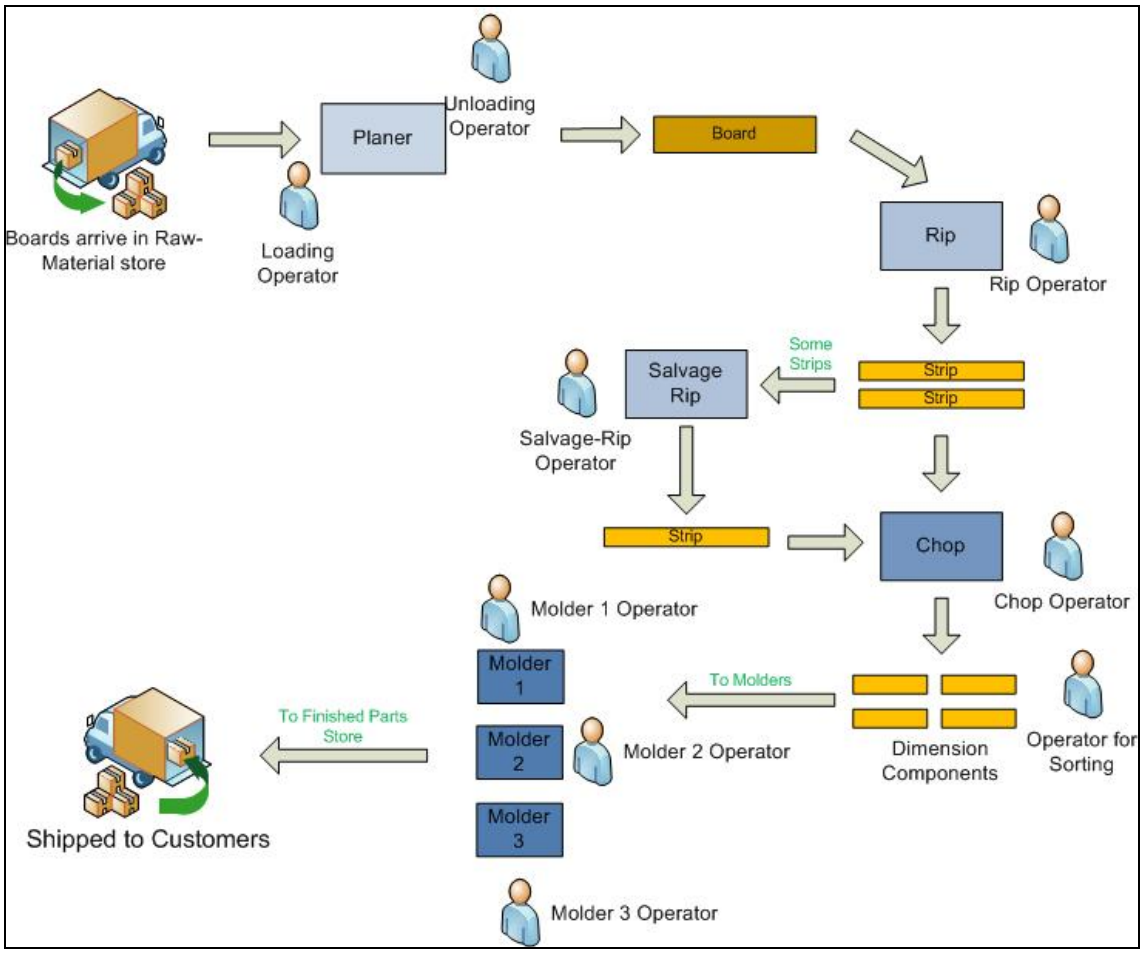

Figure 3.1 Flow of Boards in the mill

3.2 Process Simulation

In process simulation we configure various mill parameters related to the cutting of the lumber (like grade of the lumber, rip or saw first, optimization of cutting saws for cutting the lumber, etc). For our research, the simulated mill is set up to Rip first in accordance with the design of the actual mill being studied. A cutting bill, which contains

the quantities and part sizes (length and widths) required, is input into ROMI, a lumber board data file is selected, and a run of ROMI is made. The results of the ROMI run are different output files which contain information about the boards used, the quantities and part sizes obtained after Rip, and the quantities and part sizes obtained after Chop. The output of ROMI also shows the yield obtained from the run. The following sections discuss in detail, the various steps involved in Process Simulation.

3.2.1 Cutting Bill

A cutting bill is a list of parts and the respective quantities required to be produced during a production run. To quote Lamb (2002):

"The cutting bill should be evaluated from three perspectives: (1) the cutting length component, (2) the cutting width component, and (3) the quantity component."

This list of parts and their respective quantities and dimensions are determined from the available/predicted orders. Table 3.1 shows the cutting bill used in this research. As can be seen, it lists the quantities of the parts required along with their dimensions. For example, 200 pieces of length 10.0 inches and width 1.75 inches are needed from the current production run. In total, there are 17,900 parts required on this cutting bill for one day. The total cutting bill represents 32,033 lineal feet (384,440 inches) of finished parts.

Table 3.1 *Customer order (cutting bill) used in the simulation study (adapted from Steele et al. 1999).*

	Width						
Length (in.)	1.75 in.	2.00 in.	2.25 in.	3.75 in.	4.50 in.	5.00 in.	5.25 in.
10.00	200	0	0	0	0	0	0
12.25	100	0	0	0	0	0	400
13.00	0	0	400	0	0	0	0
13.50	0	0	0	100	0	0	0
14.50	0	0	150	0	0	50	0
15.00	1100	2000	0	0	0	0	200
18.75	800	400	800	0	0	0	0
20.50	200	100	900	0	0	0	0
21.00	0	0	0	0	300	0	0
22.50	0	0	800	0	0	0	0
24.75	2300	1300	2300	0	0	0	0
27.75	1000	0	0	0	0	0	0
28.25	0	0	1400	400	0	0	0
31.50	0	200	0	0	0	0	0

The objective of a dimension mill is to produce the parts required by the cutting bill at the lowest overall cost (Buehlmann, 1998). Approximately a third of the cost of operating a dimension mill is the cost of raw material. Hence, any increase in the number of rough parts produced from a given volume of raw material will markedly affect profit margins (Conners et al., 1990).

3.2.2 Lumber Mix Representation

The board data files used in this research were the files 2AC-1.dat through 2AC-9.dat provided with the ROMI software. These files are selected sub-samples of the Red Oak Lumber Data Bank (Gatchell et al. 1998), and represent a large percentage of the boards contained in the data bank's listing of 2A Common lumber. This grade is a standard grade for cabinets, millwork, and other uses requiring medium to short cuttings,

and is a favorite grade of the mill being modeled. Another reason for choosing this grade was that ROMI has the largest number of different boards of this grade in its database for use in satisfying a cutting bill. ROMI contains nine different board data files for 2A Common. However, none of the data files contain enough boards to satisfy the cutting bill as formulated for the study. So for one run, a particular data file had to be selected a number of times in order for the cutting bill to be met.

The "Makefile" feature of ROMI was used to create data files large enough not only to satisfy a daily cutting bill for this particular mill configuration, but for other, larger mills, or cutting bill requirements. For example, one large file for 2A Common lumber which contained roughly 6,000 boards of over 30,000 lineal feet of lumber was created from the 2AC-1.dat file in ROMI. This process was repeated for each of the other files, resulting in nine different files, each more than capable of filling a (one-day) cutting bill.

For this research, the simulation was to be run for 20 working days of the mill and a different set of boards would be used for each day. The figure of 20 days was decided as it would be an adequate representation of 1 month of production (4 weeks with 5 working days in each week).

To simulate the mill for 20 different days, it was important that a different set of boards of lumber were fed into the mill on each day. The first nine sets were the original data files provided by ROMI; the eleven other board data files were created by selecting the boards in the data files from 2AC-1.dat to 2AC-9.dat using the "Makefile" feature of ROMI.

Table 3.2 shows the 20 files used and the data sources (base file in ROMI) used to produce them. For example, Day 1 uses dataset 2AC-1 duplicated 70 times and so forth up to Day 9, Day 10 uses data source 2AC-all duplicated 6 times, Day 11 uses a mix of datasets 2AC-1 and 2AC-2 each duplicated 36 times, and so forth.

Table 3.2 *Data source files used for each day.*

Day	Data file used	Base File from Romi
1	2AC-1x70.dat	2AC-1.dat
2	2AC-2x70.dat	2AC-2.dat
3	2AC-3x70.dat	2AC-3.dat
4	2AC-4x70.dat	2AC-4.dat
5	2AC-5x70.dat	2AC-5.dat
6	2AC-6x70.dat	2AC-6.dat
7	2AC-7x70.dat	2AC-7.dat
8	2AC-8x70.dat	2AC-8.dat
9	2AC-9x70.dat	2AC-9.dat
10	2AC-allx6k.dat	2AC-all.dat
11	2AC-1n2x36.dat	2AC-1.dat and 2AC-2.dat
12	2AC-3n4x36.dat	2AC-3.dat and 2AC-4.dat
13	2AC-5n6x36.dat	2AC-5.dat and 2AC-6.dat
14	2AC-7n8x36.dat	2AC-7.dat and 2AC-8.dat
15	2AC-2n3x36.dat	2AC-2.dat and 2AC-3.dat
16	2AC-1n9x36.dat	2AC-1.dat and 2AC-9.dat
17	2AC-5n8x36.dat	2AC-5.dat and 2AC-8.dat
18	2AC-6n7x36.dat	2AC-6.dat and 2AC-7.dat
19	2AC-3n9x36.dat	2AC-3.dat and 2AC-9dat
20	2AC-1n4x36.dat	2AC-1.dat and 2AC-4.dat

3.2.3 Running ROMI-3

Figure 3.2 shows the main ROMI-3 interface window, with Rip-first setup.

Figure 3.2 Main ROMI-3 Interface Window (Rip First Mode).

To represent a pull manufacturing system, the mill setup (for ROMI) was as shown in the figure 3.2 below:

```
ALL part measurements are in INCHES!
Panel Specifications: Min. part width: 1.00       Max. part width:   4.50
Random width edging strips are not acceptable for use in panel production
Arbor type is  ALL BLADES MOVABLE
  Width Ranges:
   0.90   1.60   2.10   2.30   2.80   3.10   3.60   4.10
   1.60   2.10   2.30   2.80   3.10   3.60   4.10   6.50
Rough mill central controller priorities updated continuously
Primary operations avoid orphan parts.
Board placement with respect to arbor optimized in 4/16-inch steps
2/16-inch ripsaw kerf                       2/16-inch chopsaw kerf
Boards will NOT be end trimmed.
Salvage uses primary widths.                Salvage uses primary lengths.
```

Figure 3.3 Mill setup for Pull System.

Width Ranges need to be specified for the purpose of data presentation. Romi's User Guide suggests not to choose primary widths to begin or end a range as that may result in inaccurate subtotals, in the summary tables created by ROMI. For example, to specify a range that includes the two widths 1.5 and 2.0 inches, a lower range value of 1.45 and an upper value of 2.05 needs to be specified.

"Min. and Max. part widths" are the minimum and maximum primary width that are specified when processing random width part sizes.

"All-Blades-Movable Arbor" is an arbor on which all blades are allowed to move to generate specified or random-width strips that best match the characteristics of the

board and the demands of the cutting bill. Use of this option maximizes the number of alternative ripping solutions to increase rip yield from each board, but its use in simulated scenarios is necessarily limited to mills that actually have this type of rip machine or are studying the possibility of investment in this particular type of saw technology. Other rip saw settings that are available to simulate in ROMI are "Fixed-Blade", "Best-Spacing Sequence", "Fixed-Blade-Best-Feed", "Fixed-Blade with Outer-Floating-Blade", "Best-Spacing-Sequence with Outer-Floating-Blade", "Selective-Rip", and "Simple Best-Blade-Fixed-Feed". "All-Blades-Movable" was utilized throughout this study since that was the type of technology deployed in the study mill.

Kerf is the amount of wood removed by a saw blade.

The Rough mill central controller "priorities updated continuously" setting means that part counts and priorities are updated continuously. Before cutting the next part, ROMI takes note of the parts on the cutting bill that have already been cut. It then looks into the remaining parts that are needed, and then decides on what length and width of parts to cut from the next board in question. The cutting priorities change dynamically as the cutting bill is filled. For example, large parts that may be typically difficult to cut from a certain grade lumber enter the run with a high priority, causing them to be cut early in the simulation even if more optimal cutting solutions are available. However, as these parts are cut and the remaining quantity to be cut decreases, the priority calculated for that component is decreased accordingly, allowing more optimal solutions to be cut from each board. Then, as the simulated run nears its conclusion, the priority of the large piece may then be increased again if it has not yet been fulfilled through chance. In other words, as the quantity requirements for a part size are met, emphasis shifts to other sizes.

The “Primary operations avoid orphan parts” and “Salvage cuts to cutting bill requirements” settings instruct the simulator not to cut excess primary parts without determining whether the area can be salvage ripped to obtain a narrower cutting bill part. For the pull simulation, selection of these settings limit the number of parts cut to only those which match the current cutting bill requirements, and do not allow “spare” parts to be generated unless they are the result of an additional salvage operation that produces one or more companion parts that are currently called for on the cutting bill. In this way, in-process inventory is minimized in keeping with pull production principles.

Salvage cuts are those parts that are obtained by at least one additional cutting operation. The additional work makes these parts more expensive to produce but allows additional cutting bill parts to be recovered from pieces that are the unused remnant of the primary ripping operation, thereby increasing the part yield obtained from each piece of lumber. For the pull simulation, excess salvage, that is, salvage cuts allowed even though not called for in the current cutting bill, is not allowed as indicated by the selection of “Salvage cuts to cutting bill requirements” in Mill Control and the notation in Figure 3.3 that “Salvage uses primary widths” and “Salvage uses primary lengths”.

Figure 3.4 shows a strip where all possible primary parts have been cut from the full-width clear area and one salvage part from the right end. In this example, only the left-most parts are needed by the cutting bill. The remaining primary part is an excess or orphan part.

Another part, labeled Salvage, is shown in the Figure 3.4. It can be seen that the salvage part cannot be produced in the primary operation, since the rip required to produce that part would necessarily rip the needed parts, and therefore not produce those particular parts. In this case, the ROMI simulator, recognizing the potential cutting bill

part in the Salvage remnant, would produce that part in simulation according to the Salvage requirements as selected in Mill Control. If "Salvage cuts to cutting bill requirements" is selected, as it was for the pull simulation, then no salvage cut would be made if the remnant piece did not allow for at least one cutting bill piece to be produced.

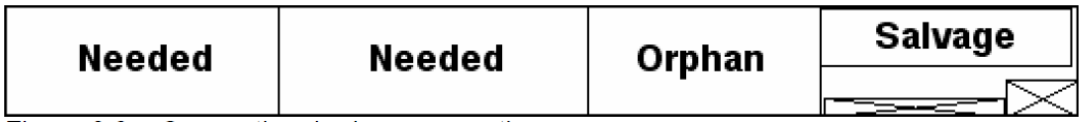

Figure 3.4 Conventional Primary operation (adopted from ROMI-3: Rough Mill Simulator 3.0 User's Guide)

The mill setup (for ROMI) to represent a push manufacturing system was as shown in the figure 3.5 below:

```
ALL part measurements are in INCHES!
Panel Specifications: Min. part width: 1.00       Max. part width:   4.50
Random width edging strips are not acceptable for use in panel production
Arbor type is  ALL BLADES MOVABLE
 Width Ranges:
   0.90   1.60   2.10   2.30   2.80   3.10   3.60   4.10
   1.60   2.10   2.30   2.80   3.10   3.60   4.10   6.50
Rough mill central controller priorities updated continuously
Primary operations DO NOT avoid orphan parts.
Board placement with respect to arbor optimized in 4/16-inch steps
2/16-inch ripsaw kerf                      2/16-inch chopsaw kerf
Boards will NOT be end trimmed.
Salvage uses primary widths.               Salvage uses primary lengths.
```

Figure 3.5 Mill setup for Push System.

The main difference in the (ROMI) mill setup for Push and Pull systems is in the production of orphan parts. An orphan part is a primary part that is cut but is not needed; that is, the cutting needs for a particular part size have been met already (Weiss & Thomas, 2005). In other words, an orphan part is an "extra (in excess of the required quantity)" piece.

The pull system avoids production of orphan parts while the push system does produce orphan parts. The logic behind this modeling decision reflects one way to simulate principles of the pull strategy, which focuses on making only those parts required by the customer in the quantity required; and of the push strategy, which optimizes product yield from raw material regardless of current production requirements, saving excess parts for future use as in-process inventory.

The other difference in Mill Control settings for the push system is that “Salvage cuts to cutting bill requirements” is not selected. This allows ROMI to generate a part from any remnant piece such as shown in Figure 3.4, even if that piece is not currently called for in the current cutting bill. ROMI would then cut any remnant piece to the largest possible piece it could using width and length measurements called for in the cutting bill, but not necessarily in the combination called for by the cutting bill. The result is a piece that will reside in in-process inventory until some future cutting bill calls for a part with that specific dimension. In actual mill operation, the mill scheduler uses his expertise to feed the mill machines certain piece dimension allowances for excess inventory from both primary and salvage operations, expecting that there will be a high likelihood of future demand for those pieces in the near future. In this respect, most dimension mill operations still operate in “push” mode.

3.2.4 Output Files of Process Simulation for Flow Simulation

After making a run of ROMI, two flow-simulation input files, Primary parts (with extension '.fs1') and Salvage parts (with extension '.fs2') are obtained.

A further distinction in the Push and Pull systems as modeled in this project is the final disposition of salvage parts produced by the ROMI simulator. In order to create as large a difference in in-process inventory situations between the two systems as could be produced, the Push system was defined to allow Salvage part production while the Pull system does not. Since there is no setting in ROMI by which the modeler can avoid making Salvage parts, this difference is handled in the flow simulation. In the flow simulation, the Push system code reads the salvage parts file produced by ROMI, while the Pull system code does not.

The Primary parts file contains data on the parts obtained from the primary operations. Figure 3.6 shows a portion of one of the Primary parts files. The first two columns contain the width and length (in inches) respectively of the board going into the mill for Rip. For example for the first board, the width is 7.75 inches and the length is 145 inches. The third column gives the number of strips obtained from the board after the Rip operation. For the first board, 4 strips are obtained after the Rip operation. The next columns give the width of the strips (in $1/16^{th}$ of an inch) and the respective number of parts obtained from the strip after the chop operation. In this case, the first strip is 28/16 inch wide and it yields 6 parts after chop, the second strip is also 28/16 inch wide and yields 5 parts after chop, the third strip is 32/16 inch wide and yields 3 parts after chop and the fourth strip is 28/16 inch and yields 4 parts after chop. The next columns give the

respective lengths (in 1/16th of an inch) of the parts obtained from the strips after chop. For example the lengths of the six parts obtained from the first strip after chop are 444/16, 160/16, 444/16, 396/16, 160/16, and 160/16 inches. The lengths of the 5 parts obtained from the second strip after the chop operation are 444/16, 300/16, 444/16, 240/16, and 160/16, and so on.

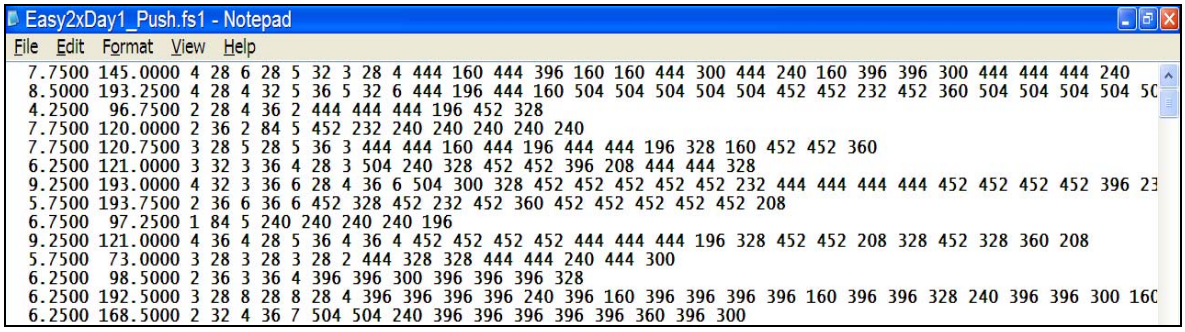

Figure 3.6 A Primary parts file

The Salvage parts file contains data on the parts obtained from the salvage operations. Figure 3.7 shows a portion of one of the Salvage parts files.

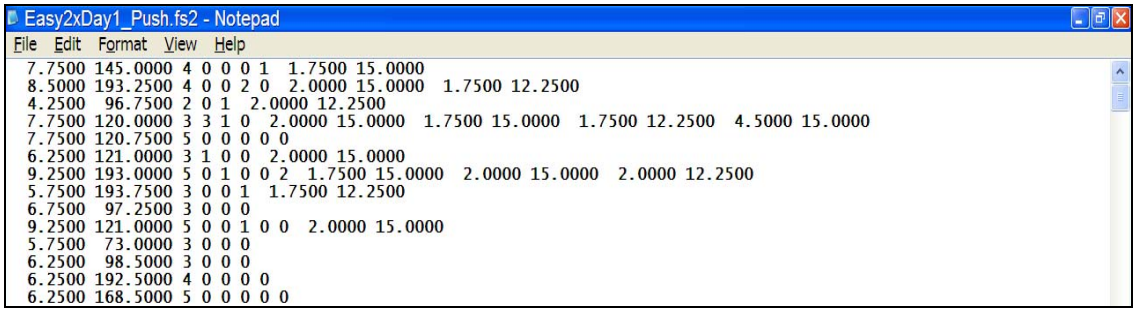

Figure 3.7 A Salvage parts file

The first two columns contain the width and length (in inches) respectively of the board going into the mill for Rip. For example for the first board, the width is 7.75 inches

and the length is 145 inches. The third column gives the number of strips obtained from the board after the Rip operation. For the first board, 4 strips are obtained after the Rip operation. The next columns give the number of salvage parts obtained from the all the strips after chop. In this case, no parts are obtained from the 1^{st}, 2^{nd}, and 3^{rd} strip and 1 part is obtained from the 4^{th} strip. The next two columns give the length and width of the salvage part (in inches). In this case, the length and width of the salvage part obtained from the 4^{th} strip are 1.75 inches and 15 inches respectively.

The output files from ROMI also contain some parts that are not on the cutting bill. These extra parts are the combinations of the width and lengths mentioned on the cutting bill, but have dimensions different from the parts on the cutting bill. We call these parts “not-needed” parts. For example, the cutting bill used for this research (Table 3.1) has no requirement of a part of width 1.75 inches and length 14.5 inches. But ROMI does produce this part.

3.3 Flow Simulation

For flow simulation, we used AutoMod, which uses a discrete-event simulation model. A discrete-event model represents the components of a system and their interactions (Banks, 2004). The flow simulation model is dynamic, in that the passage of time plays a crucial role.

For this research, the system of interest is the dimension mill and its various components are the machines (planer, rip, salvage rip, chop, and molders), people (operators), other equipment (fork-lifts, etc) and material (boards, strips, parts).

The parameters like processing times of machines, number of operators used on a machine; transfer batch sizes, batch transfer times, etc are set up and the system is observed for a period of time (for 20 8-hour working days).

3.3.1 Description of Flow Simulation Model

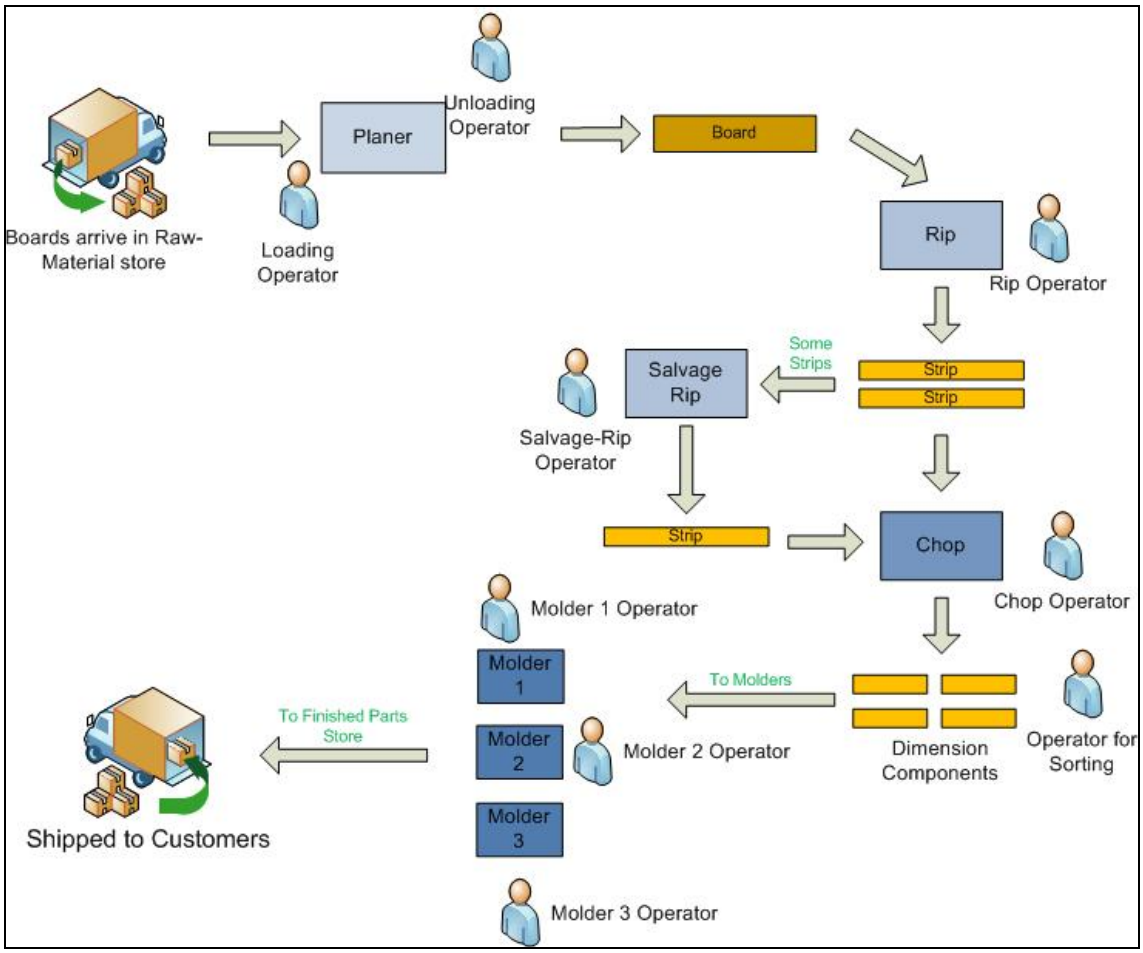

Figure 3.8 Layout of the mill

Figure 3.8 shows the general layout of the mill. Raw-material (boards) are received and stored in the raw material store. Pack of lumber is transferred from the raw material store to the planing station, using a forklift, where they are planed to get a smooth surface. There are two operators for loading and unloading of the boards at the Planer. The planed boards are stored in the output queue of the planer, where they wait to be transferred to the Rip station.

In the mill, the transfer of lumber from raw-material store to planer and from planer to the Rip station happens by a forklift. We have not modeled forklifts, but we have taken into consideration the time it takes for the lumber to be transferred and that has been modeled.

Depending on scenario (pull or push), boards are transferred in batch sizes of 50 or 100 from the raw-material store to the Planer and then from the Planer to Rip.

The primary Rip station has its input queue. There is one operator that loads the boards on to the primary Rip machine. Ripping of the boards result in strips.

The Salvage-Rip Station also has an input queue in which strips from the primary Rip station accumulate. This station is run only in the Push scenario. In the Pull system, this station is not run at all to avoid making extra parts. In the Pull system, all material that could be salvaged is treated as scrap (this is done by not reading the Salvage parts file in the AutoMod model).

The strips (from primary or Salvage Rip) are transferred to the Chop station by conveyor. The conveyor is 30 feet long, 4 feet wide and is run at a speed of 1 feet/sec.

Parts that wait to be chopped accumulate in the queue in front of the Chop machine. There is one operator that loads the boards on to the chop machine. The chopped parts are dropped in the output queue. There is one operator for unloading the chopped parts, counting the parts, and sorting them into trolleys based on their dimensions.

When sorting parts, only the parts that are needed for the current day are sent to the molder. This is because molding is a customized step and the mill molds the parts only in the quantity for which it has definite orders (i.e., they are on the cutting bill). In

the Push system, all those parts which are required on the cutting bill but are in excess quantities (orphan parts) are sent to the Extra Inventory store for later (future) use. All those parts that are not on the cutting bill (not-needed parts: have dimensions different from the parts on the cutting bill) are sent to the Not-Needed store where not-needed parts are kept. The not-needed parts are considered to be scrap by the model.

The sorted parts are sent to the molder station where they wait to be worked upon in their respective queues.

When full batch size of a part has accumulated in the part queues, they are sent to the available molder. The part type to be sent to the molder is chosen randomly from among the part types which have full batch size. A changeover is done at the molder when a new batch is sent to it.

If any part is less than the full batch size, but has accumulated the quantity required on the cutting bill, it is sent to the available molder.

The molded components wait in the output queue of Molders to be transferred to the finished goods store, from where they are shipped to the customers.

Work is released at the beginning of each day into the mill, in the form of cutting bill.

In total, there are 9 operators in the mill. One operator loads the boards on the planer and the other unloads it. There is one operator who loads the boards on the Rip machine. Another operator loads the boards on the Salvage-Rip machine. There is one operator who loads the strips on the chop machine. Another operator collects the chopped pieces. There are three operators in the molding station (which has three molders) who load the parts on the molding machines.

The operator at Salvage-Rip machine has two tasks. When there are no parts to be worked upon at the Salvage-Rip machine, he comes to the Chop station to sort and count parts.

The Planer operators also go to the Chop station to sort and count parts, when there are no more parts to be worked upon at the Planer machine.

3.3.2 Model Assumptions

The following assumptions have been made in building up the flow simulation model:

i) Whenever there is material (boards, dimension components) that is in need of being transported between operations, there are means available to transfer the material. Transfer equipment and personnel are not modeled. The transfer is modeled as a time delay.

ii) Each day, the mill has to satisfy the same cutting bill. Each day requires 17,900 parts (a total of 32,033 lineal feet). So the order for 20 days is 17,900 × 20 = 358,000 parts (32,033 × 20 = 640,660 lineal feet).

iii) Even if all the parts required on the cutting bill are not obtained in their required quantities, work is started on the next day's cutting bill,

iv) It is assumed that all the deficiency, if any, in the order is filled on the 21st day. If the orders are not completed in 20 days, the mill will run on the 21st day to complete the work.

3.3.3 Distributions Used in Mill Setting

The set-up of the dimension mill for flow simulation is based on a real dimension-mill, Lewis Lumber Products of Picture Rocks, Pennsylvania. Actual process distributions for this mill were first studied and reported by Laddad (2005). Hypothesized distributions based partially on the Laddad work were tested and subsequently established specifically to meet the objectives of this particular study and are given in Table 3.3. While the actual modeling of the process flow was approached differently from the Laddad work, these process distributions were established to closely represent a simulation that approximated actual performance by a dimension mill with design and management characteristics resembling the Lewis Lumber Mill facility as studied by Laddad. At the same time, the new model formulation and its process distributions also performs satisfactorily under a more variable set of conditions representing a more generic dimension mill simulation model, in order that the evaluation of push versus pull techniques could be conducted under conditions of equal demand fulfillment and that the five hypotheses of this study could be tested and reported.

The operation times for the machines in the mill are given in table 3.4.

Table 3.3 *Distributions used in the mill setting for Flow simulation.*

Sr. #	Description	Distribution	Remarks
1	Transfer time for boards from raw material store to the Planer.	uniform 5, 2 min	
2	Loading time at Planer	u 1.5, 5 sec	u = uniform
3	Unloading time at Planer	u 1.5, 5 sec	
4	Transfer time for Planed boards from Planer to the Rip	u 3, 1 min	
5	Loading time at Rip	u 2,1 sec	
6	Loading time at Chop	weibull 2.5, 3 sec	
7	Unloading time at Chop	weibull 1.5, 2 sec	
8	Transfer time for Chopped parts from Chop to Moulder	e 2 min	e = exponential
9	Loading time on Moulder	weibull 2.5, 2 sec	
10	Changeover time at Moulder	weibull 1.5, 3 min	
11	Transfer time from Moulder to Finished Parts Store	e 1 min	

.

Table 3.4 *Operation times for the machines.*

Machine	Operation time (inches/sec)
Planer	25 (125 lineal feet/minute)
Primary Rip	25
Salvage Rip	33.6 (168 lineal feet/minute)
Chop	u 1.5, 0.5 sec
Molder1 and Molder 2	15 (75 lineal feet/minute)
Molder 3	18 (90 lineal feet/minute)

3.4 Push and Pull: Differences and similarities

There are following differences and similarities between the Push and Pull systems as modeled for this research:

3.4.1 Differences:

Simulation Component	Push System	Pull System
ROMI-3[1]	**1)** Primary operations allow orphan* parts. **2)** Salvage** is not constrained to cutting bill requirements. Both orphans and salvage parts are produced to maximize final product yield. **3)** The cutting bill is adjusted daily according to beginning inventory levels.	**1)** Primary operations avoid orphan parts. **2)** Cutting bill is fulfilled through primary operations. Salvage operations are not used; unutilized wood from primary cutting is hogged for fuel production and reduces final product yield. **3)** No cutting bill adjustment is necessary as there is no carry-over inventory.
AutoMOD	**1)** The transfer batch sizes are twice that of the Pull System. **2)** Both the flow simulation input files (‘.fs1’ & ‘.fs2’) generated by ROMI, are read and used by the AutoMod model. *** **3)** All the excess parts from Extra Inventory store are sent to the Molders to be worked upon.	**1)** The transfer batch sizes are half that of the Push System, to represent the methodology of Pull System theory. **2)** Only the flow simulation input files (‘.fs1’) generated by ROMI are read and used by the AutoMod model. Salvage and Extra parts are treated as scrap. **3)** There are no extra parts to be sent to Molders.

* Orphan part is a primary part that is cut for yield but it is not on the cutting bill.

** Salvage parts are parts that are obtained by at least one additional cutting operation.

*** ‘.fs1’ files contain information about the parts generated by the primary operations. ‘.fs2’ files contain information about the parts generated by the salvage operations.

[1] ROMI-3: Rough-Mill Simulator Version 3.0: User’s Guide

3.4.2 Similarities:

v) Both systems have to satisfy the same customer demand. Each day, the mill has to satisfy the same cutting bill. Each day requires 17,900 parts. So the order for 20 days is 17,900 × 20 = 358,000 parts (32,033 × 20 = 640,660 lineal feet).

vi) Transportation times between same stations are same for both the systems.

vii) Both systems have the same machines, machining rates and machine capacities.

viii) Upstream machines are blocked out and discontinue processing when the queues downstream reach their capacity.

ix) Both systems use the same grade of boards for each corresponding day.

In both systems, on any given day, the molders process only those parts that are required on the cutting bill and in the quantity required on the cutting bill. This is because molding is a customized step and the mill molds the parts only in the quantity for which it has definite orders (i.e., they are on the cutting bill)

3.5 Summary

This chapter discussed the process simulation and the flow simulation models as developed for this research, the differences and similarities in the push and pull scenarios as modeled.

The next chapter discusses the research hypothesis, performance measures, experimentation and results.

Chapter 4

Experimentation and Results

4.1 Introduction

This chapter discusses the analysis and results of the model developed to compare push and pull systems. A brief discussion of the results follows the hypotheses tested using the model.

4.2 Model verification

In order to trust the results obtained from the simulation, it is imperative that it is verified that the model is behaving in the way it is supposed to behave. The model was extensively tested by using the following techniques:

i) Printing messages to the message window while running the simulation. Some conditions and values of variables can be verified:

- Values of the length and width of the board that is read from the '.fs1' file,
- Values of the dimension components,
- Values of various variables and counters,
- The right part is going into the right queue, etc.

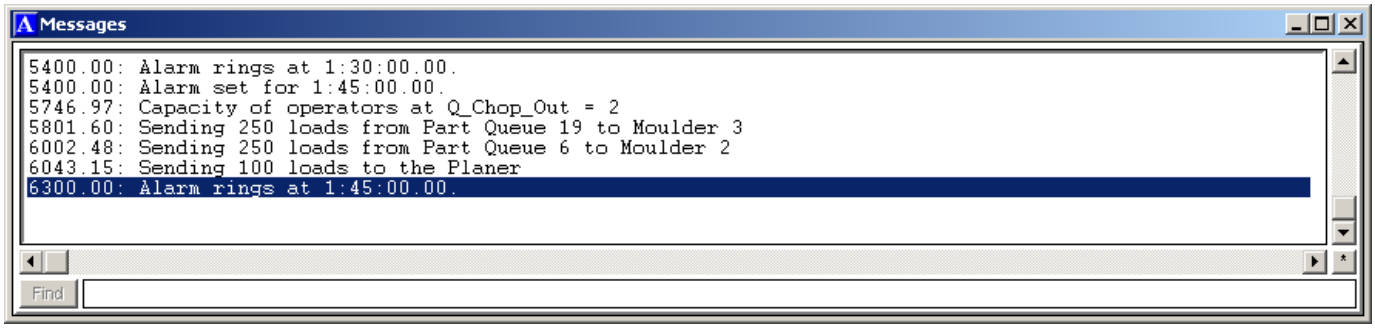

Figure 4.1 *Printing messages in the message window to verify that correct batch size of parts are being transferred.*

ii) Observation of output statistics: The output statistics were observed during the test runs. For example, there are 17,900 parts required on the cutting bill per day. The counter, C_Total_Primary, in figure 4.2 below shows a current value of 17,900 after eight hours. This shows that the mill was able to meet the cutting bill for that day.

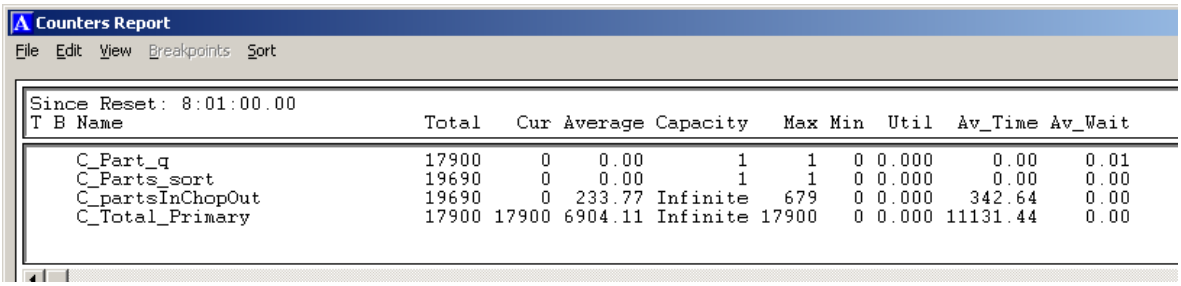

Figure 4.2 *Using reports from test run to observe output statistics.*

4.2.1 Warm up

This model does not require any warmup time. This is because we have modeled this system as a terminating system. The mill processes a fixed number of jobs (20 cutting bills) and then shuts down.

4.3 Changeover frequency at Molders

Table 4.1 shows the calculations used to determine the changeover frequency for the push system. For the Pull system, this frequency was reduced by half (since, in Pull system, all the queues and transfer batch sizes are one half of the Push system).

Table 4.1 *Changeover frequency at molders for Push System.*

Available production time = 8 hours = 480 mins = 28,800 sec

$$Takt\ Time = \frac{\text{Available Production Time}}{\text{Required Daily Production Quantity}} = \frac{28{,}800}{17{,}900} = 1.61\text{sec}$$

Changeover time at molders = weibull 1.5, 3 min

Time required by operator to load = weibull 2.5, 2 sec

Time required by machine = (length of part / 15) sec

Number of molders = 3

Average time required for the operator to load $= \frac{2.5}{3} = 0.83\text{sec}$

Average length of all the pieces = 19.45 inches

Average time needed on molder for the piece to go through $= \frac{19.45}{3 \times 15} = 0.43\text{sec}$

Therefore, average time needed for a piece to go through molder = 0.83 + 0.43 = 1.26 sec

Total time needed for 17,900 pieces $= 17{,}900 \times 1.26 = 22{,}554\text{sec} \approx 376\text{ mins}$

Time remaining for changeovers = 480 – 376 = 104 mins

Average time required for changeover at one moulder = 1.5 mins

Number of changeovers possible $= \frac{104}{1.5} \approx 69$

Allowable changeovers at each molder $= \frac{69}{3} \approx 23$

In other words, a changeover can happen every $\frac{17{,}900}{69} \approx 250$ parts.

4.4 Problem Statement

For a typical hardwood dimension mill, a customer orders dimension parts with widths ranging from 1.00 to 4.75 inches and length ranging from 5 to 85 inches (Araman 1982). The quantities of each part size vary between orders. Prediction of future customer orders is quite imprecise, increasing the difficulty of scheduling production.

These features make the production environment complex and challenge the suitability of the pull system in a dimension mill. As noted in the literature review, many studies have compared the pull and push systems theoretically or descriptively, but no studies on their relative performance have been conducted in the hardwood dimension mill environment.

The objective of this research is to compare, through discrete event simulation, the relative performance of push and pull systems with respect to average inventory levels, throughput of the system, utilization of the machines (Planer, Rip, Chop and Molders) Average lead time of the parts (from the moment boards enter the Raw Material Store to the time finished parts enter the Finished goods store) and physical product yield in a hardwood dimension mill featuring automatically-generated sawing solutions and product generation.

4.5 Research Hypotheses

Pull production system (which is the foundation of Lean Manufacturing system, or JIT, or Toyota Production System) has been widely credited for the success of Toyota Motor Company. The most noted benefits of pull system are (Hopp & Spearman, 2004):

i) Reduced WIP and Lead Time: By limiting the releases into the system, pull system results in a lower average inventory level. By Little's Law, this translates into shorter lead time.

ii) Higher throughput: By dampening fluctuations in the WIP level, pull system achieves a steadier, more predictable output stream. There is less variability in the system which in turn results in higher throughput.

iii) Improved Quality: A system with short queues cannot tolerate high level of defects as it will quickly shut the line down. Additionally, short queues reduce the time between creation and detection of a defect. A Pull system applies pressure for better quality and creates conditions to achieve it.

iv) Lower Utilization: In a pull system, no one upstream produces a good or service until the customer downstream asks for it (Womack & Jones, 1996). This means that machines are not run to maximize their utilization. They are shut down when there is no demand.

v) Reduced Costs: A Pull system "stresses" itself. By reducing WIP, problems in the system (for example, higher defects, longer changeover times, etc.) are exposed. Solving these problems makes the system more efficient and profitable.

This research does not test all the benefits mentioned above. This research has been designed to test whether the pull system as represented in this simulation has:

i) Lower average inventory

ii) Higher throughput

iii) Lower lead time

iv) Lower utilization of machines

We expected the yield to be lower in the Pull system than that of the Push system as we were scrapping the extra and unneeded parts (salvage, orphans, and not-needed parts) in the Pull system.

Following are the hypotheses that are tested through this research:

1. Lower inventory level than that of push system for a typical hardwood dimension mill, i.e.

$H_0 : Inv_{pull} = Inv_{push}$; $\quad H_\alpha : Inv_{pull} < Inv_{push}$

where : Inv_{push} is the average production inventory level in push system;
Inv_{pull} is the average production inventory level in pull system.

2. Higher throughput than that of push system for a typical hardwood dimension mill, i.e.

$H_0 : T_{pull} = T_{push}$; $\quad H_\alpha : T_{push} < T_{pull}$

where : T_{push} is the average throughput of push system;
T_{pull} is the average throughput of pull system.

3. Lower final product yield than that of push system for a typical hardwood dimension mills, i.e.

$H_0 : Y_{pull} = Y_{push}$; $\quad H_\alpha : Y_{pull} < Y_{push}$

where : Y_{push} is the average final product yield in push system;
Y_{pull} is the average final product yield in pull system.

4. Lower lead time that of push system for a typical hardwood dimension mill, i.e.

$H_0 : t_{pull} = t_{push}$; $\quad H_\alpha : t_{pull} < t_{push}$

$where : t_{push}$ is the average lead time of a component in push system;
t_{pull} is the average lead time of a component in pull system.

5. Lower utilization of machines than that of push system for a typical hardwood dimension mills, i.e.

$H_0 : U_{pull} = U_{push}$; $\quad H_\alpha : U_{pull} < U_{push}$

$where : U_{push}$ is the average utilization of machines in push system;
U_{pull} is the average utilization of machines yield in pull system.

4.6 Results

This section discusses the results obtained by running the simulation model and their analysis. Five runs of each system were made. The sources of variability in the runs are listed in table 3.3 (page 36). The analysis of the results was done using the statistical software Minitab-15. The assumption of equal variance has been made for the t-tests, as the sample sizes (5 for each) are equal for both scenarios (Banks et al, p437).

4.6.1 Inventory

Inventory in the system as measured in this simulation is the combined inventory at the raw material store, planer, rip (primary & salvage), chop and molder stations.

Figure 4.3 illustrates the inventory results for each of the push and pull simulations.

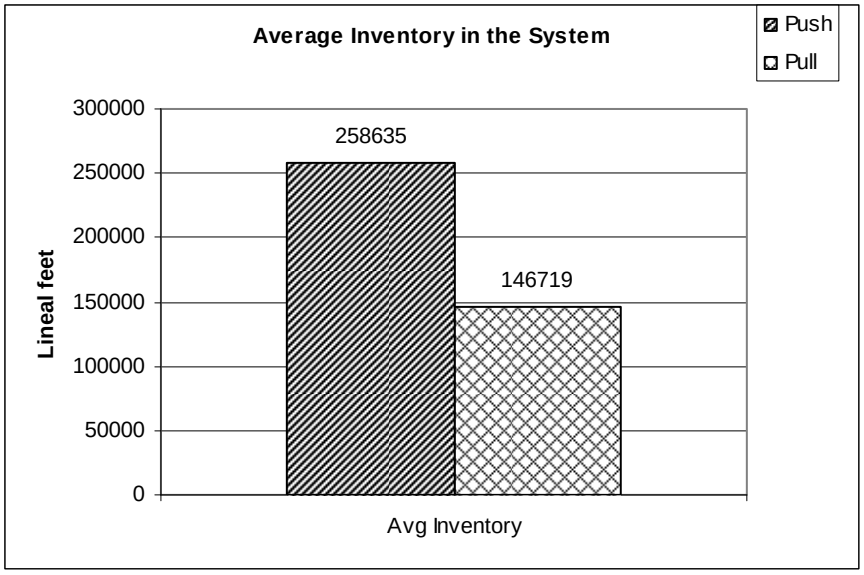

Figure 4.3 *Average Inventory (lineal feet) in the system.*

Table 4.2 shows the results of Two-Sample T-Test and Confidence Interval (CI) for the Average Inventory in the Push and Pull Systems.

Table 4.2 *Two-Sample T-Test (assuming equal variance) and Confidence Interval for Average Inventory in the Push and Pull Systems.*

```
Two-sample T for Avg_Inventory_inSystem_Pull vs Avg_Inventory_inSystem_Push

                          N      Mean  StDev  SE Mean
Avg_Inventory_inSystem_P  5  146719.0   13.8      6.2
Avg_Inventory_inSystem_P  5  258635.3   51.4       23


Difference = mu (Avg_Inventory_inSystem_Pull) – mu (Avg_Inventory_inSystem_Push)
Estimate for difference:  -111916
95% upper bound for difference:  -111872

T-Test of difference = 0 (vs <): T-Value = -4698.55  P-Value = 0.000  DF = 8
Both use Pooled StDev = 37.6617
```

As the p-value is 0, we reject the null hypothesis and can say that the inventory level in the Pull system is lower than that of the push system for a typical hardwood dimension

mill. Also, as can be seen from the 95% CI for difference, on average, inventory in the Push System is 111,916 board feet higher than that of the Pull System.

From figure 4.3, it can be seen that the inventory in the Push system is almost 50% higher than that of the Pull System.

4.6.2 Throughput

The targeted throughput of the simulations was 32,033 lineal feet (17,900 parts) per day, as called for by the cutting bill described in Table 3.1 (Chapter 3). Figure 4.4 illustrates the average throughput for each of the push and pull simulations.

Figure 4.4 illustrates throughput metrics for three categories of products; those not required by the customer order (Not Needed); those required but produced in excess (Extra or orphans); and those required and produced to schedule (Finished, required pieces). The push system produced more of all three types of products. In the push system, the orphans are sent to the molders at the beginning of the next day, while in the pull system they are scrapped on the next day. The figure of 2906 and 13 lineal feet are just what is left after the last (20^{th})day.

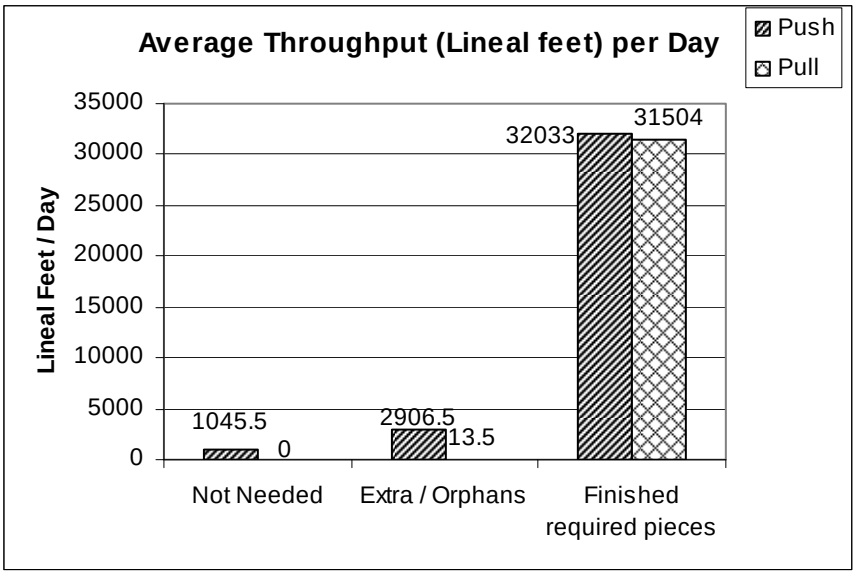

Figure 4.4 *Average Throughput (lineal feet) of the systems.*

Table 4.3 shows the results of Two-Sample T-Test and Confidence Interval (CI) for the Throughput (Finished required pieces only) in the Push and Pull Systems.

Table 4.3 *Two-Sample T-Test (assuming equal variance) and Confidence Interval for Throughput Finished, required pieces) in the Push and Pull Systems.*

```
Two-sample T for ThroughputPerDay_Push vs ThroughputPerDay_Pull

                        N       Mean  StDev  SE Mean
ThroughputPerDay_Push   5  32032.878  0.500     0.22
ThroughputPerDay_Pull   5  31504.684  0.409     0.18


Difference = mu (ThroughputPerDay_Push) - mu (ThroughputPerDay_Pull)
Estimate for difference:  528.194
95% upper bound for difference:  528.731

T-Test of difference = 0 (vs <): T-Value = 1827.42  P-Value = 1.000  DF = 8
Both use Pooled StDev = 0.4570
```

As the p-value is 1, we fail to reject the null hypothesis and can say that the throughput of the Pull System is lower than that of the Push System.

From 95% CI for difference, we can see that, on average, throughput of finished needed parts in the Pull system is 528 lineal feet higher than that of Pull system.

Fine tuning of the batch (kanban) sizes in a well-run lean production system could be expected to improve the pull system's performance to schedule, at least as far as the constraints of the physical system will allow. Here the advantages of having a simulation tool become apparent; mill management, the process researcher, or the kaizen team could use this type of tool to fine tune those lean production parameters without the usual trial and error associated with kaizen events and kanban trials.

4.6.3 Yield

Yield is measured as a function of wood product out to wood raw material in.

Figure 4.5 is an illustration of the yield performance metrics of the two systems as modeled in this effort. In this specific comparison, the Push system outperformed the Pull system in both yield metrics.

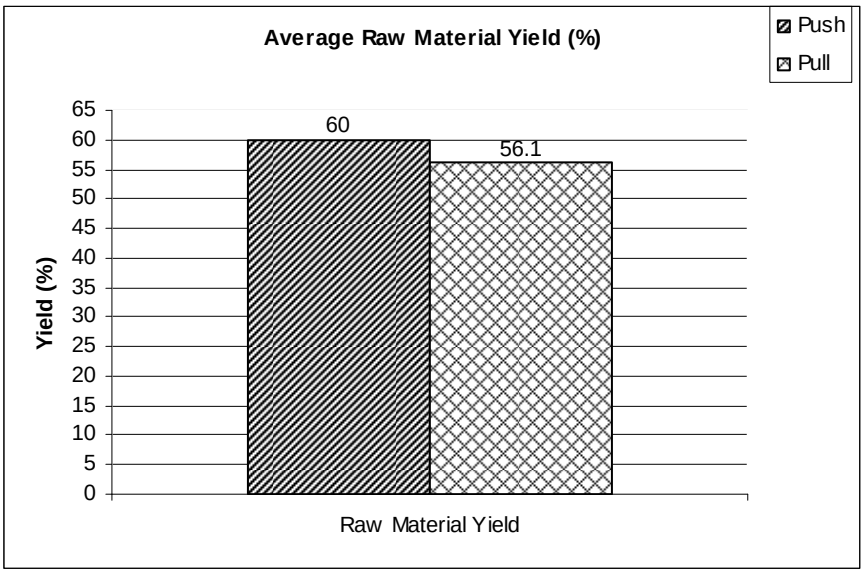

Figure 4.5 *Average Yield (percent) of the systems.*

Table 4.4 shows the results of Two-Sample T-Test and Confidence Interval (CI) for the Yield (in percent) in the Push and Pull Systems.

Table 4.4 *Two-Sample T-Test (assuming equal variance) and Confidence Interval for Yield (%) in the Push and Pull Systems.*

```
Two-sample T for Yield_Pull vs Yield_Push

             N    Mean  StDev  SE Mean
Yield_Pull  20  56.130  0.875     0.20
Yield_Push  20  60.005  0.761     0.17

Difference = mu (Yield_Pull) - mu (Yield_Push)
Estimate for difference:  -3.875
95% upper bound for difference:  -3.438

T-Test of difference = 0 (vs <): T-Value = -14.94  P-Value = 0.000  DF = 38
Both use Pooled StDev = 0.8200
```

As the p-value is 0, we reject the null hypothesis and can say that the yield of the Pull System is lower than that of the Push System.

From 95% CI for difference, we can see that, on average, yield in the Pull system is 3.9% lower than that of Push system. Though this model reflects yield loss, it does not reflect the extra time and effort required to track the excess inventory and feed it back into the production process. The higher yield may be partially offset by the increased management costs.

4.6.3 Lead Time

Lead time as measured in the system is the time boards are received in the raw material store to the time finished parts enter the finished goods store.

Figure 4.6 illustrates the average time in the system for each of the push and pull simulations.

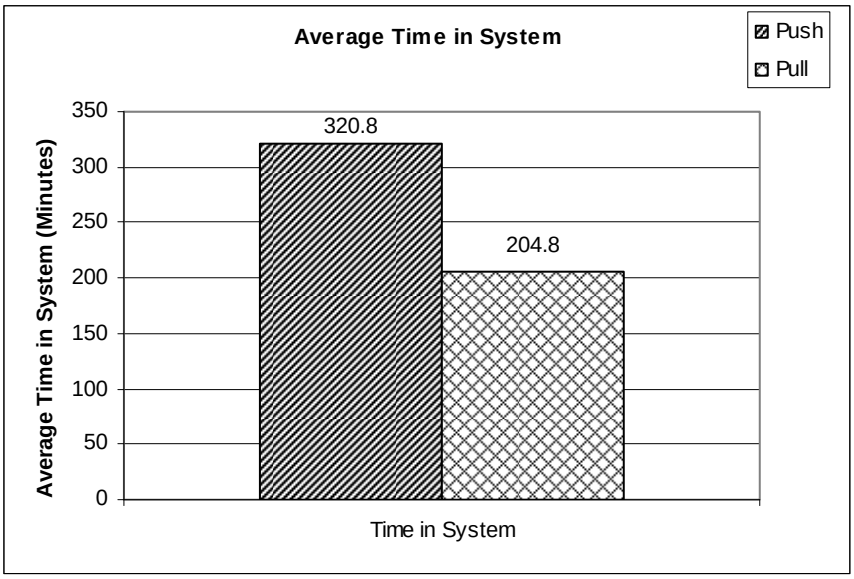

Figure 4.6 *Average Lead Time (minutes) in the system.*

Table 4.5 shows the results of Two-Sample T-Test and Confidence Interval (CI) for the Average Lead Time in the Push and Pull Systems.

Table 4.5 *Two-Sample T-Test (assuming equal variance) and Confidence Interval for Average Lead time in thePush and Pull Systems.*

```
Two-sample T for LeadTime_Pull vs LeadTime_Push

               N     Mean  StDev  SE Mean
LeadTime_Pull  5  204.889  0.248     0.11
LeadTime_Push  5  320.779  0.862     0.39


Difference = mu (LeadTime_Pull) - mu (LeadTime_Push)
Estimate for difference:  -115.890
95% upper bound for difference:  -115.144

T-Test of difference = 0 (vs <): T-Value = -288.78  P-Value = 0.000  DF = 8
Both use Pooled StDev = 0.6345
```

As the p-value is 0, we reject the null hypothesis and can say that the lead time in the Pull system is lower than that of the push system for a typical hardwood dimension

mill. Also, as can be seen from the 95% CI for difference, on average, lead time in the Push System is 115.8 minutes higher than that of the Pull System.

The higher lead time in the push system can be attributed to the higher inventories in the push system.

4.6.5 Utilization

Utilization of machine centers is typically regarded as a key mill performance metric, since unused capacity implies excess cost (Hopp and Spearman 2000, p.292). In many production cost accounting systems, cost is allocated to products according to the "operating cost" of each machine center times the amount of time each product uses that machine. The operating cost is usually calculated as a function of the invested cost of the machine and the amount of product produced from that machine…hence, the "utilization" of the machine as measured by production per unit of time is a key metric of how relatively expensive a certain machine may be. The traditional operating assumption in most push systems is that by having maintaining high utilization rates despite fluctuating demand, the operating cost of each machine is lowered and therefore products can be produced at a more competitive cost.

The Planer and the Rip Machine have to process the same length of boards in both the push and pull systems. The molders too process almost the same length (lineal feet) of material in both the systems. The difference lies in the utilization of the Salvage Rip and the Chop machine. Since the pull systems does not process any salvage parts, the Salvage Rip machine is not used and hence its utilization is 0.

Figure 4.7 illustrates the average utilization of machines for each of the push and pull simulations. As can be seen from the figure, the utilization of all the machines (except the Chop and Salvage Rip) is almost the same in both the systems. This is a somewhat surprising finding since it had been assumed that traditional assumptions about push production scheduling would hold true and result in higher machine utilization at several machine centers. That this was found to be so at only one primary machine in this mill calls into question the alleged advantage of "lower cost per unit" for push production systems in dimension mill operation.

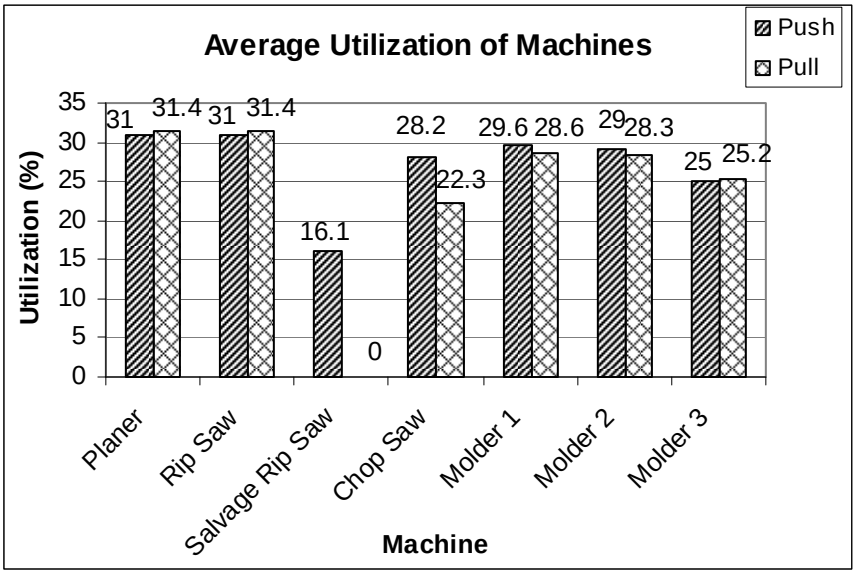

Figure 4.7 *Utilization (%) of machines.*

Table 4.6 shows the results of Two-Sample T-Test and Confidence Interval (CI) for the utilization of the chop machine in the Push and Pull Systems.

Table 4.6 *Two-Sample T-Test (assuming equal variance) and Confidence Interval for utilization of chop machine in the Push and Pull Systems.*

```
Two-sample T for Util_Chop_Pull vs Util_Chop_Push

                N    Mean  StDev  SE Mean
Util_Chop_Pull  5  22.180  0.164    0.073
Util_Chop_Push  5  28.120  0.110    0.049


Difference = mu (Util_Chop_Pull) - mu (Util_Chop_Push)
Estimate for difference:  -5.9400
95% upper bound for difference:  -5.7758

T-Test of difference = 0 (vs <): T-Value = -67.26  P-Value = 0.000  DF = 8
Both use Pooled StDev = 0.1396
```

As the p-value is 0, we reject the null hypothesis and can say that the utilization of chop machine in the Pull system is lower than that of the push system. Also, as can be seen from the 95% CI for difference, on average, the utilization of chop machine in the Pull System is around 6% lower than that of the Push System.

The lower utilization of chop machine in the pull system can be attributed to the fact that in the pull system, the chop machine does not have to process the parts from the salvage rip. That this utilization rate is shown to be significantly lower in the pull scenario of this mill has serious implications for mill management at Lewis Lumber. The company accountant had identified the chop saw as the most expensive "cost center" in the mill, and jobs were being bid and produced based at least partially on the minimization of machine time on the chop saw. Since this study shows "pull" production scheduling to effect in essence an increase in chop saw operational cost through decreased utilization of the chop saw, it may be one of the key factors preventing mill management from implementing pull in their mill.

4.7 Discussion

The early focus of this research was centered on the definition of the two contrasting systems, push and pull. That is, modeling decisions for both the ROMI and AutoMod components of the tool were put forward, tested, re-considered, and re-formulated in an iterative fashion until the models seemed to be behaving consistently with expectations and experiences of the research team in actual operations.

Once that behavior was established, the modeling effort turned to actual measurement of the performance of the two systems. The original processing parameters, for example, machine processing rates, downtimes, batch sizes, queue sizes, etc., and specified distributions of each were originally set with initial objectives to simply establish "balanced" models; that is, they were set to levels that allowed completion or near-completion of the specified cutting bills on a daily production basis

We believe that these results are specific to the systems modeled, and that the push-pull comparison results could differ for different mill configurations and definitions of push and pull. The performance of either push or pull systems is entirely dependant on how well-designed the system is relative to the daily customer demand.

In this study, customer demand as represented by the "easy" cutting bill was simplified so that each system demanded the same components daily. The dynamic complexity of real demands on a dimension mill means that this "process optimization" is an exhaustive trial-and-error process, one that could benefit from computerized simulation tools as demonstrated here.

For this specific mill "trial" of push vs. pull processing, hypotheses 1, that the pull system results in a lower level of in-process inventory, hypothesis 3, that the pull system

results in lower process yields, and hypothesis 4, that the pull system results in lower lead time, were all found to be true.

Hypothesis 1. The pull system lower in-process inventory partially resulted from limiting the queue capacities to half of what they were in the push system. This limitation was modeled on the basis of its similarity to implementing kanban inventory control as is typically done in lean production efforts targeted at achieving pull production scheduling, and its impact in reducing in-process inventories in the simulation is what would be expected through kanban implementation in another type of operation. In effect, the limiting of component cart sizes and numbers is one possible definition of kanban implementation in a hardwood dimension mill.

Another reason for lower inventory in the pull system was the scrapping of salvage and orphan parts counter to traditional dimension mill practice of maximizing yield from each piece of lumber. This logic simulated the pull system culture of focusing only on required parts and not producing parts not needed in current production schedules as represented by the cutting bills in a typical hardwood dimension mill. The result, as expected, was lower work-in-process for the simulation.

Hypothesis 4. Having lower WIP in the pull system resulted in lower lead times as the aggregated components had to wait shorter periods to reach the transfer batch size. As a result, the average length of time in the system of components in the pull system was shorter, because those components in the early part of each daily cutting bill could flow through the mill to finished inventory while the components in the push system were awaiting their defined larger batch sizes to move forward. However, these shorter lead times cumulatively did not result in cutting bills that could be completed in a shorter time

(as indicated by throughput results) because the lower yield per piece of lumber was the ultimate determinant of when the final pieces of each cutting bill were produced. This reliance on the natural variability of lumber to produce all the components of the cutting bill in the required quantities is the primary difference between wood-based manufacturing operations and those that rely on a homogeneous feedstock. Said another way, shorter average lead times of components in a dimension mill operation is of little practical benefit if the nearly-complete order is detained by a few specific pieces not able to be produced from the variable lumber feedstock.

Hypothesis 5. This phenomenon was most clearly indicated in lower utilization of the chop machine in the pull system. As hypothesized, the utilization rate of the chop machine was significantly reduced; however, the utilization rate of the other primary processing machines (planer, rip saw, and moulders) was not. The chop machine should be the operational bottleneck of any dimension mill that has multiple moulders to meet production requirements; and since it was shown to have significantly lower utilization in pull production scheduling than push production scheduling, no benefit in the form of total throughput can be expected since the bottleneck is producing less.

Hypothesis 3. As shown, scrapping the salvage and orphans in the pull system to decrease in-process inventory in agreement with pull principles decreased the process yield (wood products out divided by raw material in). In effect, we were able to demonstrate that simply reducing in-process inventory (hypothesis 1) is not an effective "lean" practice for hardwood dimension mill operation, as the benefit of shorter lead times (hypothesis 4) is offset by the penalty of lower utilization rate at the chop saw (hypothesis 5) as dictated by the slower production of required parts from the ripping

operations (hypothesis 3, essentially) resulting in the process not being able to increase throughput (hypothesis 2) as typically expected through conversion to pull production scheduling.

Hypothesis 2. Since the pull system did not result in a higher process throughput, this hypothesis proved false. The data of this specific simulated comparison resulted in almost the same amount of finished products in push and pull systems. In fact, if salvage and orphan parts are considered, the output of the push system was higher than that of the pull system.

This final conclusion is especially important when considered in the light of raw material cost. Wood costs comprise 40-70% of hardwood dimension mill operational costs, so any expected benefit of implementing a pull system in such a mill must offset the additional burden of even higher cost as incurred through the decreased yield of the operation. Since no significant increase in throughput was achieved, no justification for increasing the cost burden through forced yield reduction in order to achieve lower WIP and shorter lead times can be supported.

Chapter 5

Conclusions

5.1 Conclusions and Future Work

A general-purpose simulation modeling methodology was developed to enable the research team to test various hypotheses on the difference of push and pull operational configurations for a specific hardwood dimension mill.

For the mill and specific push and pull systems simulated, the push system demonstrated higher inventory accumulations, throughput, and yield metrics. However, the lower inventory accumulations resulting from the specified pull configuration did not benefit the system enough to attain a expected level of higher throughput. Because of the limitations of this study, this behavior cannot be stated as general performance results for push and pull systems in dimension mills for all situations. However, it suggests that system design relative to lumber inputs and dynamic customer demands has bearing on the success of any lean production implementation in the dimension mill, and that success is not guaranteed simply by adopting some certain facets of lean production techniques. More importantly for the dimension mill industry, it suggests that lean production improvement of a dimension mill is no simple matter to be achieved through cookie-cutter application of lean principles without a fuller understanding of the dynamic constraints that the variability of wood imposes on a manufacturing system. Both process and flow simulation tools of the type used in this study have been shown useful and complementary in specifying lean parameters for successful pull implementation, to test those parameters in specific mill configurations under varying conditions of market

demand, and to account for the wild card of wood raw material variability and its impact on implementation results.

Future research of this type must examine additional scenarios under variable customer requirements, variable lumber inputs, and different lean production designs. General conclusions on relative the performance of push vs. pull systems in the dimension mill may be arrived at only after extensive testing of alternative scenarios and conditions. To be specific, following are some of the areas for future research:

1) **Run the model with different cutting bills:**

   We ran our model with just one cutting bill *(adapted from Steele et al. 1999).* Each day, the mill had to satisfy the same cutting bill. The impact of having different cutting bills on different days and/or having to satisfy a mix of different cutting bills on a given day could be investigated.

2) **Replicate with different grades of lumber:**

   Our simulation used only one grade of lumber (2AC). It would be of interest to study the impact of different grades of lumber on the performance measures. It would also be interesting to investigate the effect of using a mix of different grades of lumber on the performance measures.

3) **Continue to test different lean parameters and definitions of push, pull, and combination systems:**

   The specification of push and pull systems as used in this study are but a first effort at defining these systems for dimension mill operations. The parameters were constrained in the process simulation by the capabilities

of the ROMI software used to produce the components. Future specification of these systems should borrow more from actual lean implementation efforts as they are experienced in the industry and modeled in the software. Further, the recognition of a push-pull interface must be established and modeled as it actually exists under different management scenarios so that future operations researchers can help mill industry management determine optimal lean implementations for their situation.

4) **Look at the associated costs:**

We looked at the production potential of the two systems. There are some metrics on which one systems performs well over the other and some on which it does not. For example, the Pull system has lower inventory (desirable) in the system that that of the Push system, but it also has lower yield (not desirable) than that of Push system. So what's the overall cost cutting potential of these two systems? We did not address this issue in this research. It would be desirable to develop a cost matrix to analyze the overall benefit of a system over the other.

References

Allen, J., C. Robinson, and D. Stewart. 2001. Lean manufacturing: a plant floor guide. Society of Manufacturing Engineers (SME). Dearborn, Michigan. 495p.

Anderson, R.B. 1983. Furniture rough mill costs evaluated by computer simulation. Res. Pap. NE-518. USDA Forest Service, NE Forest Experiment Station. 11 p.

Anderson, R.B. 1985. Programs for computer simulation of a crosscut-first furniture rough mill. Gen. Tech. Rep. NE-97. USDA Forest Service, NE Forest Experiment Station. 11 p.

Araman, P.A. 1977. Use of computer simulation in designing and evaluating a rough mill for furniture interior parts. USDA Forest Service Research Paper NE-361. 9 p.

Banks, J., J.S. Carson II, B.L. Nelson and D.M. Nicole. 2001. Discrete-event simulation. Prentice-Hall, Inc. Upper Saddle River, New Jersey. 437p, 594 p.

Banks, J. 2004. Getting Started with AutoMod. Second Edition. Brooks Automation, Inc., Chelmsford, MA.

Buehlmann, U. 1998. Understanding the relationship of lumber yield and cutting bill requirements: a statistical approach. Ph.D. diss. Virginia Polytechnic Institute and State University. Blacksburg, VA. 207 p.

Brunner, C.C., M.S. White, F.M. Lamb, and J.G. Schroeder. 1989. CORY - a computer-program for determining dimension stock yields. Forest Products Journal 39(2):23-24.

Chen, J., C. Lee, and T. Fujimoto. 1997. Adaptation of Lean Production in China: The Impact of Japanese Management Practice. MIT IMVP Working Paper. 30 p.

Gatchell, C.J., J.K. Wiedenbeck, and E.S. Walker. 1993. A red oak data-bank for computer-simulations of secondary processing. Forest Products Journal 43(6):38-42.

Gazo, R., and P.H. Steele. 1995. Rough mill analysis model. Forest Products Journal 45(4):51-53.

Hopp, W., and M. Spearman. 2000. Factory physics: foundations of manufacturing management. Second Edition. Irwin/McGraw-Hill. New York, NY. 698 p.

Hopp, W., and M. Spearman. 2004. To pull or not to pull: what is the question? Manufacturing & Service Operations Management 6(2): 133-148.

Krishnamurthy, A, R. Suri and M. K. Vernon. 2000. Push can perform better than pull for flexible manufacturing systems with multiple products. Proc. Industrial Engineering Research Conf., Cleveland, OH.

Kimura, O., and H. Terasa. 1981. Design and analysis of pull system: a model of multi-stage production control. Int. J. Prod. Res. 19 (3):241-253.

Kline, D.E., J.K. Wiedenbeck, and P.A. Araman. 1992. Management of wood products manufacturing using simulation animation. Forest Products Journal 42(2):45-52.

Kobayashi, H. 1978. Modeling and analysis: An introduction to system performance evaluation methodology. Addison-Wesley, Reading, MA.

Krajewski, L.J., B.E. King, L.P. Ritzman, and D.S. Wong. 1987. KANBAN, MRP, and shaping the manufacturing environment. Management Science 33(1):39-57.

Kuehl, R. O. 2000. Design of Experiments: Statistical principles of research design and analysis, Second Edition. Duxbury Press, Pacific Grove, CA.

Laddad, A. 2005. Correlation between order variability and product flow effectiveness for a hardwood dimension mill using discrete event simulation. M.S. Thesis, the Pennsylvania State University, College of Engineering, University Park, PA, 103 p.

Lamb, F.M. 2002. Eight factors that affect rough mill yield. Modern Woodworking. http://www.modernwoodworking.com/02issues/April/news/Drying.shtml , last accessed: June 2007.

Levinson, W.A., and R.A. Rerick. 2002. Lean enterprise: a synergistic approach to minimizing waste. ASQ Quality Press. Milwaukee, WI. 235 p.

Lin, W.J., D.E. Kline, and P.A. Araman. 1994. Dimension yields from factory grade-2 and grade-3 red-oak logs. Forest Products Journal 44(9):19-25.

Lin, W.J., D.E. Kline, P.A. Araman, and J.K. Wiedenbeck. 1995. Design and evaluation of log-to-dimension manufacturing systems using system simulation. Forest Products Journal 45(3):37-44.

Monden, Y. 1983. Toyota production system: practical approach to management. Industrial Engineering and Management Press, Norcross, GA. 247p.

NHLA. 1990. Rules for the measurement and inspection of hardwood and cypress lumber. National Hardwood Lumber Association. Memphis, Tenn. 108 p.

Pritsker, A.A.B. 1986. Introduction to simulation and SLAM II. A Halstead Press Book, N.Y. 612 p.

Ryan, B., B. Joiner and J. Cryer. 2005. Minitab Handbook, Fifth Edition. Thomson Brooks/Cole, Belmont, CA. 505 p.

Steele, P.H., and J.E. Aguirre. 2004. The influence of batch versus segmented processing on rough mill yields. Forest Products Journal 54(1):40-46.

Steele, P.H., and R. Gazo. 1995. A procedure for determining the benefits of sorting lumber by grade prior to rough mill processing. Forest Products Journal 45(6):69-73.

Steele, P.H., J. Wiedenbeck, R. Shmulsky, and A. Perera. 1999. The influence of lumber grade on machine productivity in the rough mill. Forest Products Journal 49(9):48-54.

Suri, R. 1998. Quick response manufacturing: a company-wide approach to lead time reduction. Productivity Press, Portland, OR.

Thomas, R.E. 1996. ROMI-CROSS: Rough mill crosscut-first simulator. Res. Paper NE-229. USDA Forest Service, NE Research Station, Radnor, PA. 83 p.

Thomas, R.E. 1998. A guide for using ROMI 2.00, a rough mill rip-first simulator. Gen. Tech. Rep. NE-259. USDA Forest Service, NE Res. Station, Radnor, PA. 64 p.

Thomas, R.E., and J. Brown. 2003. Determining the impact of sorting capacity on rip-first rough mill yield. Forest Products J. 53(7/8):54-60.

Thomas, R.E., and U. Buehlmann. 2002. Validation of the ROMI rough mill simulator. Forest Products J. 52(2):23-29.

Thomas, R.E., and U. Buehlmann. 2003. Performance review of the ROMI rough mill simulator. Forest Products J. 53(3):80-85.

Weiss, J.M., and R.E. Thomas. 2005. ROMI-3: Rough mill simulator 3.0 user's guide. Gen. Tech. Report NE - 328. USDA Forest Service, NE Research Station, Radnor, PA. 81 p.

Wiedenbeck, J.K. 1992. Simulation for rough mill options. Wood &Wood Products. 97(12):67-72.

Wiedenbeck, J.K. and D.E. Kline. 1994. Systems simulation modeling: a case study analysis of the model development life cycle. Wood and Fiber Sci. 26(2):192-204.

Womack, J.P., D.T. Jones., D. Ross. 1996. Lean thinking: banish waste and create wealth in your corporation. Simon & Schuster, New York.

Appendix A: Model Code (for Push)

```
/*This model runs 2AC lumber files as are in the folder "Easy2xPush_ver2" */

begin model initialization function

        create 1 load of type L_Initial to P_Initial
        open "dir/CB_Easy2x.txt" for reading save result as V_CuttingBill /* Cutting bill file*/
        create 1 load of type L_Read_CB to P_Read_Cutting_Bill
         /*This load (created above) will read the cutting bill and store the values as variables*/

        /*open"dir/Easy2AC1_2x.fs1" for reading save result as V_PrimaryPtr
        create 1 load of type L_read to P_read
        open "dir/Easy2AC1_2x.fs2" for reading save result as V_SalvagePtr
        create 1 load of type L_Read_S to P_read_S*/

        create 1 load of type L_RM_trans to P_Rawmaterial_transfer
        create 1 load of type L_Planer_trans to P_Planer_transfer
        create 1 load of type L_Mould_Trans to P_Moulder_transfer
        create 1 load of type L_Mould_out_Trans to P_Moulder_out_Transfer

        create 1 load of type L_Chop_Q_check to P_Chop_Q_check
/*      create 1 load of type L_Q_Chop_out_check to P_Q_Chop_out_check                */
        create 1 load of type L_to_Moulder to P_to_Moulder

        /*The following are dummy loads for creating downtimes*/
        create 1 load of type L_Op_down to P_Operator_Down

        /* for pull, these numbers will be 50, 200 and 125 respectively */
        set V_batchSizeofBoards to 100
        set V_batchSizeofComponents to 400
        set V_batchSizeofComponentsAtMoulder to 250


        /* to tabulate values in tables */
        create 1 load of type L_Tabulate to P_Tabulate

        return true
end

begin P_Read_Cutting_Bill arriving      /* This process reads the Cutting Bill file*/
        set V_P to 0
                while V_CuttingBill eof = false do begin            /*Point A */
                inc V_P by 1
                read V_CB_PartWidth(V_P), V_CB_PartLength(V_P),
V_CB_PartQuantity(V_P) from V_CuttingBill

/* These variables are array of 25
V_CB_PartWidth(V_P) = Part Width as given in the cutting bill, in inches
```

```
V_CB_PartLength(V_P) = Part Length as given in the cutting bill, in inches
V_CB_PartQuantity(V_P) = The Quantity of a particular part required as given in the cutting
bill*/

        /*print "The Part Width Required of the", V_P "part is",  V_CB_PartWidth(V_P) to
message
        print "The Part Length Required of the", V_P "part is",  V_CB_PartLength(V_P) to
message
        print "The Part Quantity Required of the", V_P "part is",  V_CB_PartQuantity(V_P) to
message*/
                end                                     /* End of while of Point A*/

        send to die
end

/*The following process generates load for board supply. */
begin P_Initial arriving
  set V_Day to 1
  while V_Day <= 20 do begin
    if V_Day = 1 then
      begin
          take down R_Op(9)
          take down R_Op(10)
          take down R_Op(11)
          set C_Op capacity = 2
          open "dir/Easy2xDay1_Push.fs1" for reading save result as V_PrimaryPtr
          open "dir/Easy2xDay1_Push.fs2" for reading save result as V_SalvagePtr
          create 1 load of type L_read to P_read
      end
    else if V_Day = 2 then
      begin
        take down R_Op(9)
          take down R_Op(10)
          take down R_Op(11)
          set C_Op capacity = 2
       bring up R_Op(1)
       bring up R_Op(2)
          open "dir/Easy2xDay2_Push.fs1" for reading save result as V_PrimaryPtr
          open "dir/Easy2xDay2_Push.fs2" for reading save result as V_SalvagePtr
          tabulate Q_Final_Parts current in T_finalPartsPerDay
          order all loads from OL_Finish to continue
          create 1 load of type L_read to P_read
          wait for 0.01 sec
          tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
          tabulate Q_not_needed current in T_notNeededPerDay
          order all loads from OL_Not_Needed to continue
          order all loads from OL_Next_Day to continue
      end
    else if V_Day = 3 then
      begin
          take down R_Op(9)
```

```
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay3_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay3_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 4 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay4_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay4_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 5 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay5_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay5_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
```

```
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 6 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay6_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay6_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 7 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay7_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay7_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 8 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay8_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay8_Push.fs2" for reading save result as V_SalvagePtr
```

```
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 9 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay9_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay9_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 10 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay10_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay10_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 11 then
  begin
        take down R_Op(9)
        take down R_Op(10)
```

```
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay11_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay11_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 12 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay12_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay12_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 13 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay13_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay13_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
```

```
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 14 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay14_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay14_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 15 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay15_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay15_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 16 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay16_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay16_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
```

```
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 17 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay17_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay17_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 18 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay18_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay18_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
else if V_Day = 19 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
```

```
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay19_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay19_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
 else if V_Day = 20 then
  begin
        take down R_Op(9)
        take down R_Op(10)
        take down R_Op(11)
        set C_Op capacity = 2
    bring up R_Op(1)
    bring up R_Op(2)
        open "dir/Easy2xDay20_Push.fs1" for reading save result as V_PrimaryPtr
        open "dir/Easy2xDay20_Push.fs2" for reading save result as V_SalvagePtr
        tabulate Q_Final_Parts current in T_finalPartsPerDay
        order all loads from OL_Finish to continue
        create 1 load of type L_read to P_read
        wait for 0.01 sec
        tabulate Q_Extra_Inv current in T_extraInvtoryPerDay
        tabulate Q_not_needed current in T_notNeededPerDay
        order all loads from OL_Not_Needed to continue
        order all loads from OL_Next_Day to continue
  end
 create 1 load of type L_Read_S to P_read_S

 wait for 482 min                /* 8 hours and 2 minutes */
 /* At the end of the day, record these three in the table */
 tabulate (V_Bfeet_FinalNeeded/12) in T_Bfeet_FinalNeeded
        tabulate (V_Lineal_foot/12) in T_Bfeet_notNeeded
        tabulate (V_Bfeet_Extra_Inv/12) in T_Bfeet_Extra_Inv

 inc V_Day by 1

  set V_Initial to 1
  while V_Initial <= V_P do begin
                set C_Final_Quantity(V_Initial) to 0
                set C_Final_Quantity_Obtained(V_Initial) to 0
                inc V_Initial by 1
         end

end             /*End of main while*/
```

```
 send to die
end

begin P_read arriving
/* This process reads the part-widths and part-lengths from the .fs1 file*/
 while V_PrimaryPtr eof = false do begin          /* While there is unread data*/          /*Point
r_1*/

          read LA_Width, LA_Length, LA_Pieces from V_PrimaryPtr
          /*print "The Board Width is",  LA_Width to message
          print "The Board Length is",  LA_Length to message
          print "The number of strips you will get is",  LA_Pieces to message*/

          if LA_Width = 10001 and LA_Length = 10001 and LA_Pieces = 1 then
           begin
            clone 1 load to P_Rawmaterial_wait nlt L_Dummy
            send to die
           end

          else

           set V_i to 1
                  while V_i <= LA_Pieces do       /*Point r_2*/
              begin
                           read LA_Stripwidth(V_i), LA_Chopparts(V_i) from V_PrimaryPtr
                           set LA_Stripwidth(V_i) to (LA_Stripwidth(V_i)/16)

                           /*print "The", V_i " Strip Width is" LA_Stripwidth(V_i) to message
                           print "The Chop parts you get form the", V_i "Strip is ="
LA_Chopparts(V_i) to message*/

                           inc V_i by 1
                    end                                                    /* End of while of Point
r_2*/

          set V_Chop_Parts to 1
          set V_i to 1
                  while V_i <= LA_Pieces do                    /*Point r_3*/
              begin
                           while V_Chop_Parts <= LA_Chopparts(V_i)do  /*Point r_4*/
                      begin
                                   read LA_Part_Length(V_i,V_Chop_Parts) from V_PrimaryPtr
                                   set LA_Part_Length(V_i,V_Chop_Parts) to
(LA_Part_Length(V_i,V_Chop_Parts)/16)
                                   /*print "The part length you get from", V_i "strips",
V_Chop_Parts " part is = ", LA_Part_Length(V_i,V_Chop_Parts)to message*/

                                   inc V_Chop_Parts by 1
                            end
          /* End of while of Point r_4*/
                    inc V_i by 1
```

```
                set V_Chop_Parts to 1
                end                                          /* End of while of Point r_3*/
 clone 1 load to P_Rawmaterial_wait nlt L_Board
        /*The following code erases all the attributres stored, so that they are not carried forward
        by the next load*/

        set V_Chop_Parts to 1
        set V_i to 1
                while V_i <= LA_Pieces do                    /*Point r_6*/
           begin
                        while V_Chop_Parts <= LA_Chopparts(V_i)do  /*Point r_7*/
                   begin
                                set LA_Part_Length(V_i,V_Chop_Parts) to 0
                                /*print "Attribute erased"to message*/
                                inc V_Chop_Parts by 1
                         end
        /* End of while of Point r_7*/
                inc V_i by 1
                set V_Chop_Parts to 1
                end                                  /* End of while of Point r_6*/

        set V_i to 1
                while V_i <= LA_Pieces do       /*Point r_5*/
           begin
                        set LA_Chopparts(V_i) to 0
                        set LA_Stripwidth(V_i) to 0
                        /*print "Attribute erased"to message*/
                        inc V_i by 1
                end                                                    /* End of while of Point
r_5*/
        /* The code to erase the attributes end here*/
 wait for 0.01 sec
 end                                /* End of while of Point r_1*/
send to die
end             /* End of process P_read*/


begin P_Rawmaterial_wait arriving
 set LA_in_RM to ac
 inc V_RM_Inv by 1
 inc V_Bfeet_inRM by LA_Length
 move into Q_RM                                  /*Infinite Capacity*/
 move into Q_RM_batch                  /* batch queue, Capacity = V_batchSizeofBoards*/
 wait to be ordered on OL_RM
end

begin P_Rawmaterial_transfer arriving
 while 1 = 1 do
 begin
  if V_RM_Inv = 0 then wait for 1 min            /*Outer if*/
  else
```

```
  begin
    wait for 5 sec
    if V_RM_Inv >= V_batchSizeofBoards and Q_Planer_wait remaining space >=
V_batchSizeofBoards and V_Break_On = 0 then /* 1st if*/
      /* make the batch = 100*/
      begin
              wait for u 3.5, 1.5 min          /*time to get the fork lift and look for the boards pack*/
              wait for u 1.5,0.5 min           /*time to take the pack from store to planer*/
       order V_batchSizeofBoards loads from OL_RM to P_Planer          /*batch of 50*/
       print "Sending " V_batchSizeofBoards " loads to the Planer" to message
       dec V_RM_Inv by V_batchSizeofBoards
      end           /* End of 1st if*/
    else if V_RM_Inv < V_batchSizeofBoards and Q_RM current loads = 0 and V_RM_Inv >= 1
and V_Break_On = 0 then /* 2nd if*/
      /* If these are the last remaining parts*/
      begin
              wait for u 3.5, 1.5 min          /*time to get the fork lift and look for the boards pack*/
              wait for u 1.5,0.5 min           /*time to take the pack from store to planer*/
       order all loads from OL_RM to P_Planer
       set V_RM_Inv to 0
       /*This last load orders all the remaining loads to the next process*/
       print "Sending All loads to the Planer" to message
      end           /* End of 2nd if*/
  end               /* End of Outer if*/
 end /*End of while*/
end


begin P_Planer arriving
 /* tabulate (ac - LA_in_RM)/60 in T_RMtime
 set LA_in_Planer to ac  */
 dec V_Bfeet_inRM by LA_Length
 inc V_Bfeet_inPlaner by LA_Length
 inc V_Planer_Inv by 1

 if this load type = L_Dummy then
           begin
            move into Q_Planer_wait
            wait for 2 sec
            move into Q_Planer
            wait for 3 sec
            move into Q_Planer_out
            wait for 2 sec
            /*wait for u 4, 1 min     /*time to get the fork lift and take to the Rip*/
            order all loads from OL_Rip to P_Rip
            /*This last load orders all the remaining loads to the next process*/
    print "Sending All loads to the Planer" to message
    set V_Planer_Inv to 0           There are no more boards in Planer */
    wait for 1 min
    send to P_Rip
   end
```

```
  else
    begin
            move into Q_Planer_wait       /*Capacity = 300*/
            move into Q_Planer                    /*Capacity = 1*/
            use R_Op(1) for u 1.5,0.5 sec  /*loading time*/
            /*print "The planer will be used for ", LA_Length/60 "seconds" to message*/
            use R_Planer for (LA_Length/25) sec          /*time for the Planer to process a
board*/
            use R_Op(2) for u 1.5,0.5 sec  /*unloading time*/
            move into Q_Planer_out                /*Capacity = 300*/
            wait to be ordered on OL_Rip
    end
end

begin P_Planer_transfer arriving
 while 1 = 1 do
 begin
   if V_Planer_Inv = 0 then wait for 1 min                  /*Outer if*/
   else
     begin
       wait for 5 sec
       if Q_Planer_out current loads >= V_batchSizeofBoards and Q_Rip_wait remaining space >=
V_batchSizeofBoards and V_Break_On = 0 then/* 1st if*/
        /* make the batch = 100*/
       begin
              wait for u 3,1 min                        /*time to get the fork lift and take to the Rip*/
        order V_batchSizeofBoards loads from OL_Rip to P_Rip    /*batch of 100*/
        print "Sending " V_batchSizeofBoards " loads to the RIP" to message
        dec V_Planer_Inv by V_batchSizeofBoards
       end          /* End of 1st if*/
       else if V_Planer_Inv < V_batchSizeofBoards and V_Planer_Inv >= 1 and V_RM_Inv = 0
and V_Break_On = 0 then /* 2nd if*/
       /* If these are the last remaining parts*/
       begin
              wait for u 3, 1 min                       /*time to get the fork lift and take to the Rip*/
        order all loads from OL_Rip to P_Rip       /*order all the remaining loads*/
        set V_Planer_Inv to 0
         /*This last load orders all the remaining loads to the next process*/
        print "Sending All loads to the RIP" to message
       end          /* End of 2nd if*/
     end   /* End of Outer if*/
 end /*End of while*/
end


begin P_Rip arriving

 if this load type = L_Dummy then
  begin
    wait for 1 min
          move into Q_Rip_wait
```

```
          wait for 3 sec
          move into Q_Rip
          wait for 3 sec
          send to P_Count
 end
else
begin
 dec V_Bfeet_inPlaner by LA_Length
 inc V_Bfeet_inRip by LA_Length

 inc V_Rip_Inv by 1
 move into Q_Rip_wait          /*Capacity = 300*/
 move into Q_Rip                        /*Capacity = 1*/
 use R_Op(3) for u 2,1 sec       /*loading time*/
 use R_Rip for (LA_Length/25) sec              /*time for the Rip saw to process a board*/
 dec V_Rip_Inv by 1
 dec V_Bfeet_inRip by LA_Length

 set V_CP_from_Strip to 1
 set V_r to 1
 set V_max to LA_Pieces
          while V_r <= V_max do                        /*Point R_1*/
          begin
            set LA_Width to LA_Stripwidth(V_r)
            set LA_Pieces to LA_Chopparts(V_r)
            /*print "The", V_r "Parts width coming out of Rip is", LA_Width to message*/

                    while V_CP_from_Strip <= LA_Pieces do        /*Point R_2*/
                    begin
              set LA_Part_Length(1,V_CP_from_Strip) to LA_Part_Length(V_r,V_CP_from_Strip)
                     /*print "The part length you get from", V_r  " strip after Rip is",
LA_Part_Length(1,V_CP_from_Strip)to message*/
                     inc V_CP_from_Strip by 1
                    end                                                          /* End of while
of Point R_2*/
            clone 1 load to P_Count nlt L_Strip
            inc V_r by 1
            set V_CP_from_Strip to 1
          end                                                    /* End of while of Point R_1*/
 send to die
 end
end

begin P_Count arriving          /* This process triggers production of Salvage parts*/
 if this load type = L_Dummy then
 begin
   order all loads from OL_Sal_Rip to P_Sal_Rip
   set V_Strip_num to 0
   set V_for_Sal to 0
          send to die
 end
```

```
 if V_Strip_num = 0 then          /* in case the dummy load comes before any other load */
          begin
                    send to P_conv
          end
 else
  begin
   inc V_Strip_num by 1                    /*V_Strip_num initial value = 1*/
   if V_Strip_num = 500 then
    begin
     dec V_Strip_num by 299
     order 120 loads from OL_Sal_Rip to P_Sal_Rip
     dec V_for_Sal by 120
    end
   send to P_conv
  end
end

/* Tabulate the inventory */

begin P_Tabulate arriving
          while 1 = 1 do begin
           wait for 5 min
                    tabulate (V_Bfeet_inRM/12) in T_Bfeet_inRM
                    tabulate (V_Bfeet_inPlaner/12) in T_Bfeet_inPlaner
                    tabulate (V_Bfeet_inRip/12) in T_Bfeet_inRip
                    tabulate (V_Bfeet_inChop/12) in T_Bfeet_inChop
                    tabulate (V_Bfeet_inMoulder/12) in T_Bfeet_inMoulder

                    set V_Bfeet_Total_InvInSystem = ( (V_Bfeet_inRM/12) +
(V_Bfeet_inPlaner/12) + (V_Bfeet_inRip/12) + (V_Bfeet_inChop/12) + (V_Bfeet_inMoulder/12)
)
                    tabulate V_Bfeet_Total_InvInSystem in T_V_Bfeet_Total_InvInSystem
          end
          send to die
end

/* ----------------- THE FOLLOWING SECTION CREATES SALVAGE PARTS ----------------
-*/

begin P_read_S arriving
 while V_SalvagePtr eof = false do begin                    /* While there is unread data*/
  read LA_Width, LA_Length, LA_Pieces_S from V_SalvagePtr
          /*print "The Board Width is",  LA_Width to message
          print "The Board Length is",  LA_Length to message
          print "The number of strips you will get is",  LA_Pieces_S  to message*/

  set V_s to 1
   while V_s <= LA_Pieces_S do
   begin
          read LA_Salvage_Parts(V_s) from V_SalvagePtr
```

```
          /*print "The Salvage parts you get from", V_s "strip is", LA_Salvage_Parts(V_s) to
message*/
          inc V_s by 1
  end

  set V_Sal_Parts to 1
  set V_s to 1
   while V_s <= LA_Pieces_S do
   begin
             while V_Sal_Parts <= LA_Salvage_Parts(V_s)do
             begin
              read LA_Sal_partwidth(V_s,V_Sal_Parts), LA_Sal_partlength(V_s,V_Sal_Parts) from
V_SalvagePtr
              /*print "The Salvage part width you get from", V_s,  "strips", V_Sal_Parts,  "part is",
LA_Sal_partwidth(V_s,V_Sal_Parts)to message
              print "The Salvage part length you get from", V_s,  "strips", V_Sal_Parts,  "part is",
LA_Sal_partlength(V_s,V_Sal_Parts)to message*/
              inc V_Sal_Parts by 1
             end
          inc V_s by 1
          set V_Sal_Parts to 1
   end
  clone 1 load to P_Sal_read nlt L_Sal_Board

          wait for 0.01 sec
/* The following code sets all the load attribute values to 0*/
  set V_Sal_Parts to 1
  set V_s to 1
  while V_s <= LA_Pieces_S do
  begin
    while V_Sal_Parts <= LA_Salvage_Parts(V_s)do
          begin
            set LA_Sal_partwidth(V_s,V_Sal_Parts) to 0
            set LA_Sal_partlength(V_s,V_Sal_Parts) to 0
            inc V_Sal_Parts by 1
          end
          inc V_s by 1
          set V_Sal_Parts to 1
  end

  set V_s to 1
  while V_s <= LA_Pieces_S do
  begin
   set LA_Salvage_Parts(V_s) to 0
   inc V_s by 1
  end /*Code for setting the load attributes to 0 ends here*/
 end
end


begin P_Sal_read arriving
```

```
  wait for 0.1 sec
  set V_S_CP_from_Strip to 1
  set V_S_r to 1
  set V_S_max to LA_Pieces_S
  while V_S_r <= V_S_max do  /* Point Sr_1*/
  begin
    set LA_Pieces_S to LA_Salvage_Parts(V_S_r)
    while V_S_CP_from_Strip <= LA_Pieces_S do /* Point Sr_2*/
    begin
      set LA_Sal_partwidth(1,V_S_CP_from_Strip) to
LA_Sal_partwidth(V_S_r,V_S_CP_from_Strip)
      /*print "The Salvage part width you get from", V_S_r,  "strips", V_S_CP_from_Strip,  "part
is", LA_Sal_partwidth(1,V_S_CP_from_Strip)to message*/
      set LA_Sal_partlength(1,V_S_CP_from_Strip) to
LA_Sal_partlength(V_S_r,V_S_CP_from_Strip)
      /*print "The Salvage part length you get from", V_S_r,  "strips", V_S_CP_from_Strip,  "part
is", LA_Sal_partlength(1,V_S_CP_from_Strip)to message*/
      inc V_S_CP_from_Strip by 1
    end                                                    /*End of while of Point Sr_2*/
   if LA_Pieces_S = 0 then clone 1 load to P_Sal_End nlt L_No_Sal_Strip
   else clone 1 load to P_Sal_Rip_wait nlt L_Sal_Strip
   inc V_S_r by 1
   set V_S_CP_from_Strip to 1
   end                                        /*End of while of Point Sr_1*/
end

begin P_Sal_End arriving
 send to die
end

begin P_Sal_Rip_wait arriving
 set LA_in_RM to ac
 inc V_for_Sal by 1
 move into Q_Sal_Rip_wait       /*Infinite Capacity*/
 wait to be ordered on OL_Sal_Rip
end

begin P_Sal_Rip arriving
 /*  set LA_in_Sal_Rip to ac */
   inc V_Sal_Rip by 1
   move into Q_waiting_for_rip  /* Capacity = 300*/
   move into Q_Sal_Rip             /* Capacity = 1*/
   use R_Op(9) for u 2,1 sec                  /*Loading time*/
   use R_Sal_Rip for (LA_Length/33.6)sec
   dec V_Sal_Rip by 1
   send to P_conv
end

/* ------------------------- SALVAGE PARTS SECTION ENDS HERE ------------------------- */

begin P_conv arriving
```

```
/*      dec V_Bfeet_inRM by LA_Length
        inc V_Bfeet_inPlaner by LA_Length
*/      inc V_conv_Inv by 1
        move into conv.sta_rip_out
        travel to conv.sta_chop_in
        send to P_Chop
end

begin P_Chop_Q_check arriving
 while 1=1 do
 begin
   wait for 20 sec
   if Q_Chop_wait current loads >= 100 and R_Rip status is up then take down R_Rip
   else if Q_Chop_wait current loads <= 20 and R_Rip status is down then bring up R_Rip
 end
end

begin P_Chop arriving
 /* set LA_in_Chop to ac
 if this load type = L_Sal_Strip then
  tabulate (ac - LA_in_Sal_Rip)/60 in T_Riptime
 else tabulate (ac - LA_in_Rip)/60 in T_Riptime */
 inc V_Bfeet_inChop by LA_Length

 move into Q_Chop_wait                  /*Capacity = 2000*/
 move into Q_Chop                       /*Capacity = 1*/
 use R_Op(4) for weibull 2.5,3 sec      /*machine setting time*/
 use R_Chop for u 1.5,0.5 sec           /*time for the Rip saw to process a board*/
 dec V_conv_Inv by 1

 if this load type = L_Sal_Strip then
 begin                                                  /*Point S_C_1*/
   /*print "I am a salvage part" to message*/
   set V_c_S to 1
   while V_c_S <= LA_Pieces_S do /*Point S_C_2*/
   begin
     set LA_Width to LA_Sal_partwidth(1,V_c_S)
     /*print "This Salvage parts Width = " LA_Width to message*/
     set LA_Length to LA_Sal_partlength(1,V_c_S)
          /*print "This Salvage parts Length = " LA_Length to message*/
          clone 1 load to P_Moulder_wait nlt L_Primary_Part
          inc V_c_S by 1
   end                  /*End of while of Point S_C_2*/
  wait for 0.01 sec
  send to die
 end      /*End of if of Point S_C_1*/

 else
  begin
   set V_c to 1
        while V_c <= LA_Pieces do       /* Point C_1*/
```

```
          begin
            set LA_Width to LA_Width
            /*print "This parts Width is", LA_Width to message*/
            set LA_Length to LA_Part_Length(1,V_c)
            /*print "This parts Length is", LA_Length to message*/
            clone 1 load to P_Moulder_wait nlt L_Primary_Part
            inc V_c by 1
          end                          /* End of while of Point C_1*/
  wait for 0.01 sec
  send to die
 end
end                 /* End of P_Chop */


begin P_Sal_Rip_Op_Check arriving
/*This process checks if the operator at the Sal_Rip process is needed there or not*/
 if V_Sal_Rip >= 1 and R_Op(9) status = 0 and R_Op(12) status = 1 and V_Op_down = 0 and
V_Break_On = 0 then
  begin
          dec C_Op capacity by 1          /*Initial Capacity of C_Op = 3*/
          set V_Op_down to 1
          wait for 3 sec
          take down R_Op(12)
    print "Capacity of operators at Q_Chop_Out = " C_Op capacity to message
    wait for u 2,1 min
    bring up R_Op(9)
    if R_Op(9) status = 0 then bring up R_Op(9)
  end
 else if V_Sal_Rip = 0 and R_Op(9) status = 1 and R_Op(12) status = 0 and V_Op_down = 1 and
V_Break_On = 0 then
  begin
          take down R_Op(9)
          wait for u 2,1 min /*time to go from Sal Rip machine 1 to chop out queue*/
    bring up R_Op(12)
    inc C_Op capacity by 1
    set V_Op_down to 0
    print "Capacity of operators at Q_Chop_Out = " C_Op capacity to message
  end
 send to die
end


begin P_Planer_Op_Check arriving
 if V_RM_Inv = 0 and V_Planer_Inv = 0 and R_Op(10) status = 0 and R_Op(1) status = 1 and
V_DupOp_down = 1 and V_Break_On = 0 then
  begin
    take down R_Op(1)
    take down R_Op(2)
    wait for u 5,1.5 min  /*Time to go from Planer to Chop out queue*/
    bring up R_Op(10)
    bring up R_Op(11)
```

```
          /*Op 15 and 16 are duplicates of Op 1 and 2. They are up only when 1 and 2 are down*/
    inc C_Op capacity by 2
    set V_DupOp_down to 0
    print "With Planer Operators, Capacity of operators at Q_Chop_Out = " C_Op capacity to
message
   end
  else if (V_RM_Inv <> 0 or V_Planer_Inv <> 0) and R_Op(10) status = 1 and R_Op(1) status =
0 and V_DupOp_down = 0 and V_Break_On = 0 then
   begin
           set V_DupOp_down to 1
           dec C_Op capacity by 2
           wait for 3 sec
           take down R_Op(10)
    take down R_Op(11)
           print "Without Planer Operators,Capacity of operators at Q_Chop_Out = " C_Op
capacity to message
           wait for u 5,1.5 min      /*Time to go from Chop out queue to Planer*/
    bring up R_Op(1)
    bring up R_Op(2)
   end
 send to die
end


begin P_Moulder_wait arriving
  inc C_partsInChopOut by 1              /* capacity = infinite*/
  move into Q_Chop_out          /*Capacity = infinite*/
  wait for 0.01 sec
  clone 1 load to P_Sal_Rip_Op_Check
  clone 1 load to P_Planer_Op_Check
  inc C_Op by 1 /*Counter with initial capacity = 2*/

  if C_Op capacity = 1 then       use R_Op(5) for weibull 1.5, 2 sec

  else if C_Op capacity = 3 then
          begin
           choose a resource from among R_Op(10),R_Op(11),R_Op(5) whose current loads = 0
             save choice as A_Rptr
           use A_Rptr for weibull 1.5,2 sec
          end

  else if C_Op capacity = 2 then
          begin
           choose a resource from among R_Op(12),R_Op(5) whose current loads = 0
             save choice as A_Rptr          /*A_Rptr  is a load attribute of type ResourcePtr,
Capacity = 1 */
           use A_Rptr for weibull 1.5,2 sec
          end

  else if C_Op capacity = 4 then
          begin
```

```
          choose a resource from among R_Op(12), R_Op(10),R_Op(11),R_Op(5) whose current
loads = 0
            save choice as A_Rptr
           use A_Rptr for weibull 1.5,2 sec
          end
  else print "error in logic" to message

  move into Q_Trolley    /*capacity = infinite*/
  dec C_Op by 1
/*  tabulate (ac - LA_in_Chop)/60 in T_Choptime */
  dec C_partsInChopOut by 1
   dec V_Bfeet_inChop by LA_Length
  send to P_Parts_sort
end

begin P_Parts_sort arriving
 wait for 0.01 sec
 inc C_Parts_sort by 1              /*Counter of capacity 1*/
 set V_index to 1
 while V_index <= V_P do
 begin             /* Point F_1*/
  if LA_Width = V_CB_PartWidth(V_index) and LA_Length = V_CB_PartLength(V_index)
then
    begin/* Point F_2*/
           inc C_Final_Quantity(V_index) by 1
     if C_Final_Quantity(V_index) current value <= V_CB_PartQuantity(V_index) then
       begin
            /*print "We have", C_Final_Quantity(V_index)current value " parts of part size",
LA_Width "by", LA_Length to message*/
                    dec C_Parts_sort by 1

                    send to P_for_Moulding
                   end
     else if C_Final_Quantity(V_index) current value > V_CB_PartQuantity(V_index) then
            begin
       /*print " We have" C_Final_Quantity(V_index)current value " EXTRA parts of part size",
LA_Width "by", LA_Length to message*/
        dec C_Parts_sort by 1

        send to P_Extra_Inv
             end
           end              /* End of if of Point F_2*/
 inc V_index by 1
 end                        /* End of while of Point F_1*/
 dec C_Parts_sort by 1


 send to P_Not_Needed
end

begin P_for_Moulding arriving
```

```
/*          set LA_in_Mould to ac  */
            inc V_M_wait_Inv by 1
            inc V_Bfeet_inMoulder by LA_Length
            wait to be ordered on OL_Moulder
end


begin P_Moulder_transfer arriving
 while 1 = 1 do
 begin
   wait for 5 sec
     if V_M_wait_Inv >= V_batchSizeofComponents and V_Break_On = 0 then /* 1st if*/
    /* If there are 400 parts and remaining space in Q_Moulder_wait >= 400*/
      begin
        wait for e 2 min                        /*time to get the fork lift and take to the Moulder*/

        order V_batchSizeofComponents loads from OL_Moulder to P_Part_q                 /*batch
of 200*/
        dec V_M_wait_Inv by V_batchSizeofComponents
      end           /* End of 1st if*/

    else if V_M_wait_Inv < V_batchSizeofComponents and V_RM_Inv = 0 and V_Planer_Inv =
0 and Q_Rip_wait current loads = 0 and Q_Rip current loads = 0 and V_conv_Inv = 0 and
Q_Chop_wait current loads = 0 and Q_Chop current loads = 0 and Q_Chop_out current loads = 0
and V_Break_On = 0 then /* 2nd if*/
      /* If these are the last remaining parts*/
      begin
        wait for e 2 min                        /*time to get the fork lift and take to the Moulder*/
        order all loads from OL_Moulder to P_Part_q                 /*take off the remaining loads*/
        set V_M_wait_Inv to 0
      end           /* End of 2nd if*/
  end /*End of while*/
end

begin P_Part_q  arriving
  wait for 0.01 sec
  inc C_Part_q by 1                 /* Counter of capacity 1*/
  inc V_M_Inv by 1
  move into Q_Moulder_wait              /*Capacity = infinite*/
    set V_EI to 1
          while V_EI <= 25 do     /*There are 25 part types*/
                begin
                  if LA_Width = V_CB_PartWidth(V_EI) and LA_Length =
V_CB_PartLength(V_EI) then
                    begin
                      move into Q_Part(V_EI)        /*Capacity = 500. Move into the
corresponding part's queue*/
                      dec C_Part_q by 1
                      wait to be ordered on OL_Part(V_EI)
             end
       inc V_EI by 1
```

```
   end
end


begin P_to_Moulder arriving
 while 1 = 1 do begin                /*1st while*/
  wait for 5 sec

  set V_tM to 1
    while V_tM <= 25 do begin  /*2nd while  batch size = V_batchSizeofComponentsAtMoulder
*/

     if OL_Part(V_tM) current loads > 0 and (Q_Moulder(1) remaining space >=
V_batchSizeofComponentsAtMoulder or Q_Moulder(2) remaining space >=
V_batchSizeofComponentsAtMoulder or Q_Moulder(3) remaining space >=
V_batchSizeofComponentsAtMoulder ) then begin
                choose a queue from among Q_Moulder(1),Q_Moulder(2),Q_Moulder(3) whose
remaining space >= V_batchSizeofComponentsAtMoulder
                                save choice as A_Qptr
        /*              print "the queue selected is " A_Qptr to message          */
                        set A_index = A_Qptr index /* set the load's attribute to the value of the
queue's "index" attribute */

        if OL_Part(V_tM)current loads >= V_batchSizeofComponentsAtMoulder and
V_Break_On = 0 then
                begin
                  print "Sending " V_batchSizeofComponentsAtMoulder " loads from Part Queue
"       V_tM " to Moulder " A_index to message
                  if A_index = 1 then begin
                        create 1 load of type L_M_1 to P_Changeover
                        wait for 0.01 sec
                order V_batchSizeofComponentsAtMoulder loads from OL_Part(V_tM) to
P_Moulder1              /*batch of V_batchSizeofComponentsAtMoulder*/
          end
          else if A_index = 2 then begin
                create 1 load of type L_M_2 to P_Changeover
                wait for 0.01 sec
                order V_batchSizeofComponentsAtMoulder loads from OL_Part(V_tM) to
P_Moulder2
          end
          else if A_index = 3 then begin
                create 1 load of type L_M_3 to P_Changeover
                wait for 0.01 sec
                order V_batchSizeofComponentsAtMoulder loads from OL_Part(V_tM) to
P_Moulder3
          end
          wait for 0.01 sec
                end
        else if OL_Part(V_tM)current loads < V_batchSizeofComponentsAtMoulder and A_Qptr
current loads = 0 and V_RM_Inv = 0 and V_M_wait_Inv = 0 and Q_Trolley current loads = 0
and V_Break_On = 0 then
```

```
                begin
                  print "Sending all loads from Part Queue "        V_tM " to Moulder " A_index
to message
          if A_index = 1 then begin
                create 1 load of type L_M_1 to P_Changeover
                        wait for 0.01 sec
                order all loads from OL_Part(V_tM) to P_Moulder1                /*sending all
remaining loads*/
          end
          else if A_index = 2 then begin
                create 1 load of type L_M_2 to P_Changeover
                        wait for 0.01 sec
                order all loads from OL_Part(V_tM) to P_Moulder2
          end
          else if A_index = 3 then begin
                create 1 load of type L_M_3 to P_Changeover
                        wait for 0.01 sec
                order all loads from OL_Part(V_tM) to P_Moulder3

          end
          wait for 0.01 sec
         end

     end /* end of if */
                  wait for 0.02 sec
                 inc V_tM by 1
      wait for 0.01 sec

     end/*end of 2nd while*/
  end             /*end of 1st while*/
end


begin P_Changeover arriving
 if this load type = L_M_1 then
          begin
                take down R_Moulder(1)
                wait for weibull 1.5, 3 min                /* weibull alpha, beta*/
                bring up R_Moulder(1)
          end
 else if this load type = L_M_2 then
          begin
                take down R_Moulder(2)
                wait for weibull 1.5, 3 min
                bring up R_Moulder(2)
          end
 else if this load type = L_M_3 then
          begin
                take down R_Moulder(3)
                wait for weibull 1.5, 3 min
                bring up R_Moulder(3)
```

```
        end
 send to die
end


begin P_Moulder1 arriving
        move into Q_Moulder(1)          /*Capacity = 252*/
        use R_Op(6) for weibull 2.5, 2 sec
   use R_Moulder(1) for (LA_Length/15) sec
   dec V_M_Inv by 1
        send to P_Moulder_out
end

begin P_Moulder2 arriving
        move into Q_Moulder(2)          /*Capacity = 252*/
        use R_Op(7) for weibull 2.5, 2 sec
   use R_Moulder(2) for (LA_Length/15) sec
   dec V_M_Inv by 1
        send to P_Moulder_out
end

begin P_Moulder3 arriving
        move into Q_Moulder(3)          /*Capacity = 252*/
        use R_Op(8) for weibull 2.5, 2 sec
   use R_Moulder(3) for (LA_Length/18) sec
   dec V_M_Inv by 1
        send to P_Moulder_out
end

begin P_Moulder_out arriving
        inc V_M_out_Inv by 1

  wait for 0.01 sec

        move into Q_Moulder_out                 /*Capacity = 1000*/
        wait to be ordered on OL_Final
end


begin P_Moulder_out_Transfer arriving
 while 1 = 1 do begin
    wait for 10 sec                 /* time to prevent CPU space*/
     if V_M_out_Inv >= V_batchSizeofComponentsAtMoulder and V_Break_On = 0 then /* 1st
if*/
     /* If there are 250 parts */
      begin
        wait for e 1 min                        /*time to get the fork lift and take to the Moulder*/
        order V_batchSizeofComponentsAtMoulder loads from OL_Final to P_Finish
           /*batch of 200*/
        dec V_M_out_Inv by V_batchSizeofComponentsAtMoulder
      end           /* End of 1st if*/
```

```
     else if V_M_out_Inv < V_batchSizeofComponentsAtMoulder and V_M_Inv = 0 and
V_M_wait_Inv = 0 and V_Break_On = 0 then /* 2nd if*/
      /* If these are the last remaining parts*/
      begin
        wait for e 1 min                    /*time to get the fork lift and take to the Moulder*/
        order all loads from OL_Final to P_Finish
        set V_M_out_Inv to 0
      end          /* End of 2nd if*/
 end /*End of while*/
end

begin P_Finish arriving
/*  tabulate (ac - LA_in_Mould)/60 in T_Mouldtime       */
  tabulate (ac - LA_in_RM)/60 in T_time_inSystem
  dec V_Bfeet_inMoulder by LA_Length
  inc V_Bfeet_FinalNeeded by LA_Length
  inc C_Total_Primary by 1

 /*print "Total number of primary-parts available now are", C_Total_Primary current value to
message*/
  move into Q_Final_Parts                   /*Capacity = infinite*/

  wait to be ordered on OL_Finish
  dec V_Bfeet_FinalNeeded by LA_Length
  send to die
end

begin P_Not_Needed arriving
 move into Q_not_needed
 inc V_Lineal_foot by LA_Length
 print "Total Lineal Foot of Parts not needed = "  V_Lineal_foot to LBL_Not_Needed
 wait to be ordered on OL_Not_Needed
 dec V_Lineal_foot by LA_Length
 send to die
end

begin P_Extra_Inv arriving       /* Also checks Inventory levels*/
  move into Q_Extra_Inv                   /*Capacity = infinite*/
  inc V_Extra_Inv by 1
  inc V_Bfeet_Extra_Inv by LA_Length
  wait to be ordered on OL_Next_Day
  dec V_Extra_Inv by 1
  dec V_Bfeet_Extra_Inv by LA_Length
  send to P_Parts_sort
end

begin P_Checking_Extra_Inv arriving
          inc C_Checking_Extra_Inv by 1          /* Counter of capacity 1*/
          wait for 0.01 sec
          dec C_Checking_Extra_Inv by 1
```

```
        send to P_Parts_sort
end


/*-----------THE FOLLOWING SECTION CREATES DOWNTIME OF MACHINES ----------
------*/

begin P_Operator_Down arriving
 while 1=1 do
 begin

        wait for 2 hr      /*From start of shift to Tea-break, wait for 2 hr and 30 min*/

        take down conv.M_sec1/*Stop the conveyor*/
        set V_OD to 1
        while V_OD <= 12 do begin
         take down R_Op(V_OD)
         inc V_OD by 1
        end
        set V_Break_On to 1              /* Variable denoting that break is on (if 1) or off (if 0)*/
        print "Opeartors on TEA BREAK and Conveyor stopped" to message

   wait for 0.5 min        /* Tea-break*/

   bring up conv.M_sec1          /*bring up the conveyor*/
   set V_OD to 1
        while V_OD <= 12 do begin
         bring up R_Op(V_OD)
         inc V_OD by 1
        end
        set V_Break_On to 0
        print "Opeartors Back from TEA BREAK and Conveyor started" to message

        wait for 3 hr      /*From Tea-break to Lunch*/

        take down conv.M_sec1
        set V_OD to 1
        while V_OD <= 12 do begin
         take down R_Op(V_OD)
         inc V_OD by 1
        end
        set V_Break_On to 1
        print "Opeartors on Lunch" to message

   wait for 0.5 min        /* Lunch-break*/

   bring up conv.M_sec1
   set V_OD to 1
        while V_OD <= 12 do begin
         bring up R_Op(V_OD)
```

```
          inc V_OD by 1
         end
         set V_Break_On to 0
         print "Opeartors Back from Lunch" to message

         wait for 3 hr      /*From Lunch to shift end, 3 hours */
         take down conv.M_sec1
         set V_OD to 1
         while V_OD <= 12 do begin
          take down R_Op(V_OD)
          inc V_OD by 1
         end
         set V_Break_On to 1
         print "Today's Shift ENDS" to message

         wait for 1 min    /* Shift end to next day*/
         bring up conv.M_sec1
   set V_OD to 1
         while V_OD <= 12 do begin
          bring up R_Op(V_OD)
          inc V_OD by 1
         end
         set V_Break_On to 0
         set V_Strip_num to 1
         set V_for_Sal to 0
         set V_DupOp_down to 1
         set V_Op_down to 0
         print " Shift BEGINS" to message
 end
end
```

Appendix B: Results

B.1 Key Variables

Variable	Unit	Remarks
T_AvgTimeInSystem	Minutes	Measures Lead time in the System. This is the measured from the moment boards enter the raw material store to the time finished parts enter the finished goods store.
T_Bfeet_InvInSystem_Avg / T_Bfeet_InvInSystem	Lineal feet	Measures average inventory in the system, for 20 days. This is the combined inventory at the raw material store, planer, rip, chop and molder stations.
T_BfeetExtraInv_Avg / T_BfeetExtraInv	Lineal feet	Measures inventory of extra parts per day.
T_BfeetFinalNeeded_AvgPerDay / T_BfeetNeededperDay	Lineal feet	Measures inventory of needed parts (parts on the cutting bill, and in the quantities needed) per day.
T_BfeetNotNeeded_AvgPerDay / T_BfeetNotNeeded	Lineal feet	Measures inventory of not-needed parts (parts not on the cutting bill) per day.
T_FinalPartsperDay_Avg / T_FinalPartsQuantity	--	Mesaures the number of needed parts obtained per day.
Utilization_Planer	% (percent)	Utilization of Planer.
Utilization_Rip	%	Utilization of Rip Machine.
Utilization_Sal_Rip	%	Utilization of Salvage Rip Machine.
Utilization_Chop	%	Utilization of Chop Machine.
Utilization_Molder1	%	Utilization of Molder 1.
Utilization_Molder2	%	Utilization of Molder 2.
Utilization_Molder3	%	Utilization of Molder 3.

B.2 Statistical Tables

B.2.1 Results for 5 Runs

B.2.1.1 Push:

Run Results for pushEasy2x_ver2SingleScenario

File Edit Responses Help

A	B	C	D	E	F
No changed factors					
	Run 1	Run 2	Run 3	Run 4	Run 5
	RN Set 1	RN Set 2	RN Set 3	RN Set 4	RN Set 5
T_AvgTimeInSystem	321.4667	319.8965	320.6241	321.7906	320.4611
T_Bfeet_InvInSystem_Avg	258690.805	258589.3348	258603.4277	258691.8104	258600.9031
T_BfeetExtraInv_Avg	2906.4704	2906.4704	2906.4704	2906.4704	2906.4704
T_BfeetFinalNeeded_AvgPerDay	32033.1952	32033.1952	32033.1952	32033.1952	32033.1952
T_BfeetNotNeeded_AvgPerDay	1045.5373	1045.5373	1045.5373	1045.5373	1045.5373
T_FinalPartsperDay_Avg	17899.9474	17899.9474	17899.9474	17899.9474	17899.9474
Utilization_Chop	0.282	0.282	0.282	0.282	0.282
Utilization_Molder1	0.299	0.296	0.291	0.295	0.298
Utilization_Molder2	0.294	0.285	0.289	0.293	0.288
Utilization_Molder3	0.244	0.254	0.255	0.249	0.25
Utilization_Planer	0.31	0.31	0.31	0.31	0.31
Utilization_Rip	0.31	0.31	0.31	0.31	0.31
Utilization_Sal_Rip	0.161	0.161	0.161	0.161	0.161

B.2.1.2 Pull:

Run Results for PULL_SingleScenario

File Edit Responses Help

A	B	C	D	E	F
No changed factors					
	Run 1	Run 3	Run 4	Run 5	Run 6
	RN Set 1	RN Set 3	RN Set 4	RN Set 5	RN Set 6
T_AvgTimeinSystem	205.0129	204.4446	204.7424	204.7148	205.1024
T_Bfeet_InvInSystem	146709.5693	146702.6842	146720.1873	146723.803	146738.5345
T_BfeetExtraInv	13.5482	13.5482	13.5482	13.5482	13.5482
T_BfeetNeededperDay	31504.7105	31504.7105	31504.7105	31504.7105	31504.7105
T_BfeetNotNeeded	0.0	0.0	0.0	0.0	0.0
T_FinalPartsQuantity	17516.5789	17516.5789	17516.5789	17516.5789	17516.5789
Utilization_Chop	0.223	0.223	0.223	0.223	0.223
Utilization_Molder1	0.287	0.284	0.284	0.29	0.287
Utilization_Molder2	0.285	0.283	0.283	0.28	0.286
Utilization_Molder3	0.25	0.254	0.254	0.252	0.249
Utilization_Planer	0.314	0.314	0.314	0.314	0.314
Utilization_Rip	0.314	0.314	0.314	0.314	0.314
Utilization_Sal_Rip	0.0	0.0	0.0	0.0	0.0

B.2.2 Summary Statistics

B.2.2.1 Push

Variable	Mean	Standard Deviation	Lower Confidence Interval	Higher Confidence Interval
T_AvgTimeInSystem	320.85	0.7708	31.9.9	321.8
T_Bfeet_InvInSystem_Avg	258,635.26	51.44	258,571.4	258,699.1
T_BfeetExtraInv_Avg	2906.47	0.0		
T_BfeetFinalNeeded_AvgPerDay	32,033	0.0		
T_BfeetNotNeeded_AvgPerDay	1045.54	0.0		
T_FinalPartsperDay_Avg	17,899.95	0.0		
Utilization_Chop	0.282	0.0		
Utilization_Molder1	0.2958	0.0031	0.2919	0.2997
Utilization_Molder2	0.2898	0.0037	0.2852	0.2944
Utilization_Molder3	0.2504	0.0044	0.2449	0.2559
Utilization_Planer	0.31	0.0		
Utilization_Rip	0.31	0.0		
Utilization_Sal_Rip	0.161	0.0		

B.2.2.2 Pull

Variable	Mean	Standard Deviation	Lower Confidence Interval	Higher Confidence Interval
T_AvgTimeInSystem	204.8	0.2615	204.48	205.13
T_Bfeet_InvInSystem	146,719	13.8	146,701.8	146,736.1
T_BfeetExtraInv	13.55	0.0		
T_BfeetFinalNeededperDay	31,504	0.0		
T_BfeetNotNeeded	0.0	0.0		
T_FinalPartsQuantity	17,516.6	0.0		
Utilization_Chop	0.223	0.0		
Utilization_Molder1	0.2864	0.0025	0.2833	0.2895
Utilization_Molder2	0.2834	0.0023	0.2805	0.2863
Utilization_Molder3	0.2518	0.0023	0.2490	0.2563
Utilization_Planer	0.314	0.0		
Utilization_Rip	0.314	0.0		
Utilization_Sal_Rip	0.0	0.0		