1991

Reduction of bottleneck operations in Just-In-Time manufacturing

Shehzad Ahmed
Iowa State University

Recommended Citation

Ahmed, Shehzad, "Reduction of bottleneck operations in Just-In-Time manufacturing" (1991). *Retrospective Theses and Dissertations*. 16776.
https://lib.dr.iastate.edu/rtd/16776

Reduction of bottleneck operations in Just-In-Time manufacturing

by

Shehzad Ahmed

A Thesis Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE

Department: Industrial and Manufacturing Systems Engineering
Major: Industrial Engineering

Iowa State University
Ames, Iowa

1991

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

By the late 1970s, Omark Industries, General Electric, and some consulting firms were all discovering what U.S. consumer electronics firms, camera makers, and auto assemblers, already knew -- that substantial parts of U.S. domestic manufacturing capability were outclassed and rapidly losing ground in the world market. Moreover, these early discoverers of the problem were aware of the means by which the Japanese were making tremendous progress. It was something called Just-In-Time (JIT) manufacturing [4]. Some authors refer to it as "Stockless Production", "Synchronized Production", or "Zero Inventory."

Just-In-Time can be defined as a production system designed to eliminate waste in the manufacturing environment. Waste is anything that does not contribute directly to the value of the product [14]. Activities such as moving, storing, counting, and sorting all add cost to the product but no value. Similarly, backup sources, expediters and safety stocks also add cost but no value. On the other hand, operations such as machining, finishing and packaging add value to the product. Just-In-Time manufacturing is based on the premise that reducing non-value added activities, such as inventory maintenance, scheduling, excess inspection and rework allow company personnel more time to concentrate on value added processes, such as machining, assembly and heat treatment. The tools used to implement JIT range from planning plant layout to minimize response time and material movement, grouping machine tools into cells to machine families of parts, and minimizing setup times [22].

JIT is an organizational philosophy which strives for excellence, indeed perfection. In the broadest sense, its aim is the elimination of all waste and consistent improvement of productivity. This total dedication to the elimination of waste is the heart and soul of the Toyota production system. It also constitutes very source of its profit [17]. However, no matter how determined one may be in his/her desire to eliminate waste, if one does not know what constitutes waste, then there is no way of eliminating it. Thus, it is important to point out where the waste is and that it should appear as waste to everybody. This is the first step toward attaining an improvement in efficiency. Other JIT goals are :

- Zero defects
- Zero setup time
- Zero lot excesses

- Zero handling
- Zero surging
- Zero breakdowns
- Zero lead time [35].

The concept of JIT was initially started in Japan, but now it is widespread throughout all the industrial world. Just-In-Time is quite different from the conventional Just-In-Case approach. Just-In-Case approach provides contingencies to cover unexpected and unforeseen circumstances and the result is excess inventory. Just-In-Time, on the other hand, strives to achieve a stockless production system. Some of the basic differences between the above two approaches are as follows:

Conventional Western Approach (Just-In-Case):

1. Inventory provides safety.
2. Setup time is given.
3. Large lots are efficient.
4. Queues are necessary.
5. Some defects are acceptable.
6. Suppliers are adversaries.

Japanese Approach (Just-In-Time):

1. Safety stock is a waste.
2. Setup time should be minimized, ideally zero.
3. Ideal lot size is one.

4. Queues should be eliminated.
5. Zero defects are necessary and attainable.
6. Suppliers are partners.

JIT means having the right part at the right place at the right time. It means "just enough," i.e., exactly the right quantity, no more no less, not only with respect to parts but with respect to tooling, money, and energy. Actually it means even more. It means we should constantly strive for improvement. We should always ask questions, such as, Can this process be simplified more? Can we produce the same item with less resources?, etc. This search for excellence is a never ending process and forms the basis for Just-In-Time philosophy.

There was a time when the Americans had a major share in the world market in manufacturing. But with the widespread of technology, everybody is in the race today. The Europeans are known for their craftsmanship and reputation for quality. The Far Eastern countries, such as South Korea, Taiwan, Singapore (better known as NIC -- Newly Industrialized Countries, or Pacific Rim countries), have inexpensive labor and a strong desire to grow. The Japanese are known for their superior quality products and a dedicated work force [33]. The Americans have a lead in the high technology industries and they are trying for a comeback by learning Just-In-Time techniques which have been proved highly successful by the Japanese companies.

In the U.S., parts are produced in batches or lots. Lot sizing, whether economic or non-economic, forces us to believe that economic order policies provide the best possible solution. In reality, these formulas take into consideration direct setup costs and inventory carrying costs while machine capacity and loads, along with the available material and its movement are ignored. Most of the companies in the U.S. use the economic order quantity formula (EOQ) to come up with the lot size. The formula for Economic Order Quantity (EOQ) is

EOQ = SQRT (2 * SC * AD / FC * CR)

where SC = Setup Cost
AD = Annual Demand
FC = Factory Cost
CR = Carrying Rate

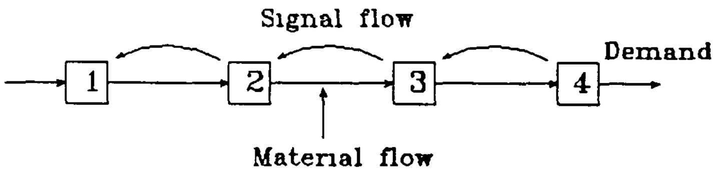
The trend in the U.S. is to keep setup cost constant and concentrate on reducing the unit factory cost (denominator), hence increasing the lot size. Thus, long lead times, large lot sizes, and buffer inventories are common practices in America. On the other hand, the Japanese concentrate on reducing the setup cost (numerator), thus reducing the lot size. They strive for single unit setup times (less than ten minutes). Under JIT, the ideal lot size is one piece. The reduced lot sizes help in minimizing inventory investment, shortening production lead times, reacting faster to demand changes, and uncovering any quality problems.

The Japanese decision to have low setup times and reduced lot sizes has very practical reasons. Being a small overcrowded nation, with limited material resources, it can not afford the luxury to build facilities needed to maintain high inventory levels. The Japanese view the manufacturing process as a giant network of interconnected work centers. At Kawasaki Heavy Industries, Japan, they regard each machine as a point. An assembly line consisting of several machine tools (points) is considered a line and multiple lines make a surface [22]. By developing points into lines, and lines into surfaces on a systematic basis, a perfect arrangement can be reached where a worker would complete his/her job on a part and pass it directly to the next worker just as that person is ready for another piece. The idea is to eliminate queues in order to minimize inventory investment, to shorten lead times, to react faster to demand changes, and to uncover any quality problems.

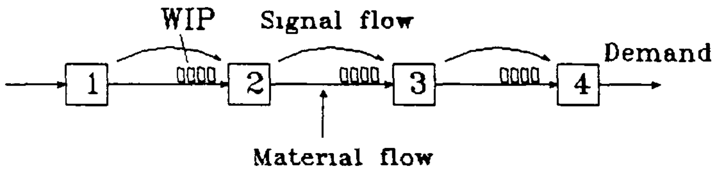
The Western approach toward manufacturing can be regarded as a "push" method. In this process, the planned production quantity is determined by inventory on hand and demand predictions. The product is produced in sequence starting from the first process. This approach calls for running each operation to its full capacity even if the next operation can not handle the output (Figure 1.1). This results in large Work-in-process (WIP) levels, increase in manufacturing lead times, large amount of work remaining at subsequent operation(s) and confusion on the floor. The Japanese approach is the "pull" method in which the procedure is repeated in reverse. As the name suggests, the process pulls work through the factory to meet customer demands. The final process gets the production plan indicating the desired types of products with their quantities and due dates. It withdraws the required quantities from the preceding processes when needed. The preceding processes produce only when the next process withdraws parts. Kanban system is used to convey this information on the factory floor. Kanban means "card" in Japanese. It is some form of paper or card carrying information regarding pickup, transfer and production. Kanban system can be divided into two categories: the dual kanban and the single kanban.

Dual Kanban System uses withdrawl and production kanbans. Withdrawl kanban is used when parts are to be moved between the output and input buffer stocks. The production kanban is used when production is to take place. The main advantage claimed for this system is the extra control it gives, but that comes at the expense of extra complexity. Toyota and its suppliers use this system. Single Kanban System consists of only one card. Parts are produced at one work center according to the daily or weekly schedule but deliveries to the next work center are controlled by move kanban. When a container is empty at next work center, the move kanban is returned to first work center where the kanban authorizes withdrawl of a full container of parts (Figure 1.2). Other devices, such as computer networks or buzzers can be used as long as the concerned operations receive the signal when work is needed at the succeeding operation. The use of kanban system checks the production system from building up excessive inventory stocks.

The Japanese consider inventory as a covering blanket which hides quality problems. The problems may be in the form of machine breakdown, poor quality and high scrap, bad raw material, late delivery of parts, worn out tools or unavailability of material handling equipment and setup persons. They use the analogy that inventory is like the water level in a river and its rocky bottom is considered as

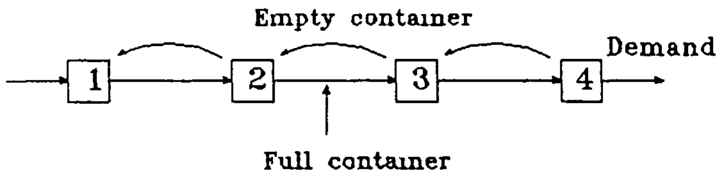


(a) Pull System

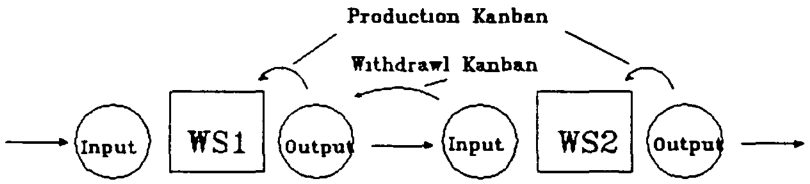


(b) Push System

Figure 1.1: Pull and Push systems



(a) Single Kanban System



(b) Dual Kanban System

Figure 1.2: Single and Dual Kanban systems

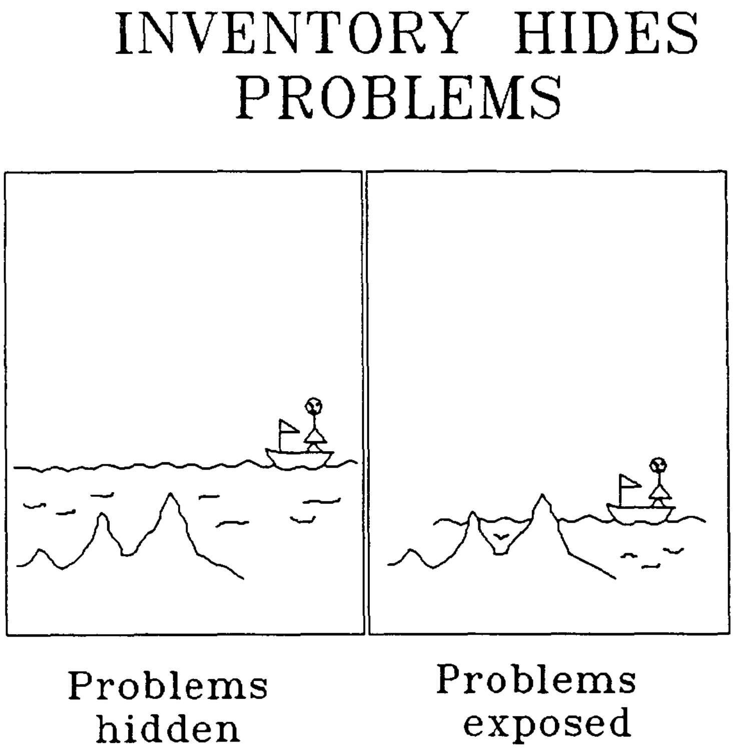


Figure 1.3: Inventory hides problems

a representation of problems that might occur in a shop (Figure 1.3). Lots of water in that river, i.e., excess inventory covers up the problems. The problems, however, are still there. The lower the water level gets, i.e., inventory decreases, the more quality and production problem rocks you can expose and solve. It is better to force the water level down on purpose, particularly in good times, expose the problems and fix them now, before they cause trouble [37].

In the U.S., we are taught negatives. Considerable effort is made to build systems to locate and correct shortcomings and defects. There are times when less than perfect quality is shipped to a customer. This raises a question as to how many times this practice has to be done? Why do the American companies accept less than 100%? [3]. This seems to be one of the problems that some companies are facing with their suppliers, e.g., if a company tells a supplier for 100 and only 100 perfect units. The supplier thinks that 100 is only a ballpark figure. Actual acceptable percentage is say, 90%, while other 10% is for scrap. However, when a company is operating to meet its JIT requirements, a given number is fixed with no if's and then's. That is why a great emphasis is placed on the company-supplier relationship. Suppliers are considered to be partners. They are selected on a long term basis, and are taught the basics of JIT philosophy before making a move towards JIT implementation. The company usually agrees to give majority of its business to one or two preferred vendors in exchange for pre-established quality levels and short delivery lead times. Quantities for parts are usually desired for delivery on short notice [19].

Under the ideal JIT system, on-time delivery of parts is desirable. However, in the real world, no company has been able to totally eliminate inventory. It is economically impractical to supply parts one at a time either between supplier and user or within a factory because of distances, machine availability, etc. The concept of JIT stresses the need for reducing fluctuations and disruptions in the flow of material. While trying to streamline an operation, one faces many problems, such as long setup times, unreliable machines, and long lead times. Moreover, in any plant, there are always some capacity constraint resources (bottlenecks). A bottleneck operation limits the throughput of the plant and dictates the due-date performance. The management should make sure that the bottleneck operation is not scheduled to produce more than its capacity and also not to waste any of its capacity by allowing any slack in its schedule. The JIT approach to solve the problem of a bottleneck operation would involve reducing setup time to produce greater capacity, finding alternative machines, purchasing extra capacity or even subcontracting excess work.

The purpose of this research is to design a simulation model with processing times for operations obtained from a company. The model will be run in a simple mini cell layout and job shop layout. In addition, the simulation will also be done by using lower setup times and by adding a capacity constrained resource (CCR) or a bottleneck machine. All these cases will involve manufacturing of the same three products (product A, B, and C) on the same machines. Each product will have a different batch size which has to be produced. A simulation study will be done to collect necessary results, such as weekly production results, and maximum and average queue sizes in front of different machines. The results of all the above cases will be analyzed and discussed. Various steps utilized by different managers to reduce or eliminate the bottleneck operations will also be discussed in some detail in a later chapter.

2. LITERATURE REVIEW

An underlying principle of JIT is to continually optimize and integrate the manufacturing system. This includes eliminating all non-value added activities, such as inspection, rework etc., and having the focus on improving value. Many U.S. companies including Hewlett Packard (HP), Omark, International Business Machine (IBM), and General Electric (GE) have proved that the JIT philosophy can be implemented successfully by non-Japanese companies.

There has been tremendous interest in JIT over the last decade. Literature is full of case studies of companies which successfully implemented JIT, both in the U.S. and other parts of the world [12,30,33]. A large number of books have also been published on the subject. However, a mid-1989 survey conducted by a market research firm, Dataquest Incorporated, indicated that only thirteen percent of the U.S. manufacturing companies responding to their survey had some type of JIT program underway [28]. Small manufacturing companies (having fewer than 500 employees) account for 98.4% of the 357,863 manufacturing firms in the U.S., and 61.4% of workers employed in manufacturing. In these small companies, the percentage is even lower [29].

The material available in the literature covers a wide range of topics regarding JIT. It includes JIT implementation, quality, delivery and supply aspects of JIT, and importance of employee commitment and participation in the program. The available literature covers broad based concepts of JIT in great details, but it is still hard to find material on specific topics. An example is bottleneck operations. A bottleneck operation is a resource whose capacity is equal to or less than the demand placed on it. Similarly, a non-bottleneck operation is a resource whose capacity is greater than the demand placed on it [9]. These are discussed in some books and journal articles, but there does not seem to be an in-depth study of the topic.

Out of the available literature, Adams, Balas and Zawack [1] discussed a method for solving the minimum makespan of job shop scheduling. They sequenced the machines one by one, successively, recording each time the machine identified as a bottleneck among the machines not yet sequenced. After each sequencing of a new machine, all previously established sequences were locally re-optimized. Items were processed on machines subject to the constraints that the sequence of machines for each job was prescribed and each machine could process only one job at a time. Ten priority dispatching rules were used to solve forty problems. The rules used were FCFS (first Come First Serve), LST (Late Start Time), EFT (Early Finish Time), LFT (Late Finish Time), MINSLK (Minimum Slack), SPT (Shortest Processing Time), LPT (Longest Processing Time), MIS (Most Immediate Successors), FA (First Available), and RANDOM. Of the forty test problems, none of the ten priority dispatching rules dominated all the others. Eight of the ten rules gave the best result on at least one problem, with the remaining two (FA and LPT) never being the best.

Kumar and Vannelli [15] discussed the issue of redesigning the traditional production system into disaggregated cellular production system using group technology (GT). The process led to evaluating critical strategic decisions regarding subcontracting of parts and balancing of capacity between the various cells. The objective of complete disaggregation of the production system was to achieve the concept of a "focused" factory. The use of subcontracting strategy was to reduce problematic capacity requirements and to induce manufacturing efficiency through disaggregated cells. The algorithm required one initial seed (part or machine) assigned to each of the predetermined number of cells. If seed was a part (machine), all machines (parts) attached to it were added to the same cell. The next step formed the boundary set of parts, one for each cell. Part nodes that belonged to more than one boundary set were considered troublesome and were assigned to a bottleneck part set. The procedure was repeated until every node was assigned to either a cell or a bottleneck part cell. The computer implementation of this algorithm was interactive and allowed the designer to freely try various system configurations.

Azadivar and Lee [2] suggested a procedure for determining the optimal number of buffer spaces for each work station so that, for a desired level of machine utilization, the overall in-process inventory was minimized. A Flexible Manufacturing System (FMS) was considered consisting of a set of work stations each having several parallel machines. In addition to a central storage for the system, each work station had a local buffer storage. The objective function was defined as the average number of in-process jobs. By setting a bound on the minimum utilization of work stations, an interesting problem was formed. It was solved by an optimization algorithm called SIMICOM (SIMulation optimization using Integer COMplex search method).

The essence of JIT is to make a product only when it is needed and to make as little as possible to satisfy current needs. There are many different ways to determine lot sizes, but some steps can be taken to move in the direction of smaller and smaller lots. An understanding to be gained about lot sizing is the effect of external constraints. This happens when the lot size is affected by some criteria outside the normal parameters of the product itself. It could be a bottleneck operation or a package size [5]. Pareto analysis shows that most of a company's capital is tied up in a small percentage of its items. If efforts are made to reduce lot sizes of large cost items first, the gains made will more than compensate for the lack of attention on the smaller value items. Thus, it is better to keep very little of the large value items and relatively large amounts of the low value items.

In JIT, the ideal lot size is one. If demand at each production level is met smoothly, both raw material and WIP inventory can be eliminated. However, literature does not have examples of companies which have been able to totally eliminate inventory by implementing JIT [31,33,40]. In most of the cases, it is economically or physically impractical because we will reach lower limit of practicality before reducing the lot size to one. Our aim should be to balance our operations on a daily basis with mixed model scheduling. In the beginning, it may require discrete batch sizes, but we should remember that in JIT there is no fixed target.

Targets are set to achieve them and once achieved, new targets are set in pursuit of a continuous improvement.

Another system for planning and scheduling manufacturing operations, which has gained considerable publicity in the last few years, is OPT (Optimized Production Technology) developed by E. M. Goldratt [10]. Some of the OPT principles are:

1. The utilization of a non-bottleneck resource is not determined by its own capacity, but by some other constraint in the system.
2. An hour lost at a bottleneck is an hour lost to the total system.
3. An hour saved at a non-bottleneck resource is a mirage.
4. Bottleneck resources govern both throughput and inventory.
5. Activating a resource is not synonymous with utilizing a resource [10].

OPT concentrates on the flow of material through the highly utilized bottleneck resources. The long-term utilization of non-bottleneck machines is fixed by the utilization of bottleneck resources. Improving the utilization of non-bottleneck resources will only result in excess inventory. This is in line with the JIT emphasis of maximizing overall efficiency and producing only what is needed, when it is needed.

In the JIT system, the production is driven by the market demand. The release of raw materials into the plant results from a chain reaction initiated when the final operation supplies material to the market place. This chain reaction is accomplished through the use of Kanban or some other signalling device. Inventory is limited and is much lower than the Just-In-Case approach. Current throughput may be lost in the case of a disruption, but in the long run the lower inventory secures future throughput by increasing the competitive edge. JIT systems prevent disruptions by their total preventive maintenance programs.

OPT emphasizes on placing buffers in front of bottleneck operations to cover up for some uncertainties present in the system. This concept is the best explained in the Drum-Buffer-Rope (DBR) approach in "The Race" [9]. Goldratt and Fox used an analogy of comparing the processes on a production floor with a troop of soldiers. In the DBR approach, a rope is directly tied from the weakest soldier to the first row of soldiers. Thus, the front row of soldiers is constrained by the pace of the weakest soldier. While the troops march, the only gap or spreading will be right in front of the weakest soldier. Faster soldiers behind him will be at his heels, and faster soldiers in front of him will be at the heels of first row of soldiers. Suppose one of the soldiers behind the weakest soldier has to stop. Under the JIT system, the whole line has to stop. In the DBR system, the weakest soldier's performance will not be affected. Some spreading (inventory) will occur, but since soldiers following the weakest one are faster (have excess capacity), they will catch up a bit later. Similarly, if a soldier in front of the weakest one stops, there will be no impact on the troop's rate of movement, as long as he starts again before the weakest soldier has closed the gap [9].

In any plant, there are always a few capacity constraint resources (bottlenecks). The strategy is to treat the major bottleneck on the production floor as the pace setter, so that its production rate serves as the pace for the entire plant. The rate at which the first operation is allowed to release material into production should be dictated by the rate at which the bottleneck operation is producing, in order to avoid excessive inventory buildup. The schedule for succeeding operations, including assembly, should be derived accordingly. The scheduling of preceding operations should be derived backwards from the bottleneck schedule. We should ensure that the capacity of a bottleneck resource is not being wasted by allowing any excess slack in its schedule. At the same time, there should be a little slack in it to cover for variations in the market demand [8].

3. JUST-IN-TIME PHILOSOPHY

The dominance of the Western world in the area of manufacturing started with the Industrial Revolution in England and spread across Europe and America. However, in the last two decades, the world saw a shifting of industrial power from the West to the East. Countries like Japan, Korea, and Taiwan started gaining market share in the areas of smokestack industries and electrical appliances in the seventies. The eighties saw American slippage of market share in automotive and electronics. Initially the losses were blamed on low cost labor of competitors, and copying and dumping of Western products. In the mid-eighties, it was realized that the shift was not the result of trivial causes. It was because of an unprecedented race in all aspects of manufacturing [9].

Quality is one of the aspects of the race which can be used to understand its impact on the world market. Until 1970, the term "Yield" was used to measure quality. Yield meant how many good parts resulted from the input material. At that time, more than 10 percent parts were scrapped. During seventies, improvements in quality occurred, and bad parts were dropped below 10 percent. By that time, the

Japanese had reduced their quality defects below one percent. When the Western companies reached that target, the Japanese had already started talking in terms of parts per million (ppm). This means that the quality has increased many orders of magnitude in the last fifteen years. The new goal which the companies are trying to achieve is zero defects or 100 percent defect-free parts.

While the emphasis of the Western companies is on the utilization of sophisticated technology, the Japanese achieve the same goals by better utilization of human resources and the available equipment. The Western companies rely on complex computer programs to continually monitor the production systems, preplan activities and adjust production accordingly. Material Requirements Planning (MRP) and Manufacturing Resource Planning (MRPII) systems are heavily used in many Western companies. Although these systems bring about improvements, they are complex and expensive. The Japanese approach is to eliminate waste in every aspect of manufacturing. Excess inventory and unused capacity are examples of waste. This Japanese philosophy, which is called Just-In-Time (JIT), is becoming very popular in the whole world.

Toyota Production System is among the pioneers of JIT. The two pillars of JIT, according to Toyota Production

System, are Kanban system and Autonomation [17]. As mentioned in chapter one, 'Kanban' is a communication system which tells the subsequent processes to go to the preceding processes to withdraw parts and materials at the time needed and in the quantity needed. The company's production plan is given only to the final assembly line. The second pillar is 'Autonomation' (automation with human touch). It means the machines are taught to do what people can do, and in the case of defective parts, they are taught to stop automatically. As long as the machines function normally, no worker is required to attend them. Only when an abnormality occurs or a machine stops, there is a need for the presence of a worker. Thus, the first step towards automation should not be how a machine processes itself automatically, but rather how it detects abnormalities and stops automatically [17].

As discussed earlier, JIT philosophy is geared towards elimination of waste. According to JIT, the waste can be classified into various categories, such as:

- Waste from overproduction
- Waste from waiting
- Waste from transporting
- Waste from processing
- Waste from unnecessary stock on hand

- Waste from unnecessary motion
- Waste from defective goods [17].

1. Waste from overproduction

It is the most common sight and the worst kind of waste. Excess parts produced accumulate in between and at the end of the production line, thus creating unnecessary buffers. Overproduction also hides the workers waiting time. Other wastes give clue how to correct them, but overproduction provides a covering blanket and prevents from making corrections.

2. Waste from waiting

This type of waste is created when a worker stands idly near a machine as a watchman and cannot do anything constructive because the machine is running. It also happens if a worker cannot work because of the failure of a preceding process to deliver parts needed in time.

3. Waste from transporting

It is created when an item is moved a distance unnecessarily, either being stored temporarily or being rearranged. Normally, parts are moved from a large storage pallet to a smaller one to somewhere near the machine before they are finally processed. The Japanese companies bring parts to the machine only in the quantities needed, thereby eliminating all unnecessary intermediate steps.

4. Waste from processing

Sometimes a part does not function properly and the worker has to give extra attention to the operation. This causes a smooth processing to be interrupted and thus, valuable time is wasted.

5. Waste from unnecessary stock on hand

Unnecessary stock on hand is a direct result of overproducing. It has the risk of aging and obsolescence. It requires preemptive use of materials and use of storage areas to accommodate the excess products.

6. Waste from unnecessary motion

In Toyota Production System, if the worker has met the required quota, he/she is taught to sit idle so that the management knows there is excess manpower. This is also true of all other companies which have JIT systems. Sitting idle is not considered the fault of the worker, which is in direct contrast with the Western philosophy where the worker is blamed for wasting time. It is management's responsibility to identify excess manpower and to utilize it effectively.

7. Waste from defective goods

If a defective part is not detected and goes forward in a production line, then other work is done on it and raw material is wasted. Any work done on a product that does not add value rather decreases the necessary raw material and wastes worker's time is not an acceptable practice.

There are many ways of finding different wastes, but the most effective way is to translate such wastes into the waste arising from waiting. It is the easiest to detect, and provides the first step towards efficiency enhancement.

Many of the operations in the Western world are characterized by optimistic forecasts, generous lead times and vaguely defined procedures. If a company has excess stock on hand, it is blamed on lack of precision in forecasting product demand and variability in supplier lead time. JIT means a tighter control on all the aspects of an operation. Working towards JIT means rethinking and changing the way the things are done.

Steps should be taken to improve the forecasting procedures and suppliers should be told to meet the delivery deadlines. Suppliers are encouraged to learn JIT techniques and JIT experts are sent to the suppliers' plants to advise them how to cut variations in their plant operations. In the early years of JIT, there was a misconception that the manufacturing plants reduce their inventory stocks and force their suppliers to stockpile inventory for them. However, anyone who studies JIT, learns that under JIT philosophy, suppliers are considered partners instead of adversaries. Long term relations are established with reliable suppliers and efforts are made to reduce forecasting variation and inventory stocks not only in one's own plants but also in the suppliers' plants.

JIT philosophy can not be implemented overnight. All employees need to be educated about various stages of JIT implementation. In a customer-oriented company, the emphasis is on service. Inability to satisfy customer demand is considered to be the biggest evil. In a conventional Just-In-Case approach, the message conveyed is never to be caught without stock, and in the case of an uncertainty, order more [24]. One is blamed more for a stockout than having a huge inventory build-up. If a company wants to implement JIT, the above thinking has to give way to seeking ways of relating production activities more directly to actual requirements of customers rather than to build buffers. The above discussion again relays the message that inventory is bad and is used to hide problems.

Sometimes, if we have a problem that we cannot solve, we simply accept it and work around it. Working towards JIT means not giving up on problems. It means concentrating on lasting solutions even if it takes time. The 'searchlight' approach, where we focus intensely on one problem and then forget about it when we move on to highlight the next problem, is not acceptable in JIT. In the Western companies, lead times are set based on intense work studies and once set, they tend to remain unchanged for ever. The JIT approach starts with temporarily accepting the lead times presently in use and then concentrating efforts on reducing them. This is true not only for lead times, but for almost any production practice. In the same manner, it is essential that we should not assume that the environment in which a company operates is fixed. Just because a source of uncertainty is 'external' does not mean that nothing can be done to minimize it. In moving from traditional manufacturing towards JIT, the central objective is to remove uncertainties from the manufacturing system, by identifying the significant sources of uncertainty. To achieve this, it is essential to ensure that the feedback is not broken by the people who recognize the problem but believe that the cause is someone else's responsibility, and thus spend their time trying to isolate the problem's effects [10].

It has been said of the Western manufacturing approach that it tends to define where we have to go in a considerable detail and then go somewhere else. In a JIT system, we do not necessarily have a well defined end point, we are just happy to know that we are heading in the right direction. Thus what is possible is determined by how far we can go rather than how far we think we can go [7].

In the West, the motivation of the managers in a production system is such that they have different priorities, and thus appear as moving in different directions. The financial manager wants to reduce inventory to save money. The production manager wants to increase inventory to avoid stockouts. The sales manager makes delivery promises to customers and then presses manufacturing to meet the promises. Each department appears to have separate goals; nobody wants to look at the whole picture. Success in a JIT system results if everyone starts considering the importance of company's goals rather than individual departmental goals.

JIT is a philosophy of common sense. It is based on the concept of continuous improvement. There is no fixed target in JIT that a company has to achieve. Targets are set and efforts are made to achieve them. Once achieved, new targets are set. Thus, it is an endless process of continuous improvement. It is not just limited to manufacturing. It also encompasses supply, delivery, quality and systems aspects of a business. In this chapter, the JIT aspects of supply and delivery will be discussed briefly, and then the JIT approach in manufacturing will be discussed.

3.1 Just-In-Time Supply and Delivery

One cannot expect to implement a JIT system in a production facility without involving its suppliers and customers. Working with suppliers means making efforts to eliminate the uncertainty which surrounds supply. Research shows that in Western industries, material costs account for 51 percent of total costs as compared to 15 percent of labor costs [24]. Many companies are making investments in automation and robotics which will reduce labor costs even further. On the other hand, companies have only recently started examining ways to reduce material costs. In a manufacturing environment, there are several operations which need hundreds of purchased parts. Going through all these parts to place orders is both boring and time consuming. Thus, it is often done poorly. Since the suppliers are convenient scapegoats, it is easier to blame them even if the fault is our own. It is important that if we need to move towards JIT, the purchasing department should be in good working condition. It should order only what is needed, and follow up any shortages that occur.

Some of the factors which should be considered when including suppliers and customers in a JIT implementation are:

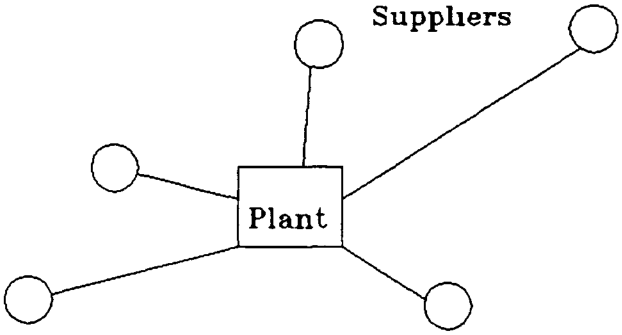
- Link with suppliers
- Multi sources vs Single source
- Short term vs Long term agreement
- Local vs Distant suppliers
- Link with customers [24].

1. Link with suppliers

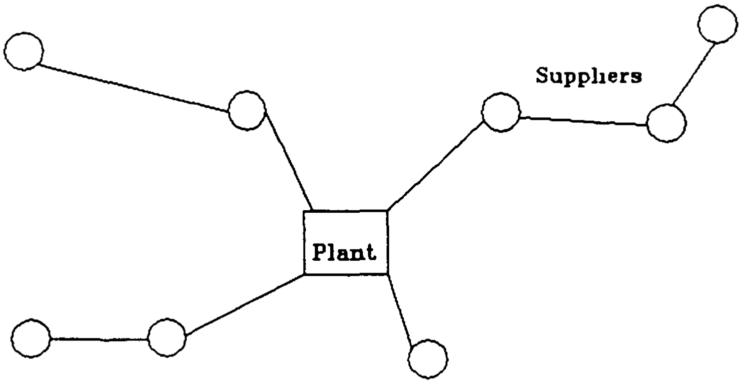
JIT philosophy stresses on low inventory and short lead times. With supplier links, one way of reducing inventory is to reduce the order quantities. But if each delivery has the same amount of paper work, it will increase with more frequent deliveries. One way of reducing paperwork is to order once per period but requiring shipments more frequently. More deliveries also lead to higher freight costs. To reduce the cost of shipping smaller volumes, a rim system of deliveries can be used, instead of the spoke system where each supplier delivers directly to the plant (see Figure 3.1). In rim system, suppliers take turns to deliver to the plant, picking up deliveries from other suppliers on the way. Large volume suppliers can keep their direct links if they want. The success of the rim system of deliveries needs some organization, but it does lower the cost of shipment.

Some important characteristics of JIT supplier links are high quality levels, reduced order quantities, and reliable lead times. If the above conditions hold true, the delivered parts can directly go to the shop floor avoiding inspections and expediting of materials. Suppliers can be classified into categories depending on their quality and delivery performances. Only the suppliers who score high in both categories should be considered for long term partnership.

In choosing a supplier, we make the decision based on the total cost not just the material cost. Total cost includes material cost, order cost, expediting cost, receiving inspection cost, and freight cost. When all the costs are taken into account, it may be true that a supplier



(a) Spoke system of deliveries



(b) Rim system of deliveries

Figure 3.1: Spoke and Rim system of deliveries

with a higher purchase cost has the lowest total cost, because costs caused by late or poor quality deliveries can be substantial [24].

2. Multi sources vs Single source

Traditionally companies order the same item from several suppliers to keep prices competitive, and to guarantee that alternative channels are kept open in case one fails. But they ignore the fact that ordering large quantities from one supplier results in cost reduction because of economies of scale. It also justifies suppliers' investment in process improvement, and reduces managerial problems associated with dealing with several suppliers. Frequently, companies implementing JIT meet with their suppliers to discuss any problems. The companies exchange task teams to study each other's manufacturing processes and to recommend improvements.

It is extremely improvement to move slowly and carefully towards single source supply. Selecting a single source just because the buyer did not have time to investigate alternative sources has no virtue. It may even lead to disastrous results. When making a commitment to a single supplier, we should be certain that the potential supplier is capable of meeting the required standards in terms of quality, service and cost.

3. Short term vs Long term agreement

JIT encourages long term cooperative relationships with a few carefully selected suppliers for the following reasons:

- more reliable deliveries
- investments for process improvement
- better quality products
- lower cost

Long term relationship gives the supplier a greater sense of security, allowing them to achieve a closer balance between the work load and capacity and makes the investment in new manufacturing machinery easier to justify. Again, the decision to enter into a long term agreement with a supplier should be made after careful thought and consideration. Companies which have successfully implemented the JIT philosophy with their suppliers have moved slowly and steadily to single source, high volume suppliers [24].

4. Local vs Distant suppliers

In the past, companies thought that the shipping cost of material had only a small impact on buying decisions. Now they realize that the true cost of moving materials over long distances must include hidden factors, such as long lead time, huge inventories, etc. Each day added to the manufacturing lead time for transportation extends the planning horizon. It also increases uncertainty in production planning. Local suppliers also reduce the risk of a large defective delivery, and therefore reduce waste through inventory associated with the delivery time. For the above reasons, the companies are trying increasingly to find local suppliers to gain the full benefits of JIT manufacturing [10].

5. Link with customers

The above discussion holds true in this case except that the roles are changed. The company is now a supplier trying to fulfill customers demand. It is important to include customers in a JIT implementation because consistency of their orders can ease a lot of planning problems. Efforts should be made to educate customers about JIT. They should be guaranteed reliable deliveries and high quality parts if they follow a firm schedule. The overall aim of building relationships with suppliers and customers is to improve the response of the JIT system to changes in market requirements.

3.2 Just-In-Time Quality

One of the goals of JIT is to make products of high quality. No matter how cheaply a product is made, if it can not be translated into money, the end result is a loss. Traditionally, the Western companies have a quality control department separate from the production department. Many quality problems are caused by what the production people do or fail to do [7]. The separation of inspection causes a time delay between a drift in quality and its detection, resulting in a considerable amount of unsatisfactory output being produced before the process is corrected. This time delay also makes it difficult to detect the actual cause of the problem, and hence its remedy. The only solution remaining is to rework the work pieces or scrap them. Now, if the rework is done in a separate area, it results in excess manpower and excess time required for sequencing expediting rework. It also gets separated from its batch and a decision has to be made whether to hold up the whole batch or to let remainder of the batch to proceed some pieces short. This causes further problems for inventory control system. To prevent the shortages occurring in the system, companies are forced to hold excess safety stocks and to operate the inventory system with an excessively large scrap allowance.

It is important to have a rapid feedback of information from inspection area to workers responsible for actual operation so that any variation in the quality of their output should be remedied quickly. The easiest way to overcome the above problem is to make workers responsible for inspecting the pieces they produce, and if the pieces need rework, let them carry it out themselves. By combining the responsibility of inspection with production, the workers feel more responsible for the quality level of their work. It also has the added benefit of immediate feedback, and thus immediate correction and gain of worker's involvement. Employees' participation can be obtained through positive reinforcement. As B. F. Skinner (a renowned psychologist) puts it, the way the reinforcement is carried out is more important than the amount. It should be specific and immediate [27].

At Toyota Production System, their slogan is, "Catch the defective in its act." All workers check their own work and inspect every piece that passes in front of them. If a subsequent process discovers a defective part, it immediately tells it to the preceding process. All the rework is done by the workers at the process responsible for it. If a worker finds a defect which needs everyone's attention, he/she presses a button to stop the whole line. This way everyone knows what the problem is and the supervisor takes necessary measures to remedy it. If a worker cannot keep up with the pace, he is taught to stop the line so that help arrives. In no circumstances he is supposed to work faster than his normal rate, as it increases chances to produce a defective part. Another measure Toyota uses to ensure stable and high quality is the process of 'foolproofing.' It is a means to create devices which discovers disorders without the worker's attention [17].

Companies interested in JIT implementation should incorporate the above measures in their system. No level of scrap should be considered acceptable. The management and the workers should always look for improvements [10]. By stating expected level of scrap for particular processes, it is easy for those figures to become regarded as acceptable levels. The charts which leads towards continuous improvements in capacity and accuracy should be preferred over charts showing a concept of acceptable quality levels. The employees should be taught the importance of 'line-stop.' Short sighted managers are tempted to allow quality levels to slip by their unwillingness to accept the line stop mentality. They argue that if they stop the line whenever there is a quality problem, they will never be able to ship anything out. This is the mentality that needs to be changed if JIT is to be implemented.

3.3 Just-In-Time Manufacturing

JIT is most commonly associated with the activity of manufacturing. An essential feature of JIT is concerned with physical changes to the manufacturing processes which increase work flow. If the manufacturing processes are not changed, then it becomes extremely difficult to achieve JIT production. Some of the techniques which are used to achieve JIT manufacturing are:

1. Reducing setup times
2. Optimizing plant configuration
3. Preventive maintenance
4. Pull system of production [7,24]

3.3.1 Reducing setup times

Setup time is the time taken to change a machine so that it can process another type of product. Until recently, very little attention was paid to reducing setup time. Economic order quantities (EOQ) were used for ordering purposes. The trend was to keep setup cost constant and concentrate on reducing unit factory cost (denominator), thus increasing the lot size. The EOQ type formulae can mislead managers who believe that by using the formula to determine lot sizes, they are obtaining the optimum batch size. In fact, the lot size obtained this way is optimum only according to the assumptions behind the formula; one of them being a fixed setup time. Excessive setup time is harmful for two reasons. First, time spent in setting up a machine is non-productive, thus reducing efficiency of the machine. Second, longer setup time results in larger batch sizes because with long setup time it is not economical to produce small batches.

Reducing setup time results in increased machine efficiency, decreased batch sizes, decreased inventory levels and decreased lead times. Small batch sizes imply frequent runs and a levelling of production activities. Lower inventory levels result in less capital being tied up in inventory and also reduces the risk of obsolescence. Reduced lead times mean quick response to market demand and speedy delivery to customers. However, small batches are economically possible only if the time taken to setup the machinery is small in proportion to the time taken to process the batch. In the 1940s, Toyota's die changes took two to three hours. By the late 1960s, it was down to a mere 3 minutes [25]. Shigeo Shingo developed the quick die change method and single minute exchange of die (SMED) concept for Toyota in accordance with the setup reduction goal [34].

Reduction of inventory levels points out the unreliable machines. If a machine breaks down, following machines quickly become starved of work. In order to avoid this disaster, JIT implementation includes an extensive preventive maintenance program to help ensure high process reliability. The idea of preventive maintenance stresses the need of reducing inventory stocks in good times to expose problems and to fix them now before they cause trouble later.

In the Western companies, costing systems are based on 'fixed' setup times and variation in setup time may not appear favorable to the accounting personnel. Bonuses often work against any attempts to reduce them. In order to reduce setup time, steps should be taken to structure the operation in a way that it appears as something which people want to achieve. Most setup times can be reduced by 75 percent [13]. Many case studies of dramatic setup time reduction are available in literature [11,32,33].

If a setup time is long, it makes sense to do a setup operation if there is enough work to justify it. This results in increased lot size, reduced flexibility, and increased inventory levels. Time required to respond to a new technological change is also increased, thus increasing the risk of obsolescence.

In an effort to reduce setup time, it is better to pick a few setups, and focus attention to reduce them. If we begin by attempting to reduce all setup times simultaneously, we will not be able to achieve any goal.

Low setup time can be achieved by the following:

- Eliminate external setup time
- Modify speed of internal setup
- Eliminate adjustment process
- Eliminate the setup step itself

1. Eliminate external setup time

It involves identifying steps that require machine to be stopped (internal setup) and those that can be done before the machine is stopped or after it begins operating again (external setup). Examples of external setup are obtaining tooling from the tool shop or adjusting a fixture while the machine is running. Internal setup is changing a tool or fixture on a machine, in which case the machine should be stopped. Steps should be taken to eliminate external setup. Putting tools needed on the machine closer to it will reduce the time required to look for them in the tool shop. Later, an effort should be made to convert as much internal setup to external, and again to eliminate it.

2. Modify speed of internal setup

While the machine is idle, steps can be taken to preheat dies, or color code tools to assist in locating them for particular purposes. The Japanese are known for modifying general purpose machines for specific applications to speed up internal setup [7]. Most of these modifications are small in cost but significant in effect.

3. Eliminate adjustment process

Adjustments are the cause for more delays and poor parts than any other single item. The principle behind this step is to change a continuously variable adjustment into a small number of discrete steps. To eliminate adjustment, dedicated tooling and automatic die positioning can be used.

4. Eliminate the setup step itself

Abolishing the setup step itself is the final setup reduction concept. Parts should be standardized to reduce the product range. Each part then can be used on a wide variety of products, thereby reducing the setup time.

Several techniques can be used to implement the above four steps [21].

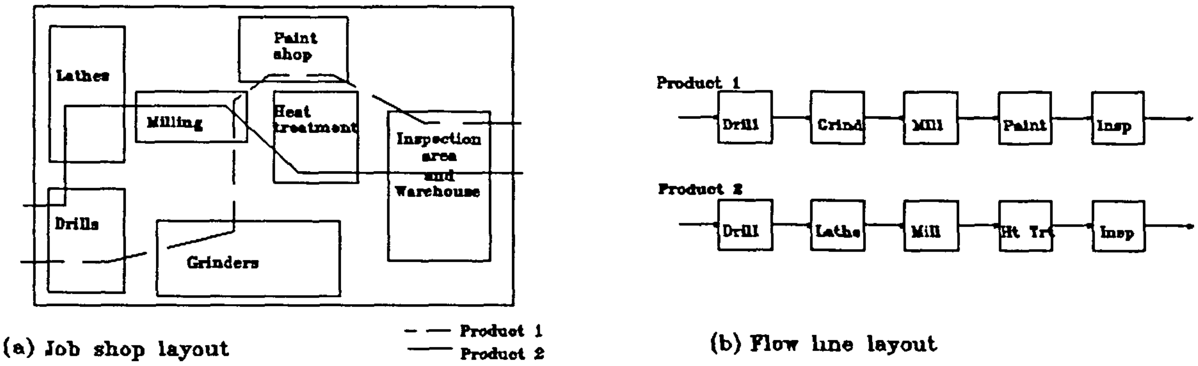
- If external setup cannot be eliminated, standardize it and record it for future reference.
- Standardize only the equipment that the machine needs.
- Use quick fasteners which need a quarter turn to fasten, as compared to traditional nuts and bolts.
- Use a supplementary tool if the part alone is difficult to handle.
- Use parallel operators if a machine requires a long setup time.
- Use a mechanical setup especially for heavier setups.
- Use dedicated machinery. If possible, buy smaller inexpensive machines and permanently configure them for a specific use.

3.3.2 Optimizing plant configuration

The conventional classification of production systems splits them into four categories: continuous flow, mass production, intermittent batch production, and job shop [10]. Continuous production is characterized as having a plant which is setup to produce a particular product continuously, e.g., petrochemicals. In mass production facilities, products are manufactured as discrete units instead of the continuous stream of the product. Batch production results when the numbers of each item produced are too low and volumes of items are insufficient to justify setting up a flow process. In a job shop, a job typically goes through several operations and spends most of its time sitting on the floor or in the store between operations. In practice, many manufacturing systems are hybrids, using mass production methods in some areas and batch manufacturing in others.

As mentioned earlier, one of the principles of JIT is the constant effort to strive for simplicity. One way of achieving this is to rearrange the factory floor from complex routes followed in a job shop towards a product layout using flow lines (Figure 3.2). The Japanese use flexible workforce to achieve this goal. When demand is high, each machine has one or more workers to operate it. During low demand, each worker operates more than one machines. When there is no demand for a particular product, workers are reallocated to another flow line.

Not all flow lines are equally desirable. In a 'Birdcage' layout, a worker operates three or more machines of the same kind (Figure 3.3). This restricts the use of flexible labor, because there are no options for restructuring the use of workers. It also makes



(a) Job shop layout

(b) Flow line layout

Figure 3.2: Job shop to Flow cell layout

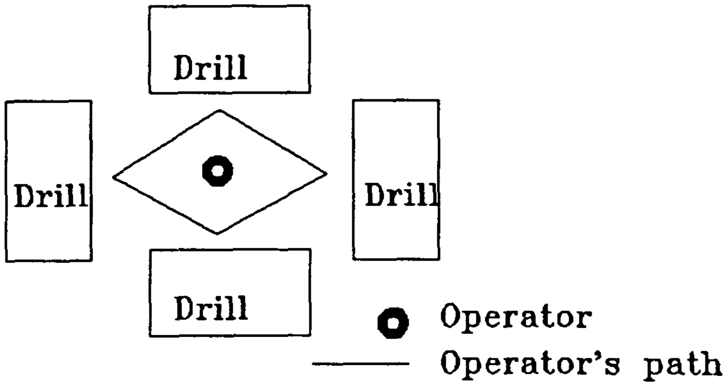


Figure 3.3: Birdcage layout

synchronization between work stations difficult, thus resulting in an increase in WIP levels. 'Remote Island' layout has one worker operating three or more machines of different kinds in a closed layout (Figure 3.4). In these mini flow lines, a stock of products produced by the workers is created because it can not easily be transported to the next location. Also, because of the small size, it becomes difficult to adjust the number of workers. The layout which has gained considerable attention by the JIT implementers is U-shaped flow line dedicated to a particular product family (Figure 3.5). Its advantages over a linear flow layout are that it assists communication and cooperation among workers, because they are physically closer. They can tell each other of quality problems arising in the layout and action can be taken quickly. Also, it allows workers to be physically closer to more machines, as compared to a linear flow line, and thus to operate more machines.

3.3.3 Preventive maintenance

Setup time reduction results in a decrease in the amount of time a machine is not running. To further reduce the amount of downtime, number of machine breakdowns must be cut. A successful JIT implementation reduces work-in-process (WIP) and inventory to a minimum, thus making

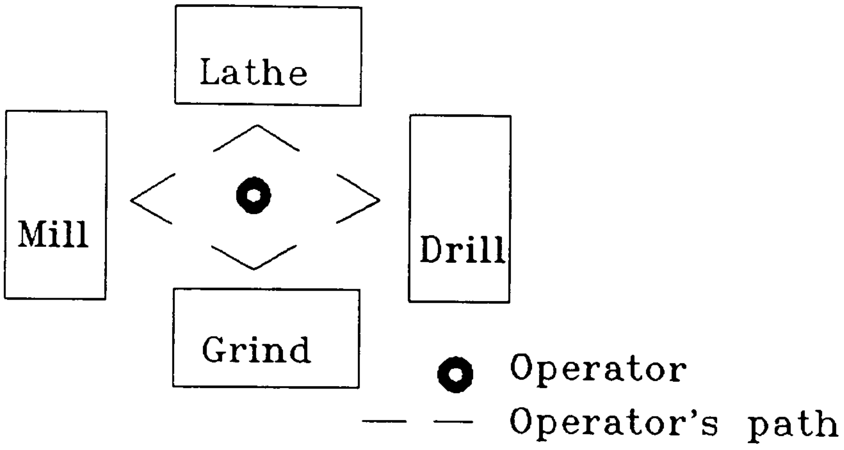


Figure 3.4: Remote island layout

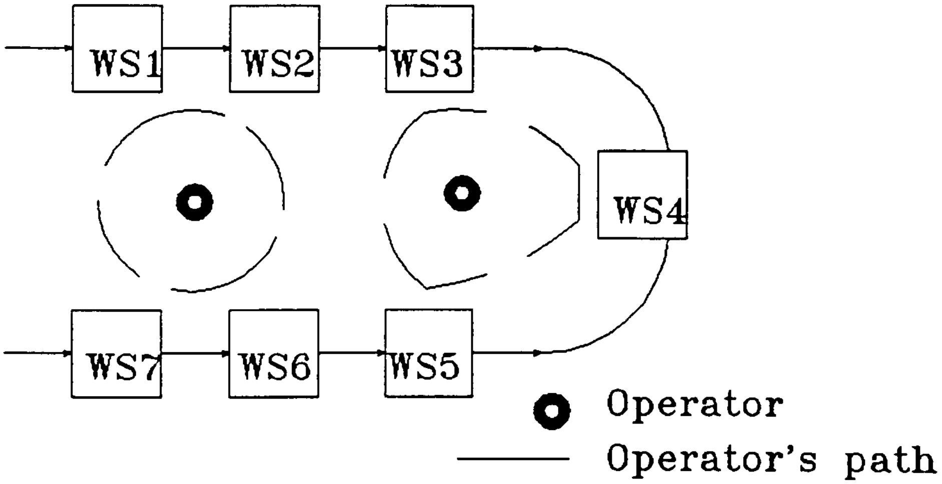


Figure 3.5: U-shaped flow line

manufacturing system more vulnerable to breakdowns. In the case of a machine breakdown, subsequent machines become starved of work; whereas without JIT, there is bigger buffer stock allowing more time to repair the machine. JIT systems rely on preventive maintenance programs with the aim of preventing breakdowns rather than repairing them once they occur.

The treatment of machine breakdown in a JIT system is a classical example of the inventory river having problems (rocks) hidden under the covering blanket of high inventory levels (the level of river). Total Preventive Maintenance (TPM) is concerned with solving the breakdown problems (removing the rocks) in order to have a smooth flowing production line with a minimum of inventory and work-in-process. The aim in the JIT system is not only to lower WIP levels and manufacturing lead times but also to identify problems before or, as soon as possible, after they occur and to force managers to take remedial actions. If we have a bottleneck operation, then traditional scheduling may alleviate the symptoms but it will never remove the problem. Actually traditional systems simply work around the problem at the cost of keeping extra WIP or rescheduling work into other less efficient processes. In a JIT system, buffer stocks are reduced so much that, in a sense, all machines are bottlenecks and a breakdown will reduce effective utilization of equipment.

In a TPM program, the management is encouraged to decentralize maintenance. Operators are asked to perform relatively simple but essential tasks of daily maintenance, such as, adding lubricant in machines if needed and checking for wear and tear. Operators are the ones who know most about the operations of their machines and are the best source to detect anything unusual. Also, the above arrangement gives the operators a sense of responsibility and ownership of their machines and they feel pride in keeping them running trouble free. Major maintenance jobs are still to be done by the maintenance department. A comprehensive educational program is required for the successful implementation of TPM. This normally involves the maintenance department training operators in the daily maintenance procedures.

Routine maintenance can usually be accommodated in normal production runs during slack times. For major maintenance, companies running one or two shifts per day can do it during the non-productive shift(s). For three shift operation, the companies have to make some non-production time available to its maintenance staff. Maintenance could be done on weekends or holidays, but that means some maintenance becomes overdue before the next weekend/holiday. The point is that running production 24 hours a day if a machine is down eight hours or if it produces poor quality parts for eight hours does not make any sense. It is important to have some non-production time for TPM so that production time results in high quality parts with little machine downtime.

3.3.4 Pull system of production

Traditional Western approach to production is to 'push' materials through successive operations starting from the first process. This is done in accordance with a pre-determined plan. JIT approach is to 'pull' work through the factory to meet customers demand. Production is planned only when a definite requirement arises.

MRP is a 'push' system because it plans what is to be produced, which is then pushed through the factory. Bottlenecks and other problems are assumed to be detected beforehand and complex monitoring systems are installed to take corrective actions. By contrast, the JIT 'pull' system eliminates the need for complex data flows, and sends simple signals to monitor work flow. When work is taken from the last operation, a signal is sent to the preceding operation to produce more. If no work is taken from the end operation, no signals are sent and hence no work is done. During slack periods, operators do other jobs such as cleaning the machine, performing preventive maintenance or attending educational sessions arranged by the management. The pull system approach ensures that production does not exceed immediate needs, thus reducing WIP and cutting manufacturing lead times.

The Japanese use the term 'Kanban' to explain the simple control mechanism they use. As mentioned in chapter one, kanban means card in Japanese. There are several variations to the kanban approach, but the basic idea remains the same. A kanban is a piece of paper, board or metal used as notification to manufacture more parts. Each kanban is an authority to produce only a fixed quantity of a particular item. Toyota and its suppliers use the double kanban system, using withdrawal and production kanbans, which gives greater control but is more complex. That is why most of the other companies use the single kanban system.

A kanban is sent before a part is needed not when we need it, otherwise some time has to be spent waiting for the part to arrive. Thus, a small working buffer has to be kept - usually one kanban, but sometimes more. The Japanese approach to find out the necessary number of kanbans is very interesting. They start with a generous number of kanbans. If no problems are encountered, they take out one. In the same way, they keep on removing kanbans one by one until problems appear. They then analyze the problem to see if it can be solved by applying the JIT principles. If that does not work, they put back one kanban, and this determines the minimum necessary number of kanbans [7].

In theory, a push system like MRP should be able to operate with minimum level of inventory, but in practice this is rarely seen because of the problems involved in keeping to the production plan. Thus, an MRP system is often characterized as one having large inventory stocks and increased lead times. However, an MRP system can be used to aid a pull system to speed up feedback of information, if long lead times are encountered at an intermediate operation in a pull system. MRP system can also be used to coordinate among a large number of shops, some or all of which may be JIT oriented. The MRP system can integrate the activities of the shops ensuring that enough supplies (raw materials, components) are available. In both the above cases, the MRP system is not used for detailed control of shop activities. It is used selectively for global coordination [24].

4. MODEL DESCRIPTION AND METHODOLOGY

To study the effect of bottleneck operations on a production floor, a mini product cell and a job shop layout were studied for the same number of products. The other two variations simulated were lowering setup times of the mini cell and adding a capacity constrained resource (CCR) to the mini cell. A plant trip was arranged with Fisher Controls International, Marshalltown, Iowa, and real life data were requested from the management.

Fisher controls makes a wide range of control valves, industrial regulators, liquid level controllers and electronic instrumentation. It operates in a job shop environment, but steps are being undertaken to move towards JIT. The shop floor is divided into various sections, with each section having machines of one type. Batches move through the shop based on the operation needed. There is an excessive amount of WIP storage needed to keep machines running close to their capacity. To make matters even more complicated, Fisher Controls has two manufacturing facilities in Marshalltown. In some cases, some operations on a batch are done in one facility and others are done in the other facility. This all adds to large lot sizes to keep cost of setup time per part low.

In the early eighties, an attempt was made to move towards group technology (GT) concepts. In GT, parts are grouped in product families according to the similarities in tooling requirements, design specifications and operation description. Based on the similar manufacturing processes, small cell layouts are set up in the factory. Each cell is dedicated to making a certain family of parts. This first attempt was unsuccessful partially because the management could not properly cross train the workers - an important prerequisite of GT and JIT.

Another attempt was initiated in the mid eighties to move towards JIT. This time, education and commitment of employees were strongly emphasized. Parts were again assigned to different group of products. Machines required for different group of families were moved closer to each other to form mini product cell layouts. In some cases, where moving machines closer was not possible due to physical and economical constraints, some machines were assigned to a particular family of products to form virtual cells. The machines were dedicated to performing operations needed to produce parts within a product family, thus reducing setup time considerably. Also, the material movement was minimized and inventory was reduced. After successful experimentation with a few cells, Fisher Controls is planning to rearrange the whole layout in the form of cells. Steps are being undertaken to avoid WIP movement from one facility to another by combining all operations required for one part in one facility.

The product family studied in the simulation study is "Bonnet." There is a large variation in size and weight of different bonnets. Routing sheets of different bonnets were obtained from Fisher Controls. The sheets have the information both for the job shop and the mini product cell cases. All bonnets share the same cell, which has six machines in it. The machines are as follows:

Code	Machine Description
0188	5A Warner and Swasey Manual Turret Lathe (Lathe 1)
0293	2AC Warner and Swasey Semi-automatic Turret Lathe (Lathe 2)
0295	4AC Warner and Swasey Semiautomatic Turret Lathe (Lathe 3)
0469	Multi Spindle Drill (Drill 1)
0422	2BH Burgmaster Turret Drill (Drill 2)
0431	4' 13" Carlton Radial Drill (Drill 3)
4003	"Black box" number for all machines in the cell

Machine code 4003 shows the total setup time when all machines are combined in a cell. Names of machines in parenthesis are the ones used in the model for simplicity. The routing sheets obtained from Fisher Controls are further classified into three subgroups. One subgroup uses four out of six machines, the second one uses five, and the third uses three machines (Figure 4.1). After being processed in the cell, all bonnets go to adjacent wash and debur area where the final inspection is also done. To keep the model simple, one part was picked from each subgroup for the simulation.

In the job shop environment, each bonnet operation had a large setup and teardown time. Set up time was large because each machine also had to be available to other products requiring different set of tools, besides bonnets. In the mini cell case, setup time was reduced by using quick fasteners and quarter turn clamps. Employees were encouraged to make suggestions, and some of the suggestions were extremely useful in reducing the setup time.

Product A:
- Machine # 1: Lathe 2
- Machine # 2: Lathe 3
- Machine # 3: Drill 1
- Machine # 4: Drill 2

Product B:
- Machine # 1: Drill 3
- Machine # 2: Lathe 1
- Machine # 3: Lathe 3
- Machine # 4: Drill 1
- Machine # 5: Drill 2

Product C:
- Machine # 1: Lathe 2
- Machine # 2: Drill 1
- Machine # 3: Drill 2

Figure 4.1: Sequence of machines for the three products

4.1 Model Assumptions

4.1.1 Mini flow cell

1. One mini flow cell having six machines is studied. Processing times from three different bonnet parts are used in the study.

2. Since only a subset of parts are used in the study, it is difficult to find the exact arrival time of orders. After discussing it with the Fisher Controls management, exponential distribution is used for the arrival time.

3. No machine breakdown occurs during a production run. Preventive maintenance is done during non-production time to keep the machines running trouble free.

4. No quality problems are encountered. Rework, if needed, is done in the cell. Processing times include time required for rework.

5. Produced parts are immediately withdrawn by the warehouse staff.

6. The cell under consideration and wash/debur area are adjacent to each other and transfer time of batches between the two areas is negligible.

7. Simulation is done by using two rules. FIFO (First-In, First-Out) and SORTED scheduling. In FIFO, batch which has been in the queue the longest is processed first. SORTED

scheduling lets the user choose the next batch from the queue based on a priority value that the user assigns. LIFO (Last-In, First-Out) scheduling is not used because it is not applicable in this manufacturing environment.

4.1.2 Job shop

1. The same three type of parts are studied in this case.
2. Each operation is performed in a different work station, so each part goes through many work stations to be processed.
3. Even though waiting time is considerable at each work station before a particular batch is processed, each work station is assumed to be available for the parts in consideration when needed.
4. Large batches are used to compensate for long setup time.
5. Transportation time is also not considered in the simulation.

4.1.3 Modified mini cells

1. For lower setup cell, a setup time decreased by 20 to 25 percent for the three processes is used as model input.
2. For additional CCR cell, two lathes of type 2 are used in the simulation scenario because for two of the three

processes, lathe 2 has the longest processing time.

3. All other model inputs of regular mini cell remain valid for these two cases.

4.2 Simulation Model Description

A model is designed by using a microcomputer based simulation package called MicroSAINT. The simulation used in the model is discrete because it is 'event driven' as compared to 'clock driven' or continuous. Discrete models are based on the assumption that anything significant happens only at the beginning or at the end of an event.

MicroSAINT was chosen because of the following reasons. It was available in the department, and its availability on a microcomputer was considered an additional convenience. It allows the statistical results generated in the simulation to be imported to other packages, such as Lotus 1-2-3 for further analysis. It also has an interactive debugger which makes debugging easier [16].

4.2.1 MicroSAINT: general information

MicroSAINT is a menu-driven general purpose simulation package. A model is built by filling in blanks on menus. Each event or task (as called in the software) is represented in the model by an oval node [39]. A network, represented by a rectangle, can be a group of sub-networks or a combination of tasks and sub-networks. MicroSAINT calls its model development as "Task Network Modelling." This concept involves breaking down of an activity or process into a series of sub-activities or tasks. The advantage of constructing network models is that it is often easier to describe small parts of an activity as compared to describing it as a whole. Each task has associated parameters which depend on what one is interested in studying. Each task has an associated menu with it (Figure 4.2). The user has to fill out the menu options. A brief discussion of each menu option is given below.

1. Task name: A user can assign any name to a task to recognize its purpose. The name can be up to twenty characters long, and can include spaces and special characters. The default is "unnamed." The name is only for the user's sake. MicroSAINT recognizes tasks only by the task number.

2. Type: It can be either task (one job) or network (collection of jobs).

3. Upper Network: It signifies the network in which the task belongs. The starting network or top network is always numbered zero, and contains all of the tasks and sub-

```
Task Number:  7.1
(1)  Name:  Setup Time, A
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  1;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  60;
(7)  Standard Deviation:  5;
(8)  Task's Beginning Effect:
(9)  Task's Ending Effect:  num1=0;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  7.2           Parts    (12)  1;
(13)                         (14)
(15)                         (16)
```

Figure 4.2: MicroSAINT task menu

networks in the model.

4. Release Condition: It prevents a task from executing until a certain condition has been met. A task cannot begin unless its release condition is non-zero. By default, the release condition is set to one so that the task can begin as soon as it is scheduled.

5. Time Distribution Type: Task times are performed according to the task distribution type. Options include exponential, gamma, normal, rectangular and user-defined functions.

6. Mean Time: It defines the average amount of time required to perform the task.

7. Standard Deviation: It shows standard deviation of the task.

8. Task Beginning Effect: It defines how the system changes as a task begins execution. Its menu lets the user develop one or more expressions that change values of variables in a model as the task begins.

9. Task Ending Effect: It defines how the system changes after a task has been executed.

10. Decision Type: It defines which job is executed next at the completion of the present job. The available choices are:

a. Single: Only one choice of job to be executed next.

b. Probabilistic: Begins one of the several listed jobs based on a probabilistic branch.

c. Tactical: Begins one of the several jobs based on the value of its associated expression. The expression that calculates to the highest value indicates the next task to be executed.

d. Multiple: Begins several tasks at the completion of the present job.

e. Last: Last job of the model. No job begins upon completion of this task [20].

In order to identify each entity going through the model, the software uses a system variable called "tag." As an entity enters a model, it is assigned a tag number which remains with it throughout the model. Another important feature to note is that each statement has to end with a semicolon. If the user forgets the ending semicolon, the software prompts to add the semicolon at the end of the command.

4.2.2 MicroSAINT: the model

The model consists of twenty seven tasks with task 2 (Order Arrives) as its first task and task 15 (End) as its last task (Figure 4.3). For the first task (task 2), an

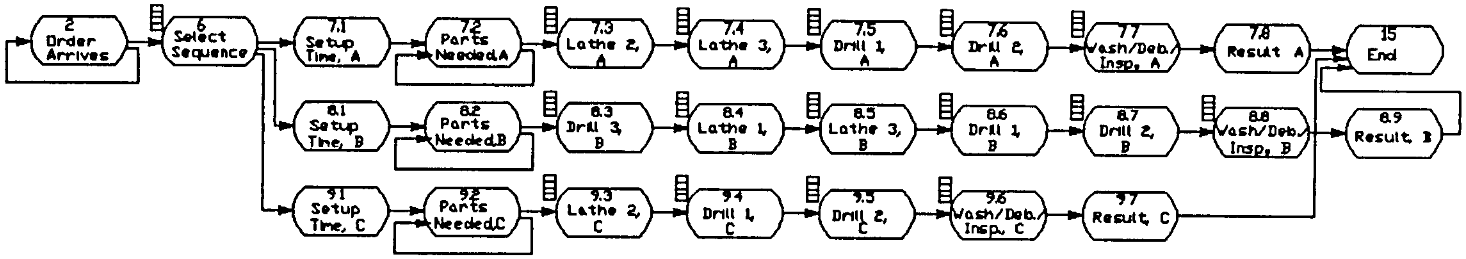


Figure 4.3: MicroSAINT model network diagram

exponential distribution with a mean time of 200 minutes is used. Each time a new order is generated, its tag value is incremented by one to differentiate it from other orders. It also passes through a selection procedure where each order is identified as being one of the three products (product A, B, or C). The selection is done through the use of a function called CHOICE.

The next task (task 6) sends the order ahead when the machines in the model become available. Once an order starts processing, other orders are stopped from getting in through the use of a release condition and the statements in the task beginning effects. A variable 'full' is used to control the release of orders. The value of 'full' is assigned an initial value of zero. In the task beginning effect, value of 'full' is increased by one, thus stopping other orders from getting in. A tactical distribution is used to redirect an order to its proper route. An order is sent to task 7.1, 8.1 or 9.1, depending on if it is product A, B, or C respectively. In either case, it passes through the 'setup time task' and the 'parts needed task' where the number of required parts in a batch are produced. Each product has to process different size of batch.

To identify separate entities within batches, tags associated with the three products, namely A, B, and C, start with 100, 1000, and 5000 respectively. Generation of number of parts required in a batch is done by looping back to the same task until the required number of parts is generated. Variables num1, num2, and num3 are used for batch sizes for product A, B, and C respectively. In the task ending effect of the parts needed task, an analysis is done to see if a new batch is about to start or an old batch is still being processed. The order then passes through the given sequence of machines. When all processing is done, it passes through a wash, debur and inspect station, and then moves on to results task (task 7.8 for product A, 8.9 for product B, and 9.7 for product C). Here the variable 'full' is decremented by one, so that its value becomes zero again and the next task can be released at task 6 (select sequence). Also, total number of batches processed so far are recorded in this task. Task 15 (End) is the last task with which the processing of a particular order ends.

The simulation of entities waiting in line is displayed through the use of queues. In MicroSAINT, queues are shown as rectangular boxes in front of a task. Queues can be ordered as first-in first-out (FIFO), last-in first-out (LIFO), or can be sorted based on a priority. In this model, queues are used to gather data in front of each machine and wash/debur area. For convenience, the name of a queue in front of a machine has the same number with the addition of a 'q', i.e., queue in front of task number 7.4 (Lathe 3) is 7.4q.

The one-time assignment of variables is controlled through the use of simulation scenario. In the case of regular mini cell layout, all machines used in the model are assigned a value of one, indicating that one machine of each type is available. When a CCR is added to the model, the value for lathe2 is changed from one to two, indicating that now two lathes of type 2 are available in the model. Also, duration of simulation is controlled in the simulation scenario. A variable 'simtime' is used to assign the duration for which the simulation should run. In this case, the simulation was run for four weeks (9600 minutes), assuming that each week has five working days and each day has 480 minutes.

5. MODEL EXECUTION AND ANALYSIS

The model described in the previous chapter was executed with different set of inputs. It was run both for the mini cell layout and the job shop environment. In the cell layout, it was run in three variations. First, it was run as the model which was designed for the study. Then, it was executed by reducing its setup times by 20 to 25 percent. The last time, it was executed with the addition of a machine in the bottleneck process. For two of the three products, lathe 2 had the longest processing time, and thus it was selected as our bottleneck.

The idea behind running the model in different variations was to find out how much was the variation between running a process both in a job shop mode and in the cell layout mode. Even though the setup times were high in a job shop, they were compensated by running large batches through the shop. In JIT, we are more concerned with reducing setup times so that we can gain the benefits of reduced inventory, small batches, and faster response to customers' needs. Before running the simulation, it looked as if the cell layout production would have a greater throughput and less queue buildup because of the huge difference in the setup times.

5.1 Model Analysis

The data obtained from Fisher Controls were used as input to the model. A parent model was designed in MicroSAINT and changes were made in its input to conform it to the four cases. Each case was run for one month (four weeks) of simulation time, but results were collected at frequent intervals to assist in a detailed study. The processing times used in the model are given in Table 5.1. Each case was run 30 times and the results were collected for further analysis. The results were obtained for weekly production runs for the four cases using FIFO and SORTED priority scheduling. The results are given in Tables 5.2 and 5.3.

For the FIFO scheduling, the production results for three cell layouts were almost similar for one week of production. The production of product A was close to 10, of product B was 32, and of product C was 52. For the job shop, the production of product A was 20, of product B was 24, and of product C was 44. The variations in production become more obvious as we move to bigger production periods. The two variations of cell layout (lower setup cell and

Table 5.1: Processing times used in the model

Product	Operations	Mean	Std. Dev.
Product A	Lathe 2	11.15	2.00
	Lathe 3	8.00	0.14
	Drill 1	1.75	0.00
	Drill 2	1.50	0.00
	Wash/Debur	0.55	0.00
Product B	Drill 3	3.41	0.00
	Lathe 1	12.58	0.00
	Lathe 3	6.38	0.38
	Drill 1	1.55	0.55
	Drill 2	1.24	0.11
	Wash/Debur	1.16	0.00
Product C	Lathe 2	4.05	0.17
	Drill 1	1.05	0.00
	Drill 2	1.76	0.19
	Wash/Debur	1.30	0.00

Note: All times are in minutes

Table 5.2: Weekly production results, FIFO scheduling

One week production results

	FIFO scheduling		
	Prod. A	Prod. B	Prod. C
Regular cell	10	32	52
Jobshop setup	20	24	44
Lower setup cell	10	39	50
Add a CCR	10	32	59

Two week production results

	FIFO scheduling		
	Prod. A	Prod. B	Prod. C
Regular cell	19	77	94
Jobshop setup	40	55	90
Lower setup cell	23	79	110
Add a CCR	22	74	117

Three week production results

	FIFO scheduling		
	Prod. A	Prod. B	Prod. C
Regular cell	30	118	145
Jobshop setup	59	89	141
Lower setup cell	36	119	170
Add a CCR	34	118	180

Four week production results

	FIFO scheduling		
	Prod. A	Prod. B	Prod. C
Regular cell	42	158	201
Jobshop setup	76	122	198
Lower setup cell	47	162	232
Add a CCR	46	162	249

Table 5.3: Weekly production results, SORTED scheduling

One week production results

	SORTED scheduling		
	Prod. A	Prod. B	Prod. C
Regular cell	6	32	73
Jobshop setup	8	21	94
Lower setup cell	7	28	83
Add a CCR	7	35	75

Two week production results

	SORTED scheduling		
	Prod. A	Prod. B	Prod. C
Regular cell	9	64	173
Jobshop setup	11	33	253
Lower setup cell	12	64	175
Add a CCR	12	78	172

Three week production results

	SORTED scheduling		
	Prod. A	Prod. B	Prod. C
Regular cell	11	99	274
Jobshop setup	12	42	429
Lower setup cell	15	105	266
Add a CCR	14	125	268

Four week production results

	SORTED scheduling		
	Prod. A	Prod. B	Prod. C
Regular cell	13	134	376
Jobshop setup	13	48	621
Lower setup cell	17	147	359
Add a CCR	15	174	363

additional CCR) produced more than the regular cell layout. For the job shop, the results were always different from the other three cases. For a four week production period, the job shop production was 76 for product A, 121 for product B, and 198 for product C. For the same period, the regular cell production was 41 for product A, 158 for product B, and 201 for product C. A feature notable in the results is that for job shop, the production was more for product A, and less for products B and C, as compared to the three cell layouts.

For SORTED scheduling, the production of product A and B was less than the production by FIFO scheduling for the same period, but was more for product C. Comparing the cases in SORTED scheduling, the two variations of the mini cell (lower setup time and additional CCR) produced more product A and B, and less product C. Though, the numbers were not significantly different from product A and C of regular cell layout. For job shop production, it produced more product A and product C, and less product B as compared to the mini cell layouts. The difference in production for product A was not noticeable, but for product B and C, it was significant.

Results were also collected for the queues. For product A and C, where the first processing operation is done at a bottleneck machine, the maximum number in the queue was one less than the maximum number of parts in a batch (Tables 5.4 and 5.5). In the case of product B, for regular mini cell and job shop, the maximum queue buildup was at the second processing operation (lathe 1), and for the other two cases, the maximum queue was at the first operation (drill 3).

There was not a significant difference in the average queue sizes for the two priority cases. In general, the maximum and average queue size for job shop layout was always greater than the other cases. Among the cell layouts, there was no significant difference between the regular cell and the lower setup cell. However, for the CCR cell, the maximum and average queue size was smaller than the other two layouts. The same observations are be made for the SORTED scheduling priority.

Overall, it can be stated that by comparing the production results, it is clear that the lower setup cell and additional CCR cell provide better results than the regular mini cell. The point to note here is that almost the same improvements are obtained by using two different approaches. One case needs capital investment of an additional machine, while in the other case, the same results are obtained by reducing setup times. Sometimes the

Table 5.4: Queue buildup (max./avg.) for each product, FIFO

Case 1	Product A Lathe 2		Product B Lathe 1		Product C Lathe 2	
	Max.	Average	Max.	Average	Max.	Average
Regular cell	9.00	4.94	25.00	7.67	49.00	25.00
Jobshop setup	44.00	22.42	65.00	63.88	159.00	18.56
Lower setup cell	9.00	4.93	34.00	17.08	49.00	25.00
Add a CCR	8.00	3.76	34.00	17.00	46.00	20.82

Table 5.5: Queue buildup (max./avg.) for each product, SORTED

Case 2	Product A Lathe 2		Product B Lathe 1		Product C Lathe 2	
	Max.	Average	Max.	Average	Max.	Average
Regular cell	9.00	4.95	25.00	7.24	49.00	25.00
Jobshop setup	44 00	22.45	65.00	56.79	159.00	18.17
Lower setup cell	9.00	4.94	34.00	17.00	49.00	25.00
Add a CCR	8.00	3.76	34.00	17.80	46.00	20.82

setup times are reduced without spending any money at all. Common sense is capable of doing wonders when it comes to reducing setup times. Job shop layout provides results different from the other three cases. Because of the bigger batch sizes and larger setup times, it produces more products of one type and less of the other type.

Just-In-Time stresses on finding simpler solutions for complex problems. In most cases, small benefits of JIT can be readily achieved without any additional capital expenditure. This can be derived from the case of lowering set up time in the mini cell. In this case, set up time was reduced by 25% (from 60 minutes to 45 minutes for product A and C and from 134 minutes to 105 minutes for product B). In none of the cases, the cell with lower set up gave worse production than the regular mini cell. This concept is so simple yet seems so hard for the companies to apply it.

5.2 Discussion on Bottleneck Operations

In each manufacturing plant, there are always processes which act as limiting factors when considering production rates. If a person working at the bottleneck can receive help, the bottleneck may disappear. However, other workers who work fast generally do not want to help others. They prefer to continue stockpiling in front of the bottleneck process. This results in a loss of overall efficiency. The supervisors should take special care of such cases and should try to balance uneven loads among workers. It is very hard to find a perfect solution specially when the lines are small with only a few workers. Still, it is good to keep in mind that work boundaries should be drawn so as to render mutual assistance [17].

The JIT approach to the presence of a bottleneck operation involves reducing setup time to produce greater capacity, finding alternative machines or processes, purchasing extra capacity or subcontracting excess work [24].

In many cases, finding a bottleneck operation is not so difficult, but in other cases it may not be so obvious. In chapter two, some research methods used to find and to schedule the bottleneck operations are discussed. Once a bottleneck operation is found, the next step is to try to solve it. A tree structured algorithm can be followed to handle the problem (Figure 5.1). The steps to be followed to eliminate or reduce a bottleneck operation are:

1. Reduce setup time

Reduction in the setup time for an operation means the machine is available for more time. Research shows that 20

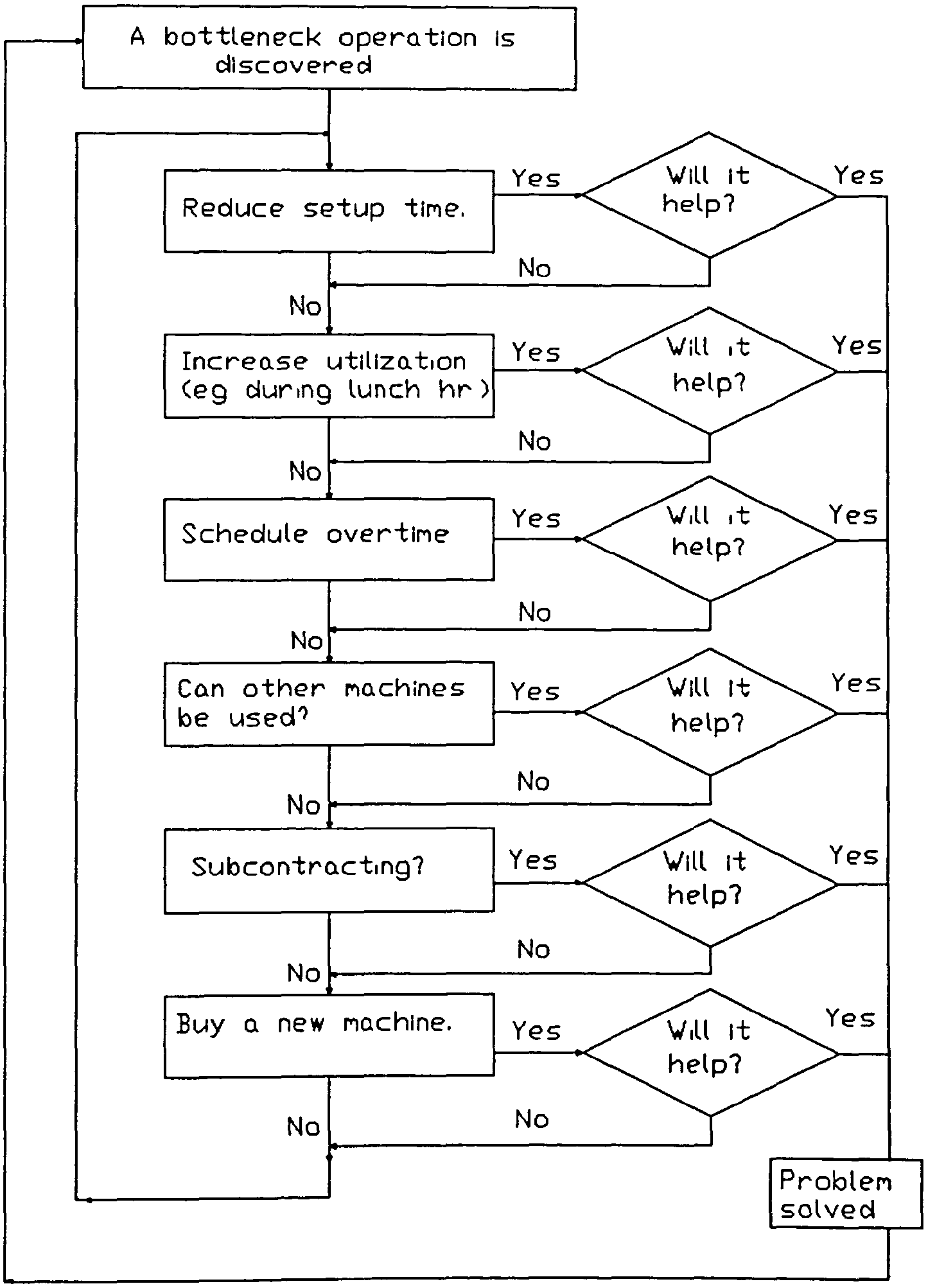


Figure 5.1: Algorithm to reduce bottleneck operations

to 30 percent of setup time can be reduced without any major expenditure [13].

2. Increase machine utilization

This can be done by scheduling to use machine when it is normally idle. A good example is during lunch hour. Scheduling can be done so that a worker or two are available during lunch hour to use the machine. Also, vacation scheduling of key personnel can be helpful so that necessary manpower is always available to use the machine.

3. Schedule overtime

If the orders on a bottleneck machine can not all be processed during the week day, overtime can be scheduled on the weekend. Working eight hours on Saturday provides 20 percent more processing time.

4. Use other machines

The management should see if other machines available in the plant can be used to partially offset the load on a bottleneck machine. This way the bottleneck machine can be freed to perform only the necessary operations which can not be scheduled on other machines.

5. Subcontracting

Whenever a part has to be produced, there always is an option to either produce it in-house or to subcontract it. A lot of factors influence the final decision, e.g., the control a plant has on scheduling and quality of a part if produced in-house versus availability at a cheaper rate from another manufacturing facility which specializes in producing it. Whatever the case is, subcontracting sure does lessen pressure on a bottleneck operation.

6. Add a new machine

This is normally the last resort because adding a new machine means capital expenditure. Decision to add another machine should be made only after all other alternatives have been exhausted.

It is better when bottleneck jumps from one operation to another. This dynamic nature implies that steps are being taken continuously to monitor the bottleneck operations and to remedy them. After one bottleneck is discovered and remedied, more studies are done to find the new bottleneck operation and the whole process is repeated again. This never ending cycle of continuous improvement is what makes the JIT philosophy so successful in all aspects of manufacturing.

6. CONCLUSION AND FURTHER RESEARCH

A simulation model designed in MicroSAINT is used in the research. It is used to study the production rates for job shop layout and mini cell layouts. The effects of reduced setup times and addition of another machine in the capacity constraint resource (CCR) or a bottleneck operation are also studied. Three products using the same machines are used in the research. The same concept can be elaborated to be used for more than three products. For the cases studied in simulation, job shop model always produced results very different from the mini cell models. For the mini cell layouts, lower setup cell and additional CCR cell consistently produced more parts for all products as compared to the regular mini cell. Lowering setup time and adding a CCR to the bottleneck operation are two of the many ways of reducing bottleneck operations.

The model can be made more complex by including percentage of parts needing rework. A rework station can be added to the model or the material to be reworked can be sent back to the required processing operation.

The MicroSAINT version used in the research is an older version which is more suitable for general purpose simulation. The new version available in the market has more features for manufacturing simulation. One thing which could not be studied through the available version, but can be studied through the new version, was the effect of adding buffer stock in front of the bottleneck machines. This concept of adding buffer stock in front of bottleneck machines is developed by E. M. Goldratt, and is a part of OPT (Optimized Production Technology). It will be interesting to compare results by running the model with and without adding necessary buffers in front of bottleneck operations.

MicroSAINT is available on all classes of computers, i.e., microcomputers, work stations, minicomputers, and mainframes. It is also compatible across all computer classes. If someone wants to use MicroSAINT for further research, it is recommended to use the mainframe version. It took a lot of time and storage space to run it on a microcomputer. The number of runs was limited to thirty for each case because of time constraints and available storage space.

7. BIBLIOGRAPHY

[1] Adams, J., Balas, E., and Zawack, D. "The Shifting Bottleneck Procedure for Job Shop Scheduling." Management Science, 34, no. 3 (March 1988): 391-401.

[2] Azadivar, F., and Lee, Y. "Optimum Number of Buffer Spaces in Flexible Manufacturing Systems." Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems. Elsevier Science Publishers, Amsterdam, 1988.

[3] Bazoian, H. "Inventory Reduction can be a Reality." Zero Inventories Seminar Proceedings, APICS, Falls Church, VA, September 1983.

[4] Burnham, J. "Excellence in Manufacturing is People Intensive." IIE Integrated Systems Conference Proceedings, IIE, Atlanta, Oct 30 - Nov 2, 1988.

[5] Clive, L. M. "Lot Sizing for Just-In-Time." International Industrial Engineering Conference Proceedings, IIE, Atlanta, May 14-17, 1989.

[6] Davies, S. F. "Strategies for Implementing the Just-In-Time Manufacturing Concepts." M.S. Thesis, Iowa State University, Ames, Iowa, 1989.

[7] Dear, A. Working Towards Just-In-Time. Kogan Page Ltd., London, 1988.

[8] Goldratt, E. M., and Cox, J. The Goal: Excellence in Manufacturing. North River Press, Inc., New York, 1984.

[9] Goldratt, E. M., and Fox, R. E. The Race. North River Press, Inc., New York, 1986

[10] Graham, I. Just In Time Management of Manufacturing. Technical Communications, London, 1988.

[11] Hay, E. "Driving Down Downtime." Manufacturing Engineering, 103, no. 3 (September 1989): 41-44.

[12] Hay, E. The Just-In-Time Breakthrough. John Wiley &

Sons, Inc., New York, 1988.

[13] Hay, E. "Reduce any Setup by 75%." Zero Inventory Philosophy and Practices Seminar Proceedings, APICS, Falls Church, VA, October 1984.

[14] Hernandez, A. Just-In-Time Manufacturing: A Practical Approach. Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[15] Kumar, K. R., and Vannelli, A. "Design of Flexible Production Systems: Capacity Balance and Subcontracting Strategies." Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems. Elsevier Science Publishers, Amsterdam, 1988.

[16] Law, A. M., and Haider, S. W. "Selecting Simulation Software for Manufacturing Applications: Practical Guidelines & Software Survey." Industrial Engineering, 31, no. 5 (May 1989): 33-46.

[17] Lu, D. Kanban - Just-In-Time at Toyota. Productivity Press, Cambridge, MA, 1986.

[18] Lubben, R. T. Just-In-Time Manufacturing: An Aggressive Manufacturing Strategy. McGraw Hill Book Company, New York, 1988.

[19] Malstrom, E. "Assessing True Cost Savings Associated with JIT Inventory Policies." IIE Integrated Systems Conference Proceedings, IIE, Atlanta, Oct 30 - Nov 2, 1988.

[20] MicroSAINT: User's Guide. Micro Analysis and Design, Inc., Boulder, 1987.

[21] Monden, Y. Toyota Production System. Industrial Engineering and Management Press, IIE, Atlanta, 1983.

[22] Mullins, P. "JIT's Competitive Edge." Production, 101, no. 7 (July 1989): 62-64.

[23] Nasbitt, J., and Aburdene, P. Megatrends 2000: Ten New Directions for 1990's. Morrow, New York, 1990.

[24] O'Grady, P. J. Putting the Just In Time Philosophy into Practice. Nichols Publishing Company, New York, 1988.

[25] Ohno, T. Toyota Production System: Beyond Large Scale Production. Productivity Press, Cambridge, 1988.

[26] Owen, J. "Bantam Costs for BANTAM Firms." Manufacturing Engineering, 103, no. 3 (September 1989): 44.

[27] Peters, T. J., and Waterman, R. H. In Search of Excellence: Lessons from America's Best-Run Companies. Warner Books, New York, 1982.

[28] Poniatowski, S. J., and Chua, R. C. "The 'softside' of JIT: A Framework for Success." International Industrial Engineering Conference Proceedings, IIE, Atlanta, May 20-23, 1990.

[29] Schmidt, P. J. "JIT and the Small Manufacturer." IIE Integrated Systems Conference Proceedings, IIE, Atlanta, Oct 30 - Nov 2, 1988.

[30] Schonberger, R. Just In Time: A Comparison of Japanese and American Manufacturing Techniques. Industrial Engineering and Management Press, IIE, Atlanta, 1984.

[31] Schonberger, R. Japanese Manufacturing Techniques: Nine Hidden Lessons in Simplicity. The Free Press, New York, 1982.

[32] Schonberger, R. World Class Manufacturing Casebook: Implementing JIT and TQC. The Free Press, New York, 1987.

[33] Sepehri, M. Just-In-Time not Just-In-Japan. American Production and Inventory Control Society (APICS), Falls Church, VA, 1986.

[34] Shingo, S. The Toyota Production System. Japan Management Association, Osaka, 1982.

[35] Tersine, R. Principles of Inventory and Materials Management. North-Holland, New York, 1988.

[36] Vannelli, A., and Kumar, K. R. "A Method for Finding Minimal bottleneck Cells for Grouping Part-Machine Families." International Journal of Production Research, 24, no. 2 (1986): 387-400.

[37] Wantuck, K. "The Japanese Approach to Productivity."

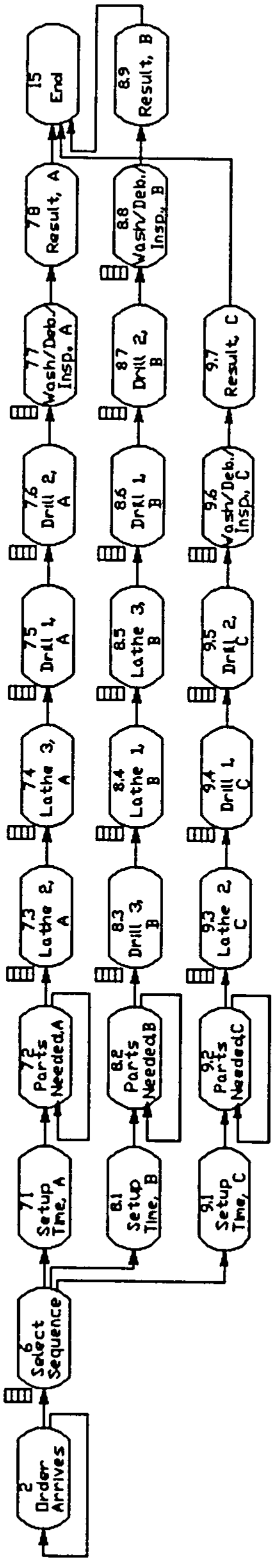
Zero Inventories Seminar Proceedings, APICS, Falls Church, VA, September 1983.

[38] Wantuck, K. "Setup Reduction for Quality at the Source." Zero Inventory Philosophy and Practices Seminar Proceedings, APICS, Falls Church, VA, October 1984.

[39] Wilt, A., and Goddin, D. "Health Care Case Study: Simulating Staffing Needs and Work Flow in an Outpatient Diagnostic Center." Industrial Engineering, 31, no. 5 (May 1989): 22-26.

[40] Xie, X. "An analysis of Sequencing rules for AS/RS in Pull-System Driven Manufacturing Assembly Lines." M.S. Thesis, Iowa State University, Ames, Iowa, 1989.

8. APPENDIX A. MICROSAINT NETWORK DIAGRAM AND MODEL


2 Order Arrives
6 Select Sequence
7.1 Setup Time, A
7.2 Parts Needed,A
7.3 Lathe 2, A
7.4 Lathe 3, A
7.5 Drill 1, A
7.6 Drill 2, A
7.7 Wash/Deb./ Insp. A
7.8 Result, A
15 End
8.1 Setup Time, B
8.2 Parts Needed,B
8.3 Drill 3, B
8.4 Lathe 1, B
8.5 Lathe 3, B
8.6 Drill 1, B
8.7 Drill 2, B
8.8 Wash/Deb./ Insp., B
8.9 Result, B
9.1 Setup Time, C
9.2 Parts Needed,C
9.3 Lathe 2, C
9.4 Drill 1, C
9.5 Drill 2, C
9.6 Wash/Deb./ Insp., C
9.7 Result, C

TASK NETWORK

```
Network Number:  0
(1)  Name:  model6
(2)  Type:  Network
(3)  Upper Network:
(4)  Release Condition:  1;
(5)  First Sub-job:  2  Order Arrives
(6)  Sub-jobs (each can be task or network):
Number:        Name:                  Type:
2              Order Arrives          Task
6              Select Sequence        Task
7.1            Setup Time, A          Task
7.2            Parts Needed, A        Task
7.3            Lathe 2, A             Task
7.4            Lathe 3, A             Task
7.5            Drill 1, A             Task
7.6            Drill 2, A             Task
7.7            Wash,Debur,Insp. A     Task
7.8            Result A               Task
8.1            Setup Time, B          Task
8.2            Parts needed           Task
8.3            Drill 3, B             Task
8.4            Lathe 1, B             Task
8.5            Lathe 3, B             Task
8.6            Drill 1, B             Task
8.7            Drill 2, B             Task
8.8            Wash,Debur,Insp. B     Task
8.9            Result B               Task
9.1            Setup Time, C          Task
9.2            Parts Needed, C        Task
9.3            Lathe 2, C             Task
9.4            Drill 1, C             Task
9.5            Drill 2, C             Task
9.6            Wash,Debur,Insp. C     Task
9.7            Result C               Task
15             End                    Task
```

```
Task Number:  2
(1)  Name:  Order Arrives
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  1;
(5)  Time Distribution Type:  Exponential
(6)  Mean Time:  200;
(8)  Task's Beginning Effect:
(9)  Task's Ending Effect:  tag=tag+1;
     CHOICE;
(10) Decision Type:  Multiple
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  6             Select   (12)  1;
(13)  2             Order    (14)  clock<simtime;
(15)                         (16)
```

```
Task Number:  6
(1)  Name:  Select Sequence
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  full==0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  1.0;
(7)  Standard Deviation:  0.50;
(8)  Task's Beginning Effect:  flag=type[tag];
     full=full+1;
(9)  Task's Ending Effect:
(10) Decision Type:  Tactical
     Following Task/Network: Tactical Expression:
      Number:       Name:
(11)  7.1           Setup    (12)  type[tag]==1;
(13)  8.1           Setup    (14)  type[tag]==2;
(15)  9.1           Setup    (16)  type[tag]==3;
(17)                         (18)
```

```
Task Number:  7.1
(1)  Name:  Setup Time, A
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  1;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  60;
(7)  Standard Deviation:  5;
(8)  Task's Beginning Effect:
(9)  Task's Ending Effect:  num1=0;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  7.2           Parts    (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  7.2
(1)  Name:  Parts Needed, A
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  1;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  0;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:
(9)  Task's Ending Effect:  count1=count1+1;
     num1=num1+1;
     if num1==1 then tag=count1+100 else tag=tag+1;
     if num1<>1 then full=full+1;
     type[tag]=1;
(10) Decision Type:  Multiple
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  7.2           Parts    (12)  if num1<10 then 1
     else 0;
(13)  7.3           Lathe    (14)  1;
(15)                         (16)
(17)                         (18)
```

```
Task Number:  7.3
(1)  Name:  Lathe 2, A
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  lathe2>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  11.15;
(7)  Standard Deviation:  2;
(8)  Task's Beginning Effect:  lathe2=lathe2-1;
(9)  Task's Ending Effect:  lathe2=lathe2+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  7.4           Lathe    (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  7.4
(1)  Name:  Lathe 3, A
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  lathe3>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  8;
(7)  Standard Deviation:  0.14;
(8)  Task's Beginning Effect:  lathe3=lathe3-1;
(9)  Task's Ending Effect:  lathe3=lathe3+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  7.5           Drill    (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  7.5
(1)  Name:  Drill 1, A
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  drill1>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  1.75;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:  drill1=drill1-1;
(9)  Task's Ending Effect:  drill1=drill1+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  7.6           Drill    (12)  1;
(13)                         (14)
(15)                         (16)



Task Number:  7.6
(1)  Name:  Drill 2, A
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  drill2>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  1.5;
(7)  Standard Deviation:  0.10;
(8)  Task's Beginning Effect:  drill2=drill2-1;
(9)  Task's Ending Effect:  drill2=drill2+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  7.7           Wash,D   (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  7.7
(1)  Name:  Wash,Debur,Insp. A
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  1;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  .55;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:
(9)  Task's Ending Effect:
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  7.8           Result  (12)  1;
(13)                        (14)
(15)                        (16)
```

```
Task Number:  7.8
(1)  Name:  Result A
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  1;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  0;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:
(9)  Task's Ending Effect:  full=full-1;
     tot[type[tag]]=tot[type[tag]]+1;
     finishd1=finishd1+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  15            End     (12)  1;
(13)                        (14)
(15)                        (16)
```

```
Task Number:  8.1
(1)   Name:  Setup Time, B
(2)   Type:  Task
(3)   Upper Network:  0  model6
(4)   Release Condition:  1;
(5)   Time Distribution Type:  Normal
(6)   Mean Time:  134;
(7)   Standard Deviation:  10;
(8)   Task's Beginning Effect:
(9)   Task's Ending Effect:  num2=0;
(10)  Decision Type:  Single choice
      Following Task/Network: Probability Of Taking
       Number:       Name:          This Path:
(11)   8.2           Parts    (12)  1;
(13)                          (14)
(15)                          (16)
```

```
Task Number:  8.2
(1)   Name:  Parts needed
(2)   Type:  Task
(3)   Upper Network:  0  model6
(4)   Release Condition:  1;
(5)   Time Distribution Type:  Normal
(6)   Mean Time:  0;
(7)   Standard Deviation:  0;
(8)   Task's Beginning Effect:
(9)   Task's Ending Effect:  count2=count2+1;
      num2=num2+1;
      if num2==1 then tag=count2+1000 else tag=tag+1;
      if num2<>1 then full=full+1;
      type[tag]=2;
(10)  Decision Type:  Multiple
      Following Task/Network: Probability Of Taking
       Number:       Name:          This Path:
(11)   8.2           Parts    (12)  if num2<35 then 1
      else 0;
(13)   8.3           Drill    (14)  1;
(15)                          (16)
(17)                          (18)
```

```
Task Number:  8.3
(1)  Name:  Drill 3, B
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  drill3>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  3.41;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:  drill3=drill3-1;
(9)  Task's Ending Effect:  drill3=drill3+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  8.4           Lathe    (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  8.4
(1)  Name:  Lathe 1, B
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  lathe1>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  12.58;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:  lathe1=lathe1-1;
(9)  Task's Ending Effect:  lathe1=lathe1+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  8.5           Lathe    (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  8.5
(1)  Name:  Lathe 3, B
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  lathe3>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  6.38;
(7)  Standard Deviation:  0.38;
(8)  Task's Beginning Effect:  lathe3=lathe3-1;
(9)  Task's Ending Effect:  lathe3=lathe3+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:              This Path:
(11)  8.6           Drill    (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  8.6
(1)  Name:  Drill 1, B
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  drill1>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  1.55;
(7)  Standard Deviation:  0.55;
(8)  Task's Beginning Effect:  drill1=drill1-1;
(9)  Task's Ending Effect:  drill1=drill1+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:              This Path:
(11)  8.7           Drill    (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  8.7
(1)  Name:  Drill 2, B
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  drill2>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  1.24;
(7)  Standard Deviation:  .11;
(8)  Task's Beginning Effect:  drill2=drill2-1;
(9)  Task's Ending Effect:  drill2=drill2+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  8.8           Wash,D  (12)  1;
(13)                        (14)
(15)                        (16)
```

```
Task Number:  8.8
(1)  Name:  Wash,Debur,Insp. B
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  1;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  1.16;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:
(9)  Task's Ending Effect:
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  8.9           Result  (12)  1;
(13)                        (14)
(15)                        (16)
```

```
Task Number:  8.9
(1)   Name:  Result B
(2)   Type:  Task
(3)   Upper Network:  0  model6
(4)   Release Condition:  1;
(5)   Time Distribution Type:  Normal
(6)   Mean Time:  0;
(7)   Standard Deviation:  0;
(8)   Task's Beginning Effect:
(9)   Task's Ending Effect:  full=full-1;
      tot[type[tag]]=tot[type[tag]]+1;
      finishd2=finishd2+1;
(10)  Decision Type:  Single choice
      Following Task/Network: Probability Of Taking
       Number:        Name:          This Path:
(11)   15             End      (12)  1;
(13)                           (14)
(15)                           (16)
```

```
Task Number:  9.1
(1)   Name:  Setup Time, C
(2)   Type:  Task
(3)   Upper Network:  0  model6
(4)   Release Condition:  1;
(5)   Time Distribution Type:  Normal
(6)   Mean Time:  60;
(7)   Standard Deviation:  5;
(8)   Task's Beginning Effect:
(9)   Task's Ending Effect:  num3=0;
(10)  Decision Type:  Single choice
      Following Task/Network: Probability Of Taking
       Number:        Name:          This Path:
(11)   9.2            Parts    (12)  1;
(13)                           (14)
(15)                           (16)
```

```
Task Number:  9.2
(1)  Name:  Parts Needed, C
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  1;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  0;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:
(9)  Task's Ending Effect:  count3=count3+1;
     num3=num3+1;
     if num3==1 then tag=count3+5000 else tag=tag+1;
     if num3<>1 then full=full+1;
     type[tag]=3;
(10) Decision Type:  Multiple
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  9.2           Parts    (12)  if num3<50 then 1
     else 0;
(13)  9.3           Lathe    (14)  1;
(15)                         (16)
(17)                         (18)
```

```
Task Number:  9.3
(1)  Name:  Lathe 2, C
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  lathe2>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  4.05;
(7)  Standard Deviation:  0.17;
(8)  Task's Beginning Effect:  lathe2=lathe2-1;
(9)  Task's Ending Effect:  lathe2=lathe2+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  9.4           Drill    (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  9.4
(1)  Name:  Drill 1, C
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  drill1>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  1.05;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:  drill1=drill1-1;
(9)  Task's Ending Effect:  drill1=drill1+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  9.5           Drill    (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  9.5
(1)  Name:  Drill 2, C
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  drill2>0;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  1.76;
(7)  Standard Deviation:  0.19;
(8)  Task's Beginning Effect:  drill2=drill2-1;
(9)  Task's Ending Effect:  drill2=drill2+1;
(10) Decision Type:  Single choice
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)  9.6           Wash,D   (12)  1;
(13)                         (14)
(15)                         (16)
```

```
Task Number:  9.6
(1)   Name:  Wash,Debur,Insp. C
(2)   Type:  Task
(3)   Upper Network:  0  model6
(4)   Release Condition:  1;
(5)   Time Distribution Type:  Normal
(6)   Mean Time:  1.30;
(7)   Standard Deviation:  0;
(8)   Task's Beginning Effect:
(9)   Task's Ending Effect:
(10)  Decision Type:  Single choice
      Following Task/Network: Probability Of Taking
       Number:       Name:          This Path:
(11)   9.7           Result   (12)  1;
(13)                          (14)
(15)                          (16)



Task Number:  9.7
(1)   Name:  Result C
(2)   Type:  Task
(3)   Upper Network:  0  model6
(4)   Release Condition:  1;
(5)   Time Distribution Type:  Normal
(6)   Mean Time:  0;
(7)   Standard Deviation:  0;
(8)   Task's Beginning Effect:
(9)   Task's Ending Effect:  full=full-1;
      tot[type[tag]]=tot[type[tag]]+1;
      finishd3=finishd3+1;
(10)  Decision Type:  Single choice
      Following Task/Network: Probability Of Taking
       Number:       Name:          This Path:
(11)   15            End      (12)  1;
(13)                          (14)
(15)                          (16)
```

```
Task Number:  15
(1)  Name:  End
(2)  Type:  Task
(3)  Upper Network:  0  model6
(4)  Release Condition:  1;
(5)  Time Distribution Type:  Normal
(6)  Mean Time:  0;
(7)  Standard Deviation:  0;
(8)  Task's Beginning Effect:
(9)  Task's Ending Effect:
(10) Decision Type:  Last task
     Following Task/Network: Probability Of Taking
      Number:       Name:          This Path:
(11)                        (12)
(13)                        (14)
(15)                        (16)
```

FUNCTION LIBRARY

```
Function Name:  CHOICE
Expression:  x=rand();
if x<=.33 then type[tag]=1;
if x>.33&x<=.67 then type[tag]=2;
if x>.67 then type[tag]=3;
n[type[tag]]=n[type[tag]]+1;
```

SIMULATION SCENARIO

```
( 1)   Event Time:        0.00
( 2)   Expression: drill1=1;drill2=1;drill3=1;
                   lathe1=1;lathe2=1;lathe3=1;
```

```
( 1)   Event Time:        0.00
( 2)   Expression:        simtime=9600;
```

JOB QUEUES

(1) Queue Number: 7.4q
(2) Queue Name: Wait for Lathe 3, A
(3) Queue forms in front of job:
7.4 Lathe 3, A
(4) Order: FIFO
(6) Entering Effect: lathe3q=lathe3q+1;
(7) Departing Effect: lathe3q=lathe3q-1;

(1) Queue Number: 7.5q
(2) Queue Name: Wait for Drill 1, A
(3) Queue forms in front of job:
7.5 Drill 1, A
(4) Order: FIFO
(6) Entering Effect: drill1q=drill1q+1;
(7) Departing Effect: drill1q=drill1q-1;

(1) Queue Number: 7.6q
(2) Queue Name: Wait for Drill 2, A
(3) Queue forms in front of job:
7.6 Drill 2, A
(4) Order: FIFO
(6) Entering Effect: drill2q=drill2q+1;
(7) Departing Effect: drill2q=drill2q-1;

(1) Queue Number: 7.7q
(2) Queue Name: Wait for W,D,I, A
(3) Queue forms in front of job:
7.7 Wash,Debur,Insp. A
(4) Order: FIFO
(6) Entering Effect: WDIq=WDIq+1;
(7) Departing Effect: WDIq=WDIq-1;

```
( 1)   Queue Number:          8.3q
( 2)   Queue Name:            Wait for Drill 3, B
( 3)   Queue forms in front of job:
             8.3            Drill 3, B
( 4)   Order:                          FIFO
( 6)   Entering Effect:    drill3q=drill3q+1;
( 7)   Departing Effect:   drill3q=drill3q-1;
```

```
( 1)   Queue Number:          8.4q
( 2)   Queue Name:            Wait for Lathe 1, B
( 3)   Queue forms in front of job:
             8.4            Lathe 1, B
( 4)   Order:                          FIFO
( 6)   Entering Effect:    lathe1q=lathe1q+1;
( 7)   Departing Effect:   lathe1q=lathe1q-1;
```

```
( 1)   Queue Number:          8.5q
( 2)   Queue Name:            Wait for Lathe 3, B
( 3)   Queue forms in front of job:
             8.5            Lathe 3, B
( 4)   Order:                          FIFO
( 6)   Entering Effect:    lathe3q=lathe3q+1;
( 7)   Departing Effect:   lathe3q=lathe3q-1;
```

```
( 1)   Queue Number:          8.6q
( 2)   Queue Name:            Wait for Drill 1, B
( 3)   Queue forms in front of job:
             8.6            Drill 1, B
( 4)   Order:                          FIFO
( 6)   Entering Effect:    drill1q=drill1q+1;
( 7)   Departing Effect:   drill1q=drill1q-1;
```

```
( 1)   Queue Number:          8.7q
( 2)   Queue Name:            Wait for Drill 2, B
( 3)   Queue forms in front of job:
             8.7            Drill 2, B
( 4)   Order:                         FIFO
( 6)   Entering Effect:   drill2q=drill2q+1;
( 7)   Departing Effect:  drill2q=drill2q-1;
```

```
( 1)   Queue Number:          8.8q
( 2)   Queue Name:            Wait for W,D,I, B
( 3)   Queue forms in front of job:
             8.8            Wash,Debur,Insp. B
( 4)   Order:                         FIFO
( 6)   Entering Effect:   WDIq=WDIq+1;
( 7)   Departing Effect:  WDIq=WDIq-1;
```

```
( 1)   Queue Number:          9.3q
( 2)   Queue Name:            Wait for Lathe 2, C
( 3)   Queue forms in front of job:
             9.3            Lathe 2, C
( 4)   Order:                         FIFO
( 6)   Entering Effect:   lathe2q=lathe2q+1;
( 7)   Departing Effect:  lathe2q=lathe2q-1;
```

```
( 1)   Queue Number:          9.4q
( 2)   Queue Name:            Wait for Drill 1, C
( 3)   Queue forms in front of job:
             9.4            Drill 1, C
( 4)   Order:                         FIFO
( 6)   Entering Effect:   drill1q=drill1q+1;
( 7)   Departing Effect:  drill1q=drill1q-1;
```

```
( 1)   Queue Number:          9.5q
( 2)   Queue Name:            Wait for Drill 2, C
( 3)   Queue forms in front of job:
             9.5            Drill 2, C
( 4)   Order:                           FIFO
( 6)   Entering Effect:     drill2q=drill2q+1;
( 7)   Departing Effect:    drill2q=drill2q-1;
```

```
( 1)   Queue Number:          9.6q
( 2)   Queue Name:            Wait for W,D,I, C
( 3)   Queue forms in front of job:
             9.6            Wash,Debur,Insp. C
( 4)   Order:                           FIFO
( 6)   Entering Effect:     WDIq=WDIq+1;
( 7)   Departing Effect:    WDIq=WDIq-1;
```

```
( 1)   Queue Number:          7.3q
( 2)   Queue Name:            Wait for Lathe 2, A
( 3)   Queue forms in front of job:
             7.3            Lathe 2, A
( 4)   Order:                           FIFO
( 6)   Entering Effect:     lathe2q=lathe2q+1;
( 7)   Departing Effect:    lathe2q=lathe2q-1;
```

```
( 1)   Queue Number:          6q
( 2)   Queue Name:            Selection sequence
( 3)   Queue forms in front of job:
             6              Select Sequence
( 4)   Order:                           FIFO
( 6)   Entering Effect:     qsize=qsize+1;
( 7)   Departing Effect:    qsize=qsize-1;
```

```
          VARIABLE  CATALOG

Name:                 Category:        Type:
finishd1              Display          Integer
count1                Display          Integer
count2                Display          Integer
count3                Display          Integer
lathe1q               Display          Integer
lathe2q               Display          Integer
lathe3q               Display          Integer
drill1q               Display          Integer
drill2q               Display          Integer
drill3q               Display          Integer
WDIq                  Display          Integer
finishd2              Display          Integer
finishd3              Display          Integer
simtime               Control          Integer
type                  Control          Array of
                                       Integers
   Number of dimensions:               1
   First dimension numbered 0 through: 15000
full                  Control          Integer
flag                  Control          Integer
n                     Control          Array of
                                       Integers
   Number of dimensions:               1
   First dimension numbered 0 through: 15000
tot                   Control          Array of
                                       Integers
   Number of dimensions:               1
   First dimension numbered 0 through: 15000
num1                  Control          Integer
num2                  Control          Integer
num3                  Control          Integer
qsize                 Control          Integer
x                     Control          Real
drill1                Task             Integer
drill2                Task             Integer
drill3                Task             Integer
lathe1                Task             Integer
lathe2                Task             Integer
lathe3                Task             Integer
clock                 System           Real
tag                   System           Integer
```

SNAPSHOTS OF EXECUTION

```
( 1)  Trigger:          Enter trigger
( 2)  Job Queue:        7.3q   Wait for Lathe 2, A
( 6)  Snapshot File:    mod6Aq
Variables to Store:
( 7)  lathe2q                     ( 8)  lathe3q
( 9)  drill1q                     (10)  drill2q
(11)  WDIq                        (12)  count1
(13)                              (14)
(15)                              (16)
```

```
( 1)  Trigger:          Enter trigger
( 2)  Job Queue:        7.4q   Wait for Lathe 3, A
( 6)  Snapshot File:    mod6Aq
Variables to Store:
( 7)  lathe2q                     ( 8)  lathe3q
( 9)  drill1q                     (10)  drill2q
(11)  WDIq                        (12)  count1
(13)                              (14)
(15)                              (16)
```

```
( 1)  Trigger:          Enter trigger
( 2)  Job Queue:        7.5q   Wait for Drill 1, A
( 6)  Snapshot File:    mod6Aq
Variables to Store:
( 7)  lathe2q                     ( 8)  lathe3q
( 9)  drill1q                     (10)  drill2q
(11)  WDIq                        (12)  count1
(13)                              (14)
(15)                              (16)
```

```
( 1)  Trigger:          Enter trigger
( 2)  Job Queue:        7.6q   Wait for Drill 2, A
( 6)  Snapshot File:    mod6Aq
Variables to Store:
( 7)  lathe2q                     ( 8)  lathe3q
( 9)  drill1q                     (10)  drill2q
(11)  WDIq                        (12)  count1
(13)                              (14)
```

```
( 1)  Trigger:        Enter trigger
( 2)  Job Queue:      7.7q   Wait for W,D,I, A
( 6)  Snapshot File:  mod6Aq
Variables to Store:
( 7)  lathe2q                    ( 8)  lathe3q
( 9)  drill1q                    (10)  drill2q
(11)  WDIq                       (12)  count1
(13)                             (14)
(15)                             (16)


( 1)  Trigger:        Enter trigger
( 2)  Job Queue:      6q     Selection sequence
( 6)  Snapshot File:  mod66q
Variables to Store:
( 7)  lathe1q                    ( 8)  lathe2q
( 9)  lathe3q                    (10)  drill1q
(11)  drill2q                    (12)  drill3q
(13)  qsize                      (14)  count1
(15)  count2                     (16)  count3


( 1)  Trigger:        Enter trigger
( 2)  Job Queue:      8.3q   Wait for Drill 3, B
( 6)  Snapshot File:  mod6Bq
Variables to Store:
( 7)  drill3q                    ( 8)  lathe1q
( 9)  lathe3q                    (10)  drill1q
(11)  drill2q                    (12)  WDIq
(13)  count2                     (14)
(15)                             (16)


( 1)  Trigger:        Enter trigger
( 2)  Job Queue:      8.4q   Wait for Lathe 1, B
( 6)  Snapshot File:  mod6Bq
Variables to Store:
( 7)  drill3q                    ( 8)  lathe1q
( 9)  lathe3q                    (10)  drill1q
(11)  drill2q                    (12)  WDIq
(13)  count2                     (14)
(15)                             (16)
```

```
( 1)  Trigger:           Enter trigger
( 2)  Job Queue:         8.5q   Wait for Lathe 3, B
( 6)  Snapshot File:     mod6Bq
Variables to Store:
( 7)  drill3q                      ( 8)  lathe1q
( 9)  lathe3q                      (10)  drill1q
(11)  drill2q                      (12)  WDIq
(13)  count2                       (14)
(15)                               (16)
```

```
( 1)  Trigger:           Enter trigger
( 2)  Job Queue:         8.6q   Wait for Drill 1, B
( 6)  Snapshot File:     mod6Bq
Variables to Store:
( 7)  drill3q                      ( 8)  lathe1q
( 9)  lathe3q                      (10)  drill1q
(11)  drill2q                      (12)  WDIq
(13)  count2                       (14)
(15)                               (16)
```

```
( 1)  Trigger:           Enter trigger
( 2)  Job Queue:         8.7q   Wait for Drill 2, B
( 6)  Snapshot File:     mod6Bq
Variables to Store:
( 7)  drill3q                      ( 8)  lathe1q
( 9)  lathe3q                      (10)  drill1q
(11)  drill2q                      (12)  WDIq
(13)  count2                       (14)
(15)                               (16)
```

```
( 1)  Trigger:           Enter trigger
( 2)  Job Queue:         8.8q   Wait for W,D,I, B
( 6)  Snapshot File:     mod6Bq
Variables to Store:
( 7)  drill3q                      ( 8)  lathe1q
( 9)  lathe3q                      (10)  drill1q
(11)  drill2q                      (12)  WDIq
(13)  count2                       (14)
(15)                               (16)
```

```
( 1)  Trigger:          Enter trigger
( 2)  Job Queue:        9.3q  Wait for Lathe 2, C
( 6)  Snapshot File:    mod6Cq
Variables to Store:
( 7)  lathe2q                     ( 8)  drill1q
( 9)  drill2q                     (10)  WDIq
(11)  count3                      (12)
(13)                              (14)
(15)                              (16)
```

```
( 1)  Trigger:          Enter trigger
( 2)  Job Queue:        9.4q  Wait for Drill 1, C
( 6)  Snapshot File:    mod6Cq
Variables to Store:
( 7)  lathe2q                     ( 8)  drill1q
( 9)  drill2q                     (10)  WDIq
(11)  count3                      (12)
(13)                              (14)
(15)                              (16)
```

```
( 1)  Trigger:          Enter trigger
( 2)  Job Queue:        9.5q  Wait for Drill 2, C
( 6)  Snapshot File:    mod6Cq
Variables to Store:
( 7)  lathe2q                     ( 8)  drill1q
( 9)  drill2q                     (10)  WDIq
(11)  count3                      (12)
(13)                              (14)
(15)                              (16)
```

```
( 1)  Trigger:          Enter trigger
( 2)  Job Queue:        9.6q  Wait for W,D,I, C
( 6)  Snapshot File:    mod6Cq
Variables to Store:
( 7)  lathe2q                     ( 8)  drill1q
( 9)  drill2q                     (10)  WDIq
(11)  count3                      (12)
(13)                              (14)
(15)                              (16)
```

```
( 1)   Trigger:            Begin trigger
( 2)   Task/Network:       7.8    Result A
( 6)   Snapshot File:      mod6rsA
Variables to Store:
( 7)   lathe2q                       ( 8)   lathe3q
( 9)   drill1q                       (10)   drill2q
(11)   WDIq                          (12)   finishd1
(13)                                 (14)
(15)                                 (16)


( 1)   Trigger:            Begin trigger
( 2)   Task/Network:       8.9    Result B
( 6)   Snapshot File:      mod6rsB
Variables to Store:
( 7)   drill3q                       ( 8)   lathe1q
( 9)   lathe3q                       (10)   drill1q
(11)   drill2q                       (12)   WDIq
(13)   finishd2                      (14)
(15)                                 (16)


( 1)   Trigger:            Begin trigger
( 2)   Task/Network:       9.7    Result C
( 6)   Snapshot File:      mod6rsC
Variables to Store:
( 7)   lathe2q                       ( 8)   drill1q
( 9)   drill2q                       (10)   WDIq
(11)   finishd3                      (12)
(13)                                 (14)
(15)                                 (16)


( 1)   Trigger:            Clock trigger
( 3)   Snapshot Time:      0.000000
( 4)   Snap Interval:      2400.000000
( 5)   Snap Stop Time:     9600.000000
( 6)   Snapshot File:      model6d
Variables to Store:
( 7)   lathe1q                       ( 8)   lathe2q
( 9)   lathe3q                       (10)   drill1q
(11)   drill2q                       (12)   drill3q
(13)   finishd1                      (14)   finishd2
(15)   finishd3                      (16)
```

```
( 1)   Trigger:            Clock trigger
( 3)   Snapshot Time:      0.000000
( 4)   Snap Interval:      2400.000000
( 5)   Snap Stop Time:     7200.000000
( 6)   Snapshot File:      model6c
Variables to Store:
( 7)   lathe1q                        ( 8)   lathe2q
( 9)   lathe3q                        (10)   drill1q
(11)   drill2q                        (12)   drill3q
(13)   finishd1                       (14)   finishd2
(15)   finishd3                       (16)


( 1)   Trigger:            Clock trigger
( 3)   Snapshot Time:      0.000000
( 4)   Snap Interval:      2400.000000
( 5)   Snap Stop Time:     4800.000000
( 6)   Snapshot File:      model6b
Variables to Store:
( 7)   lathe1q                        ( 8)   lathe2q
( 9)   lathe3q                        (10)   drill1q
(11)   drill2q                        (12)   drill3q
(13)   finishd1                       (14)   finishd2
(15)   finishd3                       (16)


( 1)   Trigger:            Clock trigger
( 3)   Snapshot Time:      0.000000
( 4)   Snap Interval:      2400.000000
( 5)   Snap Stop Time:     2400.000000
( 6)   Snapshot File:      model6a
Variables to Store:
( 7)   lathe1q                        ( 8)   lathe2q
( 9)   lathe3q                        (10)   drill1q
(11)   drill2q                        (12)   drill3q
(13)   finishd1                       (14)   finishd2
(15)   finishd3                       (16)
```

9. APPENDIX B. MICROSAINT MODEL INPUTS

MICROSAINT MODEL INPUTS

1. Regular mini cell layout

Batch sizes:
Product A: 10
Product B: 35
Product C: 50

Setup times (min.):	Mean	Std. Dev.
Product A:	60	5
Product B:	134	10
Product C:	60	5

Simulation scenario:
Lathe1 = 1, Lathe2 = 1, Lathe3 = 1,
Drill1 = 1, Drill2 = 1, Drill3 = 1.

2. Job shop layout

Batch sizes:
Product A: 45
Product B: 90
Product C: 160

Setup times (min.):	Mean	Std. Dev.
Product A:	263	20
Product B:	343	25
Product C:	224	15

Simulation scenario:
Lathe1 = 1, Lathe2 = 1, Lathe3 = 1,
Drill1 = 1, Drill2 = 1, Drill3 = 1.

3. Lower setup cell

Batch sizes:
Product A: 10
Product B: 35
Product C: 50

Setup times (min.):	Mean	Std. Dev.
Product A:	45	5
Product B:	105	8
Product C:	45	5

Simulation scenario:
Lathe1 = 1, Lathe2 = 1, Lathe3 = 1,
Drill1 = 1, Drill2 = 1, Drill3 = 1.

4. Additional CCR cell

Batch sizes:
Product A: 10
Product B: 35
Product C: 50

Setup times (min.):	Mean	Std. Dev.
Product A:	60	5
Product B:	134	10
Product C:	60	5

Simulation scenario:
Lathe1 = 1, Lathe2 = 2, Lathe3 = 1,
Drill1 = 1, Drill2 = 1, Drill3 = 1.

10. APPENDIX C. SAMPLE MODEL OUTPUT

SAMPLE MODEL OUTPUT

MEANS AND STANDARD DEVIATIONS OF SNAPSHOT DATA

Snapshot file: lsdel6a
Number of snapshots: 60

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	0.00	2400.00	1200.000	1210.127
lathe1q	0.00	24.00	2.333	6.072
lathe2q	0.00	42.00	4.900	11.175
lathe3q	0.00	0.00	0.000	0.000
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
drill3q	0.00	30.00	1.000	5.025
finishd1	0.00	69.00	7.333	13.983
finishd2	0.00	106.00	27.783	33.922
finishd3	0.00	319.00	83.183	100.100

Snapshot file: lsdel6b
Number of snapshots: 90

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	0.00	4800.00	2400.000	1970.570
lathe1q	0.00	24.00	3.967	7.399
lathe2q	0.00	42.00	4.922	10.803
lathe3q	0.00	0.00	0.000	0.000
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
drill3q	0.00	34.00	1.600	6.498
finishd1	0.00	90.00	11.778	18.687
finishd2	0.00	230.00	64.111	62.121
finishd3	0.00	600.00	175.300	171.383

MEANS AND STANDARD DEVIATIONS OF SNAPSHOT DATA

Snapshot file: lsdel6c
Number of snapshots: 120

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	0.00	7200.00	3600.000	2694.532
lathe1q	0.00	25.00	4.542	7.348
lathe2q	0.00	42.00	5.167	11.002
lathe3q	0.00	0.00	0.000	0.000
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
drill3q	0.00	34.00	2.108	7.011
finishd1	0.00	90.00	14.750	20.733
finishd2	0.00	230.00	105.092	92.694
finishd3	0.00	600.00	265.708	230.542

Snapshot file: lsdel6d
Number of snapshots: 150

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	0.00	9600.00	4800.000	3405.483
lathe1q	0.00	25.00	4.420	7.179
lathe2q	0.00	46.00	5.793	11.672
lathe3q	0.00	0.00	0.000	0.000
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
drill3q	0.00	34.00	2.207	7.145
finishd1	0.00	110.00	17.067	22.773
finishd2	0.00	455.00	146.540	120.818
finishd3	0.00	1100.00	359.400	291.547

MEANS AND STANDARD DEVIATIONS OF SNAPSHOT DATA

Snapshot file: lsd6rsa
Number of snapshots: 4510

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	81.59	19098.62	12845.035	4922.780
lathe2q	0.00	8.00	3.098	2.612
lathe3q	0.00	0.00	0.000	0.000
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0 000	0.000
WDIq	0.00	0.00	0.000	0.000
finishd1	0.00	249.00	80.265	52.408

Snapshot file: lsd6rsb
Number of snapshots: 16625

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	154.08	16885.35	8239.448	4101.940
drill3q	0.00	27.00	3.174	7.097
lathe1q	0.00	25.00	12.626	7.481
lathe3q	0.00	0.00	0.000	0.000
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
WDIq	0.00	0.00	0.000	0.000
finishd2	0.00	804.00	291.179	182.232

Snapshot file: lsd6rsc
Number of snapshots: 23750

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	67.58	10686.58	5339.485	2932.864
lathe2q	0.00	48.00	22.952	14.365
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
WDIq	0.00	0.00	0.000	0.000
finishd3	0.00	1149.00	419.974	267.287

MEANS AND STANDARD DEVIATIONS OF SNAPSHOT DATA

Snapshot file: lsd6aq
Number of snapshots: 4288

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	60.32	18978.65	12784.659	4914.293
lathe2q	0.00	9.00	4.943	2.595
lathe3q	0.00	1.00	0.053	0.225
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
WDIq	0.00	0.00	0.000	0.000
count1	2.00	250.00	81.892	52.376

Snapshot file: lsd6bq
Number of snapshots: 32300

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	127.36	16550.19	8030.756	4100.255
drill3q	0.00	34.00	17.000	9.824
lathe1q	0.00	25.00	6.632	8.345
lathe3q	0.00	0.00	0.000	0.000
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
WDIq	0.00	0.00	0.000	0.000
count2	2.00	805.00	300.929	182.278

Snapshot file: lsd6cq
Number of snapshots: 23275

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	59.68	10479.22	5232.009	2932.397
lathe2q	1.00	49.00	25.000	14.140
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
WDIq	0.00	0.00	0.000	0.000
count3	2.00	1150.00	421.474	267.273

MEANS AND STANDARD DEVIATIONS OF SNAPSHOT DATA

Snapshot file: lsd66q
Number of snapshots: 1334

Variable	Minimum	Maximum	Mean	Std. Dev.
clock	34.05	10125.68	5060.721	2763.544
lathe1q	0.00	25.00	5.399	8.082
lathe2q	0.00	49.00	8.329	14.430
lathe3q	0.00	0.00	0.000	0.000
drill1q	0.00	0.00	0.000	0.000
drill2q	0.00	0.00	0.000	0.000
drill3q	0.00	34.00	1.873	6.282
qsize	0.00	29.00	10.799	6.379
count1	0.00	110.00	19.513	22.345
count2	0.00	455.00	161.488	106.878
count3	0.00	1150.00	379.273	251.346