COMPUTER-AIDED DESIGN OF INTEGRATED PRODUCTION
PLANNING AND INVENTORY CONTROL SYSTEMS,

A Thesis Presented to
The Faculty of the College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Masters of Science

by
Dawud Kamal Handal
November, 1987
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>iv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER I: INTRODUCTION</td>
<td>2</td>
</tr>
<tr>
<td>1. Background</td>
<td>2</td>
</tr>
<tr>
<td>2. The Research Objective</td>
<td>8</td>
</tr>
<tr>
<td>3. Thesis Organization</td>
<td>10</td>
</tr>
<tr>
<td>CHAPTER II: THE FLEXIBLE SIMULATION MODEL (FSM)</td>
<td>12</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>12</td>
</tr>
<tr>
<td>2. Performance Measures</td>
<td>14</td>
</tr>
<tr>
<td>3. Production Smoothing [Macro Planning]</td>
<td>17</td>
</tr>
<tr>
<td>3.1 Forecasting</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Production Scheduling</td>
<td>17</td>
</tr>
<tr>
<td>3.2.1 Determining the Bottleneck Operations</td>
<td>27</td>
</tr>
<tr>
<td>4. Job Sequencing Models (Rules) [Micro Planning]</td>
<td>27</td>
</tr>
<tr>
<td>5. Inventory Models [Micro Planning]</td>
<td>29</td>
</tr>
<tr>
<td>6. FSM Structure</td>
<td>37</td>
</tr>
<tr>
<td>6.1 Forecasting</td>
<td>37</td>
</tr>
<tr>
<td>6.2 Production Scheduling and Inventory Control</td>
<td>41</td>
</tr>
<tr>
<td>7. Input/Output Sessions and Files</td>
<td>50</td>
</tr>
<tr>
<td>7.1 Input/Output Session for Forecasting</td>
<td>50</td>
</tr>
<tr>
<td>7.2 Input/Output Sessions for Production Scheduling and Inventory Control</td>
<td>57</td>
</tr>
<tr>
<td>CHAPTER III: THE STRUCTURED ANALYSIS AND DEVELOPMENT APPROACH</td>
<td>63</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>63</td>
</tr>
<tr>
<td>2. Structured Analysis and Development, a Systems Approach</td>
<td>65</td>
</tr>
<tr>
<td>Table</td>
<td>Page No.</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>5.1</td>
<td>The Groups of Signs According to the Ohio Uniform Traffic Control Signs Chart</td>
</tr>
<tr>
<td>5.2</td>
<td>Size Categories for Aluminum Signs</td>
</tr>
<tr>
<td>6.1</td>
<td>Average Inventory Carrying Costs for the Warehouses and the Sign Shop, the Percentage of Orders Satisfied from Inventory, and the Penalty Costs for not Providing Adequate Service</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Periodic-Review Policy</td>
<td>30</td>
</tr>
<tr>
<td>2.2</td>
<td>Continuous-Review Policy</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Graphical Representation of the System's Boundary and its Components</td>
<td>67</td>
</tr>
<tr>
<td>3.2</td>
<td>Micro-Analysis of System Components</td>
<td>69</td>
</tr>
<tr>
<td>3.3</td>
<td>Hierarchical System Structure</td>
<td>70</td>
</tr>
<tr>
<td>4.1</td>
<td>CONTEXT Diagram</td>
<td>89</td>
</tr>
<tr>
<td>5.1</td>
<td>A Map of the Twelve Districts and the Location of the Sign Shop with Respect to Each District</td>
<td>177</td>
</tr>
<tr>
<td>5.2</td>
<td>Examples of Traffic Signs</td>
<td>181</td>
</tr>
<tr>
<td>5.3</td>
<td>Sequence of Operations for Aluminum Signs Which Have a Rectangular Shape</td>
<td>185</td>
</tr>
<tr>
<td>5.4</td>
<td>Sequence of Operations for Aluminum Signs Which Have a Triangular or a Pentagonal Shape</td>
<td>186</td>
</tr>
<tr>
<td>5.5</td>
<td>Sequence of Operations for Extruded Aluminum Signs</td>
<td>188</td>
</tr>
<tr>
<td>5.6</td>
<td>Sequence of Operations for Plywood Signs</td>
<td>189</td>
</tr>
<tr>
<td>5.7</td>
<td>Bill of Materials for Aluminum, Extruded Aluminum, and Plywood Signs</td>
<td>191</td>
</tr>
<tr>
<td>5.8</td>
<td>Categories According to Which Signs are Grouped</td>
<td>195</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENT

This thesis is dedicated to my family in gratitude for their caring and encouragement. I would like to thank them from the deepest of my heart for their love and support throughout the years.

I wish to express my deepest appreciation to my thesis director, Dr. Robert Terry, for his dedication throughout this research. I would like to thank him for his guidance and cooperation. Introducing me to this research topic and his availability for help made it a very delightful experience.

To my wife Lisa, I give my deepest gratitude for her support and encouragement. Her caring and love in the most delightful atmosphere is my most beautiful experience. I would also like to sincerely thank her family for their encouragement and advice. Their guidance and help have been greatly appreciated.

I also wish to thank the other members of my committee, Dr. Eswaran and professor Ralph Sims for their invaluable criticism.

The friendship and companionship of my fraternal brothers made my stay in the United States of America a very happy one. I thank them all.
CHAPTER I

INTRODUCTION

1. BACKGROUND

An integrated production planning and inventory control systems sets and monitors the course that a production and distribution system (PDS) takes as it moves through time. Such a system must operate within the limits that are set by the organization's goals, and be flexible enough to accommodate any changes in the environment. The process of designing planning production and controlling inventories requires making decisions at two different levels. The first is macro planning, while the second is micro planning.

The purpose of macro planning is to minimize the cost of satisfying the demand for a product line which is subject to seasonal fluctuations. There are a number of production smoothing strategies which can be used to accomplish this objective: (1) building inventories during periods of low demand and depleting them during periods of high demand, (2) changing the size of the work force by hiring and laying off, and (3) assigning overtime hours.
The first step of macro planning involves forecasting the demand for the products. The forecast must extend over a horizon which is at least as long as the seasonal cycle present in the product's demand. The second step uses this forecast as an input to a process that determines the mixture of the strategies mentioned above, which will minimize the sum of the relevant costs. Examples of such relevant costs are: (1) inventory carrying costs, (2) the cost of changing the work force, (3) subcontracting costs, and (4) the cost of overtime.

The second level in the process of planning production and controlling inventory is micro planning. This involves three major types of activities: (1) requirements planning which involves deciding when and how much to order of the raw materials needed to produce the respective products, (2) job sequencing which involves determining the order in which the jobs (production orders) in a given time block are to be produced, and (3) inventory control which involves controlling the inventory of products and raw materials at both the production facilities and the warehouses.

The traditional approach to tackling the problem of designing an IPPIC, which encompass forecasting, production scheduling, requirements planning, job sequencing, and
inventory control, has been to solve these subproblems independently of one another. Evidence of this can be obtained by referring to texts for production planning and inventory control. Examples of such texts are Buffa and Miller (1979), Johnson and Montgomery (1974), Peterson and Silver (1979), Elsayed and Boucher (1985), and Chase and Aquilano (1977).

Unfortunately, since there are factors that can cause these problems to be mutually dependent, the approach taken by many studies cannot detect interactions between the different components and may lead to suboptimal results. There are two causal mechanisms which may cause such dependencies. The first results from the fact that better production scheduling and job sequencing can reduce setup times. The reduction in setup times in effect allows a larger proportion of the time available to be devoted to actual production. This, in turn, has the potential for reducing manufacturing lead times. Reduced manufacturing lead times can be used to provide better customer service without increasing inventory levels. Alternatively, reduced lead times can be used to reduce inventories while maintaining the same level of customer service.

The second causal mechanism results from the fact that a better prediction of demand could reduce the need to
interrupt an established production schedule to produce rush orders. This, in turn, could lead to a reduction in the number of setups. As explained in (1) above, this could lead to a reduction in the level of inventory required to provide a fixed level of service, or an improvement in service without the need for increasing inventory. Alternatively, more accurate demand predictions could lead to a reduction in the level of inventories required for providing a specified level of customer service. If a set of mutually dependent problems are solved as if they were independent, then the solution to the total problem is likely to be sub-optimal.

The process of designing an IPPIC system is complicated further by the following factors:

(1) The number of decision variables can be quite large. Because of the interaction with environmental variables, the effects created by a decision variable may be influenced by the level of certain environmental variables.

(2) Some of the decision variables may have vastly different effects on different performance measures which are necessary to adequately describe the performance of the system. A change in one variable, while holding other variables constant, may produce a positive impact on one
response variable and a negative impact on another. For example, increasing the level of inventories may improve customer service, but increases inventory carrying costs. In addition, the multiple measures makes it difficult to describe the systems performance as a single numerical value.

(3) The effect of the decision variables on the various performance measures may be subject to uncertainty. Various sources of uncertainty can influence the performance of the system such as the uncertainty associated with environmental factors. Examples of such factors are market demand and availability of raw material when ordered from the supplier. Demand is always subject to change "without notice".

(4) It is very difficult for the human mind to understand how changes in one part of a complex systems are propagated to other parts of that system. This complicates the task of synthesizing the solutions to the separate subproblems into the solution to the overall system design problem. For example, it is hard to quantify how changing the production sequencing rule will affect order backlogs.

(5) Since the economic conditions frequently change, the external environment in which the PDS operates will also be
subject to change. Examples of such changes include economic recessions, natural disasters, strikes, and military conflicts. These environmental changes can cause the plan for scheduling production and controlling inventories to become obsolete. To prevent this, the production scheduling and inventory control problem should be resolved whenever major environmental changes occur. As noted above, solving such problems can be a complex and time consuming process.

In contrast to the traditional approach, the work done in this thesis recognizes that designing an IPPIC system involves solving the subproblems (forecasting, production scheduling, requirements planning, job sequencing, and controlling inventory) simultaneously. In order to accomplish this task, it was recognized that a computerized and systematic procedure is needed for synthesizing the solution to the original problem (designing an IPPIC system) from the solutions to the subproblems. This procedure should consist of models for determining optimal solutions to problems associated with manufacturing and warehousing products. This would enable it to be used to evaluate different designs of the IPPIC system in order to determine how a particular combination of models (forecasting, production scheduling, job sequencing, and inventory control models) would affect overall system
This procedure would have the capacity to (1) keep track of the large number of decision variables and evaluate the effects that they produce and (2) keep track of the decision variables and evaluate the effects they have on the various performance measures. It would also have the capacity to (3) perform a number of trials and then calculate the average to determine what the effect will be on the average, and (4) determine how changes in one part of a complex system are propagated to other parts.

2. THE RESEARCH OBJECTIVE

The main objective of this research is to develop a flexible simulation model (FSM) which can be used to help design IPPIC systems, and has the capabilities of the systematic procedure mentioned above. The FSM consists of three different model banks: (1) forecasting models, (2) production scheduling and job sequencing models, and (3) inventory management models. The FSM also contains an evaluation facility which can be used to evaluate the performance of the system for various combinations of the three different types of models. For each model, the FSM has a library of different alternative models from which the IPPIC system designer can choose. The goal of the designer is to determine the optimal architecture of the
optimal fashion.

The long term objective of the research program of which this thesis was a part, is to develop an FSM which is capable of modeling the most commonly encountered type of PDS. The PDS may either have its own transportation vehicles or subcontract a commercial carrier to transport the products. However, the scope of this thesis was restricted to a PDS which consists of a single flow shop production facility with a single echelon system of geographically decentralized warehouses. It is also assumed that the products are shipped from the production facility by a commercial carrier. As a result of, the transportation capability of the PDS is assumed to be adequate.

A secondary objective of this thesis is to develop the FSM in a modular fashion in order to accommodate any changes that would be required when considering other types of production and distribution systems, and provide the best foundation for such modifications. To facilitate this objective, structured systems analysis and development (SSAD) methodology was used. The SSAD methodology was utilized to generate the specifications of the model and help understand how the different components of the system function and how they interact with one another.
The FSM consists of modules that represent the different components depicted by the specifications, which consist of three different model banks: (1) forecasting models, (2) production scheduling and job sequencing models, and (3) inventory control models. Each bank consists of a number of specific model alternatives from which the IPPIC designer can choose. The designer's goal should be to select a particular combination of models which will enable the PDS to be operated in an optimal fashion. The FSM also contains an evaluation facility which can be used to evaluate the performance of the system for various combinations of the three different types of models.

Throughout this thesis, the main objective will be referred to as the "FSM development goal (goal 1)", and the secondary objective will be referred to as the "FSM enhancement goal (goal 2)".

3. THESIS ORGANIZATION

Chapter II provides a detailed description of the FSM, which consists of a description of its architecture and the different types of models included in the FSM. Chapter III discusses the structured analysis and development (SSAD) tools that are used in developing the specifications of the
generate the specifications of the FSM. The detailed structure of the FSM and the developed specifications, showing how the different parts of the system are interrelated, are also described in this chapter. Chapter V provides a background of a case study. It gives a description of the type of manufacturing environment involved and the type of products being produced. It also describes the product structure and the different operations and processes required for producing the products. Chapter VI details the results obtained from the case study. Both the quantitative and qualitative findings will be drawn and analyzed from the experimental results. Chapter VII concludes this thesis with the summary and contributions of this research. Further research problems are also presented.

Readers interested in using the FSM to designing IPPIC systems for a PDS which consist of a flow shop and a single echelon system of geographically decentralized warehouses, should concentrate on reading chapters II, V, and VI. Readers interested in modifying the FSM to provide means for designing IPPIC systems for other types of PDSs, should concentrate on reading chapters III, IV, and section 3.2 of chapter II.
CHAPTER II

THE FLEXIBLE SIMULATION MODEL (FSM)

1. INTRODUCTION

As discussed in the previous chapter, the main objective of this thesis is to develop a flexible simulation model (FSM) to help design an integrated production planning and inventory control (IPPIC) system. The present version of the FSM assumes that the IPPIC system is to be designed for a production and distribution system (PDS) which consists of a flow shop production facility and a single echelon system of geographically decentralized warehouses.

The goal of the production and distribution systems (PDS) designer is to determine the optimal architecture of the IPPIC system, which consists of three different types of decision support models: (1) forecasting models, (2) production scheduling, and (3) inventory control. For each model, the FSM has a library of alternative models from which the designer can choose. The designer’s objective should be to select the particular combination of models, which will enable the PDS to be operated in an optimal
or nearly fashion.

The FSM was written in PASCAL 8000 for use on the IBM 4341 mainframe series. With simple modifications, this model can be enhanced for use on microcomputers. PASCAL was used because it is a highly structured procedural language. Through this language, a model can be constructed in a modular fashion. This will provide a great deal of flexibility when the model has to be custom tailored to handle a PDS which is different from the one considered here. This modularity provides the capability to reuse applicable modules in orders to minimize the amount of work required to create a FSM for analyzing other types of PDSs.

The remainder of this chapter is organized in the following fashion: section 2 describes the performance criteria used for evaluating the impact of the different combinations of forecasting, production scheduling, job sequencing, and inventory models on the system's performance. Section 3 gives a description of the production smoothing models. Section 4 describes the job sequencing models. The inventory control models are described in section 5. Section 6 lists the different modules of the FSM. This section also describes the functions and the different inputs and outputs of each
2. PERFORMANCE MEASURES

Although different industries typically have different goals, most of them strive for better customer service, i.e., completion of the work on or before the promised delivery date [Brown (1968)]. There are two ways for providing better customer service. The first is to maintain larger and more diverse inventories, but this will increase the costs of carrying inventories. Hence, a second goal may be to avoid excessive inventories. The second way for providing better customer service is by increasing the capacity of the shop to handle rush orders. Since such capacity is expensive, a third goal might be to avoid unnecessary capacity. In this research, the criteria for comparing alternative IPPIC designs were used:

1) Total inventory holding costs. This is the cost of holding inventory of finished products at both the central production facility and the geographically decentralized warehouses. It is calculated by multiplying the inventory holding cost ($/unit) by the time average of the number of units in inventory (units/week).
2) **Total inventory shortage costs.** This is the cost of incurring shortages in the inventory of finished products. It is calculated by multiplying the inventory shortage cost ($/unit short) by the average number of units short (units/week).

3) **Average factory to warehouse delivery lead time.** This is the average time it takes the production facility to deliver the products to the warehouses after receiving an order.

4) **Average warehouse to customer delivery lead time.** This is the average time it takes the warehouses to deliver the products to the customers after receiving an order.

5) **Machine utilization.** This consists of two measures:
   (1) the percentage of time a machine is utilized for manufacturing products. This measures the average utilization of each machine for every period and the maximum and minimum utilization of the machine for all periods.
   (2) the average time a machine is utilized for setup.

6) **Average lateness of warehouse orders.** The lateness of the orders is the number of days an order is
delivered to the warehouses beyond the due date.

7) **Average lateness of customer orders.** The lateness of the orders is the number of days an order is delivered to the warehouses beyond the original due date.

8) **Labor efficiency.** This consists of two measures:
   (1) the fraction of time the machine operators spend on machine setup and on actual production with respect to labor hours available.
   (2) the amount of time operators spend on machine setup with respect to the operators active time (active time equals the sum of machine setup and actual production times).

9) **Percentage of orders satisfied from inventory.** This criteria measures the number of orders satisfied from inventory with respect to the total orders made for stocked products.

The FSM gives the user the capability of choosing any combination of the above criteria as the performance measures. Since the purpose of this thesis is to describe the FSM and illustrate its functions, only brief discussions are provided about the forecasting, production
scheduling, job sequencing, and inventory control models used.

3. PRODUCTION SMOOTHING [Macro Planning]

3.1 FORECASTING

Forecasts are essential in many types of business entities and organizations in order for them to be capable of making intelligent decisions. In production scheduling the prediction of demand must be available so that the firm can plan production schedules and inventory maintenance. Forecasts are also needed to (1) plan the allocation of total inventory investments, (2) place replenishment orders, and (3) identify the needs for additional production capacity.

Forecasting methods can be grouped into two general categories [Banks, Spoerer, and Collins (1986)]: qualitative and quantitative. Qualitative, or non-statistical, methods include S-curves, game theory, and the Delphi method. In this study, the attention is focused on quantitative forecasting techniques.

Quantitative forecasting methodologies can be divided further into two major sub-categories: (1) non-causal or time series models and (2) causal models. Time series
models assume that historical data contains all the information which is useful for predicting future values. On the other hand, causal models utilize leading variables as inputs for forecasting the demand. In this thesis attention will be restricted to time series models. Furthermore, this thesis considers only two types of time series models: (1) the simple moving average model and (2) the simple exponential smoothing model.

3.1.1 Moving Average Model

This model assumes that the following data generating process underlines the time series which is to be forecasted:

\[ d_T = D + V \]

where

- \( d_T \) is the actual demand during period \( t \),
- \( D \) is the average demand during period \( t \),
- \( V \) is a random variable with a mean zero and constant variance \( \sigma^2 \) over time.

The equation above describes a situation where the underlying demand generating process is constant from one period to the next. Since, for any particular period, the value of \( (D) \) is subject to changes due to random events [Peterson and Silver (1979)], a moving forecast of level demand is calculated as follows:
where

\[ D_{T,N} = (D_{T-1,N}) + \frac{d_T - d_{T-N}}{N} \]

where

\( D_{T,N} \) is the \( N \) period moving average calculated at time \( T \) and \( N \) is the number of past periods selected for calculating the moving average.

The above equation also states that the moving average is simply the average of the \( N \) most recent observations. To facilitate finding the best \( N \), which gives the lowest value of the forecast errors, the FSM is configured to give the user the option to choose a range for \( N \). The errors for each value of \( N \) are then displayed, giving the user the opportunity to study how different values of \( N \) affect the forecasting errors.

3.1.2 Exponential Smoothing

This is an alternative approach to the moving average model described above. It attempts to "track" changes in a time series manner by using newly observed time series values to "update" the estimates of the parameters describing the time series [Bowerman and O'Connell, 1987]. The exponentially smoothed average \( (D_T) \) at time \( t \) is calculated as follows [Peterson and Silver, 1979]:

\[ D_T = (1 - \alpha) \cdot D_{T-1} + \alpha \cdot d_T \]

where \( \alpha \) is the smoothing factor (0 < \( \alpha \) < 1).
where $a$ is a smoothing constant, $0 < a < 1$

The longer the forecast update period, the larger should be the smoothing constant, $(a)$. The role of $(a)$ is quite similar to the role of $N$ in the moving average model. In fact, given $n$ from a simple moving average model, a value of $(a)$ can be computed to produce an equivalent simple exponential smoothing model using the following formula [Peterson and Silver, 1979].

$$a = 2 / (N+1)$$

In addition, simple exponential smoothing models require less computations and less storage of data than is the case with moving averages.

The simple exponential smoothing and moving average models are appropriate only when the data exhibits a constant process. For processes that exhibit linear trends, the double exponential smoothing model is recommended. If the process exhibits a quadratic trend in the mean, then triple exponential smoothing is more appropriate. On the other hand, if the data shows seasonality, the Holt and Winter method of exponential
smoothing should be used.

Other models that can be used for time series analysis are models known as the ARMA family. In fact, this family of models is sufficiently flexible to include the moving average and simple, double, and triple exponential smoothing models as special cases [Bowerman and O'Connell, 1987]. Moreover, it is flexible enough to include Holt and Winter methods of exponential smoothing as special cases.

3.2 PRODUCTION SCHEDULING

In general, production scheduling involves the assignment of dates to start the production of specific jobs or operation steps. The purpose of production scheduling is to optimize the use of resources so that (1) delivery due dates can be met and (2) inventory levels can be updated in a sufficient manner.

The flow of production through the production facility can be maximized by maximizing the flow through the bottleneck operation (the bottleneck is defined to be that operation which cannot produce the demand in a given period). In order to accomplish this task, a linear programming (LP) model was developed. This LP model is used to determine the optimal quantities of products to be
produced for the specific periods. The LP model can also be used to determine which of the operations (machine center) is the bottleneck operation. The following notation will be used in describing the linear programming model.

Stated mathematically, the objective is to minimize

\[ z = \sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{p=1}^{P} \left[ C_{tmp}(X_{tmp}) + HC_{sp}(I_{sp}) \right] \]

The first term in the brackets represents the cost of producing \( X \) units of product \( p \) during period \( t \) on machine \( m \). The second term represents the cost of holding products in inventory.

There are two constraints which must be satisfied:

1) \( X_{tp} + I_{t(p-1)} = D_{tp} + I_{tp} \)

This constraint states that the number of units to be produced of product \( p \) during period \( t \) plus the inventory \( I_{t(p-1)} \) of that product during period \( t-1 \) must be equal to the demand (\( D_{tp} \)) plus the inventory \( I_{tp} \) of the same product.

2) \( \sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{p=1}^{P} X_{tmp} (PT_{mp}) \leq MH_{tm} \)
The second constraint states that the total time required to produce $X$ units of products at the end of period $t$ must be least equal to the machine hours available ($MH_{tm}$) for that period.

where

$C_{tmp}$ = production cost of product $p$ during period $t$ on machine $m$,  

$X_{tmp}$ = number of units to be produced of product $p$ during period $t$ on machine $m$,  

$HC_{tp}$ = inventory holding costs of product $p$ during time $t$,  

$I_{tp}$ = inventory level of product $p$ during period $t$,  

$D_{tp}$ = demand of product $p$ during period $t$,  

$PT_{tm}$ = processing time required to produce product $p$ on machine $m$ during period $t$,  

$MH_{tm}$ = machine hours available on machine $m$ during period $t$,  

$T$ = the number of periods in the planning horizon,  

$M$ = process number (machine centers),  

$P$ = number of products,

The purpose of the LP model was to determine the bottleneck operation. After the bottleneck has been determined, the sequencing rules are then used to determine
the order in which the various products should be processed through the bottleneck. Then, the bill of materials is exploded and the various raw materials are launched at a time to allow them to arrive at the bottleneck just before they are needed.

The above LP model provides a scheduling system for producing the products given a set of constraints such as demand constraints, inventory balance constraints, raw materials and machine hours available. It determines the optimum number of units that can be produced at a given period while simultaneously maintaining the inventory levels at a minimum.

Hence, in conjunction with the inventory control models (described in section 5) and the job sequencing models (described in section 4), this LP model is used to simultaneously (1) ensure the availability of products for planned production and product deliveries, (2) maintain the lowest possible level of inventory, and (3) plan manufacturing activities, delivery schedules, and purchasing activities.

**Linear Programming Model INPUTS**

LP requires a number of inputs: (1) planned production orders for products, (2) the inventory status records, (3)
the processing time required for processing the products on the machines, and (4) the machine hours available at each machine center for each period.

The planned production orders show how much of each product is needed and when it is required. The inventory status record contain the status of the inventory levels of products. The bill of materials (BOM) contains the information about the different raw materials required for manufacturing the products at every level. This information is used to generate the raw materials constraint in the LP model.

**Linear Programming Model OUTPUTS**

The outputs generated from the LP model are the optimum number of units that can be produced and when they should be produced. Note that the units that can be produced may be different from the units planned for if the shop's capacity was insufficient. An indirect output, determined from the slack variables of the final LP tableau, is the bottleneck operation. It will be shown later how the bottleneck is determined.

The above model is valid only if the demand for an item is not directly related to or dependant on the demand for a higher level item. But, with simple modifications, that
are shown below, this LP approach can also be used for the case where demand for an item is directly related to the demand for a higher level item.

For example, assume that an automobile manufacturer has two plants, one for manufacturing the tires and the second for manufacturing the car itself. Assume also that the spare tire for each car is the same as the other four tires, and that the time lag for production between the car and the tires is two weeks.

Hence, defining \( X_{\text{car},t} \) to be the number of cars produced at time \( t \) and \( X_{\text{tire},k} \) to be the number of tires produced at time \( k \), the LP model can be modified to account for this dependency and "time-phase" planning by including the following constraint:

\[
X_{\text{car},t} = (X_{\text{tire},t+2})/5 \\
\text{or} \quad 5X_{\text{car},t} = X_{\text{tire},t+2}
\]

In addition, the linear programming model described above applies only to one flow line type of facility (flow shop LP (FSLP)) and does not consider the case where multiple routes can be used to manufacture a product (job shop LP (JSLP)). For this case the LP model would have to be modified to consider the multiple routes by which
products can be produced. The notations used for describing this model is presented in Appendix A for the interested readers.

3.2.1 Determining the Bottleneck Operations

The above linear programming model is solved for the optimal number of units that can be produced. From the final solution tableau, if a slack variable corresponding to a machine center equals zero, then the machine is classified as a bottleneck. On the other hand, if the slack variable is greater than zero, then the machine is classified as a non-bottleneck, otherwise, the machine is classified as a bottleneck.

4. JOB SEQUENCING MODELS (Rules) [Micro Planning]

Two of the most common and effective sequencing rules, which are utilized in this FSM to the order in which products are sequenced through the production facility:

1) Shortest Processing Time (SPT).

This rule, which is also known as the shortest operation time, selects first the job with the shortest processing time on the machines. This sequencing rule minimizes work-in-process inventories (Carroll, 1965). This rule also
minimizes mean flow time, mean completion time, and mean waiting time. However, this is accomplished at the expense of keeping the bigger jobs longer. This suggests that the variance of delivery lead times might increase. This variance might provide useful information for determining if poor scheduling is responsible for too long delivery lead times.

2) Shortest Slack Time (SST).
This rule, which is also known as the least slack time, selects first the job with the smallest slack. Slack is defined as the number of days remaining before the due date minus the duration time (processing time) of the job. Therefore, this rule is commonly used when the main criteria is customer service (i.e. shorter delivery lead times).

The FSM has been designed in a modular fashion so that it can be easily enhanced to consider more sequencing rules such as FCFS-first come first serve, SS-static slack (due date less time of arrival at machine center), RPT-remaining processing time, NOR-number of operations remaining, and many others.
5. INVENTORY MODELS (Policies) [Micro Planning]

Inventory models refer to the review and ordering policies used in deciding when orders should be placed and how much to order (orders could be orders for finished products carried in the warehouses, or production orders for products issued within the production facility). The inventory models utilized in the FSM are:

1) Periodic-Review Model

Under this model, inventory levels are reviewed at the end of equal periods of time, $K$ ($K$ is the length of the review period). If at the end of $K$, the inventory level is above a predetermined reorder level (the reorder level $r$ is defined to be the inventory level at which an order is issued for products or raw materials), no action is taken. Otherwise, an order is placed to bring the inventory level back to the desired level. Shown in Figure 2.1, this model can be represented as follows:

$$Q_k = \begin{cases} 
0 & \text{if } I_k > r \\
I_{\text{max}} - I_k & \text{if } I_k \leq r 
\end{cases}$$
$T = T_2 = T_3 = \ldots = T_n$

Figure 2.1 Periodic-Review Policy
\( I_k \) = inventory level at the end of period \( K \).
\( r \) = reorder level (point)
\( I_{\text{max}} \) = maximum inventory level
\( Q_k \) = replenishment quantity \( (I_{\text{max}} - I_k) \).

2) Order Up To \( I_{\text{max}} \) Model

Under this model, the \( r \) is set equal to \( I_{\text{max}} \).
Consequently, an order of size \( Q_k = I_{\text{max}} - I_k \) is always placed at the end of period \( K \).

3) Continuous-Review Model

Under this model, the inventory level is continuously monitored and orders of size \( Q_k = I_{\text{max}} - I_k \) are always placed whenever the inventory level \( I_k \) drops to the reorder level \( r \), or below.
The difference between the periodic-review model and the continuous model is that under the periodic model, orders may or may not be placed at the end of period \( K \) depending on the inventory level. Whereas under the continuous model, orders are always placed when the inventory level drops to, or below, \( r \). This model is represented in Figure 2.2.

4) Base Stock Model

Under this model the reorder level \( r \) is set to \( I_{\text{max}} \), and orders are placed after each withdrawal
Figure 2.2 Continuous-Review Policy.
from inventory. Consequently, the sum of the amount of
inventory on hand, $I_k$, and the amount being ordered, $Q_k$,
must equal $I_{\infty}$ at all times. $I_{\infty}$ is referred to as the
base stock level.

5) On-Demand (On-Order) Model

Under this model no inventory is maintained.
Basically, as an order for products is received, a
production order is issued for that product.

5.1 Calculating safety stock ($S$), $r$, and $Q$

The demand for products can either be deterministic or
variable. In both cases, the present version of the FSM
considers the case where the lead time is constant, or the
variation in lead time is small enough, in relation to the
average lead time, that it can be closely approximated by a
constant lead time. In addition, it only considers the
case where backorders are allowed and does not consider the
lost sales case. Below is a brief illustration of how
these parameters are determined for each case.

5.1.1 Demand is deterministic

The reorder point ($r$) is obtained by determining the
demand that will occur during the lead time period. When
the inventory level \([(\text{on hand})+(\text{on order})-(\text{back orders})]\)
reaches the reorder point, an order will be placed for Q units. \( r \) is calculated as follows:

\[
r = \left( \frac{DL}{52} \right) = \text{reorder point in units}
\]

where \( L \) is the lead time expressed in weeks and \( D \) is the annual demand for products in units.

Since back orders are permitted, the reorder point is also calculated as the lead time demand minus the number of units back ordered, or

\[
\text{Reorder point} = (\text{lead time demand}) - (\text{back orders})
\]

\[
r = \left( \frac{DL}{N} \right) - (Q - I_{\text{max}})
\]

where \( N \) is the number of operating days per year and \( L \) is the lead time in days, and \( Q \) is calculated as

\[
Q = \frac{2(OC)D}{CC}
\]

where \( OC \) is ordering cost per year, \( D \) is the annual demand in units, and \( CC \) is the inventory carrying cost per unit per year.

The maximum inventory \( (I_{\text{max}}) \) is calculated from the equation above as follows:
5.1.2 Demand is variable

When the demand follows a Poisson or a continuous distribution, the annual cost of safety stock \( (S) \) can be calculated as follows [Starr and Miller, 1972]:

Annual cost of \( (S) \) = holding cost + stockout cost

\[
ACS_a = CC(r-L) + \frac{(BC)(D) P(L>r)}{Q}
\]

where

\( CC = \) carrying costs per unit per year
\( r = \) reorder point
\( L = \) average lead time demand in units
\( BC = \) backorder cost per shortage
\( D = \) average annual demand in units
\( P(L>r) = \frac{[(CC)Q]/(BC)D}{L} \)
\( Q = \) order quantity in units

Taking the derivative of the above formula with respect to the reorder point and setting it equal to zero, the following is obtained:

\[
f(r) = \frac{[(CC)Q]/[(BC)D]}{}
\]
Using proper tables, the value of \( f(r) \) can be used to obtain the values of the reorder point and the safety stock. If the demand follows a normal distribution, the reorder point and safety stock can be determined from standard normal distribution tables and the following formulas:

\[
Z = \frac{r - \bar{L}}{\sigma_{D} \sqrt{L}} = \frac{r - DL}{\sigma_{D} \sqrt{L}} = \frac{S}{\sigma_{D} \sqrt{L}}
\]

\[
S = Z\sigma_{D} = Z\sigma_{D} \sqrt{L}
\]

\[
r = DL + S
\]

where

\( Z \) = standard normal deviate,

\( L \) = lead time,

\( S \) = safety stock in units,

\( \sigma_{L} \) = standard deviation of lead time demand,

\( \sigma_{D} \) = standard deviation of demand for a time other than the lead time,
6. FSM STRUCTURE

As mentioned earlier, the simulation model was written in PASCAL 8000 because it is a highly structured procedural language. The simulation model is a menu driven model which consists of two main parts. Part one, which involves forecasting, consists of 9 modules. Part two, which involves production scheduling and inventory control, consists of 23 modules. The rest of this section provides a description these modules and their functions. Section 7 provides a description of the input/output sessions of each part.

6.1 FORECASTING

(1) VARIABLE DECLARATION.

This module declares the global variables used by the model. Each variable is assigned to one of the following types: integer, real, character, or boolean. Along with each assignment, a definition of what the variable represents is given.

1. Integer: a whole number not containing decimal points (1, 5, 10, 33, etc.).

2. Real: a number containing decimal points (1.20, 3.04, 4.43, 100.01, etc.).

3. Character: a variable represented as a letter. For example, COLOR='Y' indicates that the variable COLOR has the value 'Y' which represents "yellow".

4. Boolean: under this type, a variable can have only one of two values: TRUE or FALSE.
(2) INITIALIZE.

This module sets the variables to their initial values. Such variables include the time sequenced demand of each product for each period during the analysis horizon, the number of periods, the number of observations $n$, mean square error, mean absolute deviation, mean absolute percentage error, etc.

(3) OPEN/CLOSE.

This module initializes the different input/output files.

(4) INTRODUCTION.

This module gives the user a brief introduction to the model. It provides a list of the different forecasting models from which the user can select the ones desired. It gives the user the option to look at the description of each model and its input requirements. During this session, the user selects the desired model to forecast the demand for products.

(5) READ-DATA.

This module reads in the data from the input file. For each period, every data line consists of the product code and its demand. This module also asks for the number of periods in the planning horizon.
(6) MOVINGAVG.

This module performs the appropriate functions to forecast the demand for products using the moving average method. First, it asks the user for the appropriate input data as shown below. During the session, this module displays the following results on the screen: the mean square error (MSE), mean absolute deviation (MAD), and mean absolute percentage error (MAPE) for each of the moving average as mentioned in section 3.1.1. Then, it asks the user if any changes are desired in: (1) the range or the exact value of the number of terms included in the moving average nor (2) the number of periods to forecast ahead. Finally, it writes the results to an output file.

Input: demand data, the range or exact number of terms included in the moving average n, the number of periods in the future to forecast.

Output: mean square error, mean absolute deviation, mean absolute percentage error, and the forecasted demand for each product by period.

(7) EXPONENTIAL.

This module performs the appropriate functions to forecast the demand for products using the simple exponential smoothing method. First, it asks the user for the appropriate input data as shown below. During the session, this module displays the following results on the
screen: the residual analysis, mean square error (MSE), mean absolute deviation (MAD), and the mean absolute percentage error (MAPE). Then, it asks the user if any change in the smoothing constant is desired. Finally, it writes the results to an output file.

Input: demand data; the number of periods for initialization if the user wishes that the program calculates the initial value, otherwise, the initial value; the smoothing constant; and the number of periods in the future to forecast.

Output: residual values, mean square error, mean absolute deviation, mean absolute percentage error, the smoothing constant, the number of periods in the future to forecast, the smoothing statistics, and the forecasted demand for each product by period.

(8) PRINT-RESULTS.

This module sends the results to the printer if the user wishes to obtain a hard copy of the results.

(9) MAIN SECTION

This is the main section of this part. It calls the above modules in the appropriate sequential manner to execute a simulation run.
6.2 PRODUCTION SCHEDULING AND INVENTORY CONTROL

(1) VARIABLE DECLARATION.
This module is the same as module (1) mentioned in section 6.1 above.

(2) INIT1 and INIT2.
Set the variables to their initial values. Such variables include inventory levels, reorder points, shipments, maximum inventory levels, etc.

(3) OPEN_CLOSE.
Initializes the different input/output files. It opens those files to be read from and written to. It also closes files where appropriate.

(4) INTRO.
Gives the user a brief introduction to the model. It provides a list of the different rules and policies from which the user can select the ones desired.

(5) READDATA.
Reads in the data from the input file. Each data line consists of the product code, size of product, quantity ordered, warehouses number, unit cost, raw materials, date ordered, date wanted, and any other related information.
(6) DWRHouses.

This module represents the warehouses operations that involve dispatching the products from inventory to fill an order if they are available in stock, and signaling the appropriate modules if products are not available in stock.

**Input:** period number, planned product orders (product type, size, color, shape, quantity, order date, due date, etc.), inventory levels, and back orders.

**Output:** product dispatches, updated back orders, updated inventory levels, new orders for products, the signals to appropriate modules, and the number of units short from inventory.

(7) UPDATE_INV.

Updates the stock level of products at the warehouses and the production facility after a retrieval from inventory (dispatches and shipments) or a receipt of finished products at the warehouses.

**Input:** product shipments, dispatches, and receipts at the production facility and the warehouses.

**Output:** updated inventory levels at the production facility and the warehouses.

(8) INVCOST.

Calculates inventory carrying and shortage costs for
both the warehouses and production facility for each period.

**Input:** period number, the number of units short inventory at the production facility and warehouses (calculated by modules DWRHOUSES and FLOWSHOP), the number of units in inventory at the production facility and warehouses, and inventory carrying and shortage costs per unit.

**Output:** the period's inventory carrying and shortage costs at the production facility and warehouses.

(9) ORDER1.

Applies the inventory policy selected to control the inventory level of the products at the warehouses. This module contains three of the five inventory policies used in this model: (1) periodic review, (2) order up to $I_{max}$, and (3) continuous review.

**Input:** period number, inventory policy, inventory policy parameters, product dispatches from the warehouses to the market, and the inventory level of each product.

**Output:** orders for products if any is needed.

(10) ORDER2.

This module is a continuation of ORDER1, and contains
the other two inventory policies: (1) base stock and (2) on-demand policies. The reason for having two modules for these policies was space limitations.

**Input:** period number, inventory policy, inventory policy parameters, product dispatches, and the inventory level of each product.

**Output:** orders for products if any is needed.

(11) **BACKORDER.**

Keeps track of the back orders and changes their status from outstanding to satisfied when the respective order is received.

**Input:** product dispatches, outstanding back orders, and product inventory levels.

**Output:** updated status of back orders.

(12) **FLOWSHOP.**

Represents the operations performed at the production facility which involve shipping the products if available in stock, and signaling the appropriate modules if products are not available in stock.

**Input:** inventory policy, period number, planned product orders (product type, size, color, shape, quantity, order date, due date, etc.), product inventory levels.

**Output:** product shipments, production orders, the
signals to the appropriate modules, and the number of units short from inventory.

(13) CHECKINV.

Applies the inventory policy selected to control the inventory level of the products at the production facility.

Input: inventory policy, period number, product shipments, and product inventory levels.

Output: production orders for products if needed.

(14) GROUP.

Groups the product orders according to the categories given the set of rules to be used for grouping the jobs (items).

Input: number of jobs and type, size, color, shape, quantity of each product order, and grouping rules.

Output: the grouped jobs.

(15) PROCTIMES.

Depending on the type of product, this module calls the respective module which contain the processing times data to calculate the processing time required for producing the products. This processing time includes transportation and material handling time. The setup time is added on after the module SETUPT, described below, is called.
Input: number of jobs (items) and type, size, color, shape, quantity of products, etc. for each product order.

Output: the processing times required for producing the products at each machine center.

(16) SETUPT.
Assigns the setup time for each job on each machine that it will processed it.

Input: number of jobs and type, size, color, shape, quantity of products, etc. for each product order.

Output: the time required to prepare the machines for production.

Modules 17, 18, and 19 were added specifically for the case study discussed in the next chapter. These modules are used for assigning the processing times for manufacturing the different products at the different machine centers.

(17) EXTRU.
Calculates the processing time, for extruded aluminum products, at the machines on which will process the job.

Input: width, height, color, and quantity of products, and the number of letters and symbols to be displayed on the product.
Output: the processing times required for producing each product at each machine center.

(18) PLY.
Calculates the processing time, for plywood products, at the machines which the job will be processed on.

Input: width, height, color, and quantity of products, and the number of letters and symbols to be displayed on the product.

Output: the processing times required for producing each product at each machine center.

(19) ALUM.
Calculates the processing time, for aluminum products, at the machines which will process the job.

Input: width, height, color, shape, and quantity of products, and the information to be displayed.

Output: the processing times required for producing each product at each machine center.

(20) SPROCT.
Schedules the production for the grouped jobs, and then sequences them for production using the shortest processing time sequencing rule.

Input: number of jobs, setup times, processing times, the current date, and material requirements.
Output: the processing times required for producing each product at each machine center.

(21) SSLACKT.

Schedules the production for the grouped jobs, and then sequences them for production using the shortest slack time sequencing rule.

Input: number of jobs, setup times, processing times, the current date, and material requirements.

Output: the processing times required for producing each product at each machine center.

(22) PERFORMANCE.

Calculates the average inventory carrying and shortage costs (per period) at both the warehouses and the production facility, the average delivery lead time from the production facility to the warehouse, and the average delivery lead time from the warehouses to the customers. It also calculates the mean lateness of jobs, the percentage of orders satisfied from inventory for stocked products, machine utilization, and labor efficiency. It then writes these results, which measure the performance of the system, to an output file. An example of this output file is given at the end of this chapter.

Input: total inventory carrying and shortage costs for the planning horizon, the total delivery lead
time of all deliveries made, the total time the orders are late, the number of orders placed by the warehouses for stocked and non-stocked products, total time machine is utilized for setup and for actual production, the total time labor spends on machine setup and on actual production, machine hours available, and labor hours available.

Output: the average inventory carrying and shortage costs at the production facility and warehouses, the average delivery lead time, the mean lateness of jobs, machine utilization, labor efficiency, and percentage of orders satisfied from inventory for stocked products.

(23) MAIN SECTION.

This is the main part of the model. It calls the above modules in the appropriate sequential manner to execute a simulation run. It also writes the periodic results from the production facility and warehouses operations to output files. Examples of these output files are given at the end of this chapter.
7. INPUT/OUTPUT SESSIONS AND FILES

The input sessions are divided into two major sessions: (1) Forecasting session, where the optimal values of the moving average \( n \) and the smoothing constant \( a \) are determined, and the demand for products is forecasted. (2) Production scheduling and inventory control session, where the production sequencing and inventory models are selected to evaluate the impact of each particular combination of models on the overall system performance. The output of the first session is then used as the input to the second session.

7.1 Input/Output Session for Forecasting:

(Screen 1):

**********************************************************************************
* * DESIGNING INTEGRATED PRODUCTION PLANNING AND INVENTORY CONTROL SYSTEMS * *
* * A THESIS RESEARCH PROJECT * *
* * Part I: FORECASTING * *
* * BY * *
* * DAWUD K. HANDEL * *
* * ADVISOR: DR. W. R. TERRY * *
* * INDUSTRIAL AND SYSTEMS ENGINEERING * *
* * OHIO UNIVERSITY * *
* * ATHENS, OHIO * *
* * (August, 1987) * *
* *
**********************************************************************************
(Screen 2):

YOUR OPTIONS ARE:

(1) A DESCRIPTION OF THIS MODEL
(2) START THE PROGRAM
(3) QUIT

Enter selection (1, 2, or 3) ---->

If the user's selection was (1), screen 2.1 below appears, if the user's selection was (2), screen 2.2 shown later appears, otherwise, the user is given the following message:

"THANK YOU, HAVE A GOOD DAY".

(Screen 2.1):

THIS PROGRAM WILL GENERATE FORECASTS OF DEMAND. THE USER CAN CHOOSE EITHER THE SIMPLE MOVING AVERAGE METHOD OR THE SIMPLE EXPONENTIAL SMOOTHING MODEL TO MAKE THE FORECASTS.

Do you wish to see a description of each method and their input requirements (Y/N)? ---->

If the user answers "Y" (yes) to the above, then screen 2.1.1 below appears, otherwise, the screen 2.2 shown later appears.

(Screen 2.1.1):

1. THE MOVING AVERAGE METHOD: THIS MODEL PROVIDES A GOOD FIT TO DATA FROM A PROCESS WHICH HAS A CONSTANT MEAN. AT ANY TIME T, THE AVERAGE OF THE N MOST RECENT OBSERVATIONS IS COMPUTED, AND THIS IS REFERRED TO AS THE MOVING AVERAGE AT PERIOD T.

INPUTS: HISTORICAL DEMAND, THE NUMBER OF PERIODS IN THE FUTURE TO FORECAST, AND THE VALUES FOR N TO BE USED IN THE PROGRAM.

OUTPUTS: FORECASTED DEMAND, RESIDUAL VALUES, AND
MEASURES OF MODEL ACCURACY (MEAN SQUARE ERROR (MSE), MEAN ABSOLUTE DEVIATION (MAD), AND MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)).

(Screen 2.1.2):

2. THE EXPONENTIAL SMOOTHING METHOD: THIS MODEL IS RECOMMENDED WHEN THE DATA COMES FROM A PROCESS WITH A CONSTANT MEAN. AT TIME ZERO, THE USER INITIATES THE PROCESS BY SUPPLYING AN INITIAL VALUE D(0). AT EACH TIME PERIOD, T, AFTERWARDS, A SMOOTHED STATISTIC D(T) IS COMPUTED AS A WEIGHTED AVERAGE OF THE OBSERVATION X(T) AND THE SMOOTHED STATISTIC D(T-1). ALPHA (0 <= a <= 1) IS A USER SPECIFIED WEIGHT CARRIED BY X(T), WHILE 1-a IS CARRIED BY D(T-1).

INPUTS: THE INITIAL SMOOTHING VALUES, THE HISTORICAL DATA POINTS, THE VALUE FOR ALPHA (a), AND THE DESIRED NUMBER OF PERIODS IN THE FUTURE FOR FORECASTING.

OUTPUTS: FORECASTED DEMAND, RESIDUAL VALUES, AND MEASURES OF MODEL ACCURACY (MEAN SQUARE ERROR (MSE), MEAN ABSOLUTE DEVIATION (MAD), AND MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)).

After this introduction the following screen appears:

(Screen 2.1.3):

YOUR OPTIONS ARE:

(1) START THE PROGRAM
(2) QUIT
Enter selection (1 or 2) ------>

If the user selects (1), screen 2.2 shown below appears, otherwise the session terminates.

(Screen 2.2):

1. WHICH MODEL WOULD YOU LIKE TO USE, THE MOVING AVERAGE (MA) MODEL OR THE EXPONENTIAL SMOOTHING MODEL (ES) ?

Enter selection (MA or ES) ------>
2. HOW MANY TIME PERIODS (N) ARE IN THE DATA? ----->

3. HOW MANY PRODUCTS (P) ARE THERE? ----->

4. DO YOU WISH TO:
   a) INPUT THE DATA FROM THE KEYBOARD, OR
   b) READ DATA FROM A DATA FILE

Enter selection (a or b) ----->

If the user selects (a), screen 2.2.4.a below appears, otherwise screens 2.2.4.b shown later appears.

(Screen 2.2.4.a): (entering data from the keyboard)

FOR EACH PRODUCT ENTER THE FOLLOWING DATA FOR THE (N) PERIODS:

PERIOD 1:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PERIOD N:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Screen 2.2.4.b): (entering data from a file)

TYPE IN THE NAME OF THE DATA FILE ----->

PLEASE MAKE SURE THAT THE FORMAT OF THE DATA, FOR EACH PERIOD, IN THE DATA FILE LOOKS AS FOLLOWS:

<table>
<thead>
<tr>
<th>PERIOD</th>
<th>DEMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Screen 3):

YOUR OPTIONS ARE:

(1) RUN THE PROGRAM
(2) QUIT

Enter selection (1 or 2) ----->

If the user selects (1) and the moving average (MA) model was chosen, screen 4-MA below appears, but if the user selected the exponential smoothing (ES) model, then the screen 4-ES shown later appears. Otherwise the session terminates.

(Screen 4-MA):

PLEASE GIVE A RANGE (BETWEEN 2 AND (N-1)) FOR THE NUMBER OF TERMS INCLUDED IN THE MOVING AVERAGE (N). (the program will display a short output of: Mean Square Error (MSE), Mean Absolute Deviation (MAD), and Mean Absolute Percentage Error (MAPE) for N).

PLEASE ENTER THE RANGE FOR (N):

n1 ----->
n2 ----->

HOW MANY PERIODS (T) AHEAD DO YOU WISH TO FORECAST ? ( range: 1 <= T < (N-n2) ) ----->

The following screen is displayed:

(Screen 4.1-MA):

(1) ACCURACY FOR (N) = N1 .
(2) MEAN SQUARE ERROR (MSE) = XXXX .
(3) MEAN ABSOLUTE DEVIATION (MAD) = XXXX .
(4) MEAN ABSOLUTE PERCENTAGE ERROR (MAPE) = XXXX .

Re-run OR Terminate (R or T) ----->

If the user chooses to terminate, an output file is created as shown below, otherwise screen 4-MA described above appears again. The user can easily display this file on the screen or obtain a printout of it.
Output file of the exponential smoothing method:

The following is an example of how the output file will look like.

<table>
<thead>
<tr>
<th>PERIOD</th>
<th>DEMAND</th>
<th>MOVING AVERAGE</th>
<th>FORECAST</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4.8</td>
<td>5.22</td>
<td></td>
<td>.38</td>
</tr>
<tr>
<td>7</td>
<td>5.6</td>
<td>5.27</td>
<td>5.22</td>
<td>.33</td>
</tr>
<tr>
<td>8</td>
<td>5.6</td>
<td>5.47</td>
<td>5.47</td>
<td>-.07</td>
</tr>
<tr>
<td>9</td>
<td>5.4</td>
<td>5.47</td>
<td>5.47</td>
<td>1.03</td>
</tr>
<tr>
<td>10</td>
<td>6.5</td>
<td>5.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>5.1</td>
<td>5.5</td>
<td></td>
<td>-.48</td>
</tr>
<tr>
<td>24</td>
<td>6.0</td>
<td>5.77</td>
<td>5.68</td>
<td>.32</td>
</tr>
<tr>
<td>25</td>
<td>5.8</td>
<td>5.77</td>
<td>5.77</td>
<td>.03</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>5.77</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ACCURACY MEASURES FOR N (N=25) = 6
MEAN SQUARE ERROR (MSE) = .305
MEAN ABSOLUTE DEVIATION (MAD) = .452
MEAN ABSOLUTE PERCENTAGE ERROR (MAPE) = 7.792

(Screen 4-ES):

YOUR OPTIONS ARE:

(1) INPUT INITIALIZATION VALUE [ S(0) ] YOURSELF
(2) HAVE THE PROGRAM GENERATE THIS INITIAL VALUES.

Enter selection (1 or 2) ----->

If the user selects (2), screen 4.1-ES shown below appears, otherwise the following appears before this screen

ENTER THE INITIALIZATION VALUE [ S(0) ] ----->
1. ENTER THE NUMBER OF PERIODS (n) FOR INITIALIZATION
   (range: 1 <= n <= N-1) ---->

2. ENTER THE SMOOTHING CONSTANT (0 < a < 1)
   (default value is 0.2) ---->

3. HOW MANY PERIODS AHEAD DO YOU WISH TO FORECAST
   (range: 1 <= T <= N-n-1) ---->

After providing the above data, a data file is created
which contains the residual values and the forecast data as
shown below. The user can easily display this file on the
screen or obtain a printout of it.

Output file of the moving average method:

The following is an example of how the output file will
look like.

RESIDUAL ANALYSIS
-------------------

<table>
<thead>
<tr>
<th>PERIOD</th>
<th>DEMAND</th>
<th>LEVEL</th>
<th>RESIDUALS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9074</td>
<td>9785.7</td>
<td>-711.7</td>
</tr>
<tr>
<td>2</td>
<td>10128</td>
<td>9785.7</td>
<td>342.3</td>
</tr>
<tr>
<td>3</td>
<td>10155</td>
<td>9785.7</td>
<td>369.3</td>
</tr>
</tbody>
</table>

ACCURACY MEASURES FOR THE FIRST 3 OBSERVATIONS:

MEAN SQUARE ERROR (MSE) = 380034.4
MEAN ABSOLUTE DEVIATION (MAD) = 474.44
MEAN ABSOLUTE PERCENTAGE ERROR (MAPE) = 4.9533

FORECAST 1 PERIODS AHEAD FOR ALPHA (a) = 0.2

<table>
<thead>
<tr>
<th>PERIOD</th>
<th>DEMAND</th>
<th>SMOOTHING STATISTICS</th>
<th>FORECAST</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>9746</td>
<td>9777.70</td>
<td>9785.70</td>
<td>-39.7</td>
</tr>
<tr>
<td>5</td>
<td>9397</td>
<td>9701.59</td>
<td>9777.70</td>
<td>-380.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>6</td>
<td>9012</td>
<td>9563.70</td>
<td>9701.59</td>
<td>-689.6</td>
</tr>
<tr>
<td>7</td>
<td>9084</td>
<td>9467.70</td>
<td>9563.70</td>
<td>-479.7</td>
</tr>
<tr>
<td>8</td>
<td>9447</td>
<td>9463.59</td>
<td>9467.70</td>
<td>-20.7</td>
</tr>
<tr>
<td>9</td>
<td>10062</td>
<td>9583.30</td>
<td>9463.59</td>
<td>598.4</td>
</tr>
<tr>
<td>10</td>
<td>10420</td>
<td>9750.59</td>
<td>9583.30</td>
<td>836.7</td>
</tr>
</tbody>
</table>

**Mean Square Error (MSE)**

\[
\text{MEAN SQUARE ERROR (MSE)} = 820538.90
\]

**Mean Absolute Deviation (MAD)**

\[
\text{MEAN ABSOLUTE DEVIATION (MAD)} = 841.2139
\]

**Mean Absolute Percentage Error (MAPE)**

\[
\text{MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)} = 7.5259
\]

7.2 Input/Output Sessions for Production Scheduling and Inventory Control

(Screen 1):

***************
*           *
* DESIGNING INTEGRATED PRODUCTION PLANNING AND INVENTORY CONTROL SYSTEMS *
*                        A THESIS RESEARCH PROJECT *
*                           Part II: PRODUCTION SCHEDULING AND INVENTORY CONTROL *
*                        * BY *
*                       DAWUD K. HANDEL *
*                        ADVISOR: DR. W. R. TERRY *
*                    INDUSTRIAL AND SYSTEMS ENGINEERING *
*                         OHIO UNIVERSITY *
*                        ATHENS, OHIO *
*                        (August, 1987) *
*                     ***************
(Screen 2):

YOUR OPTIONS ARE:

(1) A DESCRIPTION OF THIS PROGRAM
(2) START THE PROGRAM
(3) QUIT

Enter selection (1, 2, or 3) ---->

If the user selects (1), screen 2.1 shown below appears, if the user selects (2), screen 2.2 shown later appears, otherwise the following message appears

"THANK YOU. HAVE A NICE DAY"

(Screen 2.1):

THIS IS SESSION 2 OF A MODEL DEVELOPED TO HELP DESIGN INTEGRATED PRODUCTION PLANNING AND INVENTORY CONTROL SYSTEMS. IT CONSISTS OF PRODUCTION SEQUENCING AND INVENTORY CONTROL MODELS FOR DETERMINING OPTIMAL SOLUTIONS TO PROBLEMS ASSOCIATED WITH MANUFACTURING AND WAREHOUSING PRODUCTS. SESSION 1 CONSISTED OF FORECASTING MODELS FOR PREDICTING THE DEMAND FOR PRODUCTS.

THE PRESENT VERSION OF THIS PROGRAM CONSIDERS TWO SEQUENCING RULES (shortest processing time and shortest slack time) AND FIVE INVENTORY POLICIES (periodic-review, order up to I(MAX), continuous-review, base stock, and on-order (on-demand)).

DO YOU WISH TO SEE A DESCRIPTION OF THESE MODELS (Y/N)?

If the user answers "Y" (yes), screen 2.1.1 shown below appears, otherwise screen 2.2 shown later appears.

(Screen 2.1.1):

I) THE DIFFERENT INVENTORY POLICIES YOU CAN CHOOSE FROM TO CONTROL THE INVENTORY OF PRODUCTS ARE:
1. PERIODIC-REVIEW POLICY:

Under this policy inventory levels are reviewed at the end of equal periods of time, K.

2. ORDER UP TO Imax POLICY:

Under this policy an order equal to (maximum inventory - current inventory) is always placed at the end of period K.

3. CONTINUOUS-REVIEW POLICY:

Under this policy the inventory is continuously reviewed and orders are placed (maximum inventory - current inventory) whenever the current inventory level drops to the reorder level r, or below.

*** PLEASE HIT ANY KEY TO CONTINUE ----->

(Screen 2.1.1-2):

4. BASE STOCK POLICY:

Under this policy orders are placed after every withdrawal from inventory. Each order is equal to (maximum inventory - current inventory).

5. ON-DEMAND (ON-ORDER) POLICY:

Under this policy no inventory is maintained.

*** PLEASE HIT ANY KEY TO CONTINUE ----->

(Screen 2.1.1-3):

THE DIFFERENT PRODUCTION SEQUENCING RULES YOU CAN CHOOSE FROM TO SEQUENCE THE JOBS FOR PRODUCTION ARE:

1. SHORTEST PROCESSING TIME:

This rule sequences the jobs according to their processing time, from shortest to longest.
2. SHORTEST SLACK TIME:
---------------------
This rule sequences the jobs according to their slack time (number of days remaining before due date - processing time), from shortest to longest.

*** PLEASE HIT ANY KEY TO CONTINUE ------>

After hitting a key, the following screen appears:

(Screen 2.1.2):
YOUR OPTIONS ARE:

(1) RUN THE SIMULATION
(2) QUIT

Enter selection (1 or 2) ------>

If the user selects (1), screen 2.2 shown below appears, otherwise the session terminates.

(Screen 2.2):
THE INVENTORY POLICIES YOU CAN CHOOSE FROM ARE:

1. Periodic-review policy,
2. Order Up to $I_{\text{MAX}}$ policy,
3. Continuous-review policy,
4. Base stock policy,
5. On-Demand policy.

* Please Choose a Policy for the Warehouses
  (1, 2, 3, 4, or 5) ------>

* Please Choose a Policy for the Production Facility
  (1, 2, 3, 4, or 5) ------>

(Screen 2.2.1):
THE PRODUCTION SEQUENCING RULES YOU CAN CHOOSE FROM TO SEQUENCE THE JOBS FOR PRODUCTION ARE:

1. Shortest processing time,
2. Shortest slack time.
Output files:

Once the user is done with the input session described above, the simulation starts with initializing the variables and input/output files. At the end of each simulation run three files are created, two of which are updated at the end of each period and the third is created once at the end of the run. Examples of these output files are shown in Appendix C. The three output files are described below:

1)WAREHOUSES DATA FILE:

This file contains the warehouses data by period. It shows, for every period, the demand for the products, the different orders placed by the warehouses, the inventory levels of each respective product, and how much of each product was dispatched to the market for that period. This file also shows the inventory carrying and shortage costs at the warehouses.

2) PRODUCTION FACILITY DATA FILE:

This file contains the production facility data by period. It shows the different production orders which were issued for products either to update inventory levels or satisfy orders received from districts. It also shows
how much of each product was produced and when the products were shipped. Finally, it shows the inventory carrying and shortage costs at the production facility.

3) RESULTS DATA FILE:

This file is created at the end of each simulation run. It contains the data regarding the performance of the system given the specific rule and policies combination. This data consists of the inventory carrying and shortage costs at the warehouses and the production facility per period, the average delivery lead times, the mean lateness of jobs, machine utilization, labor efficiency, and percentage of orders satisfied from inventory for stocked products.
CHAPTER III

THE STRUCTURED ANALYSIS AND DEVELOPMENT APPROACH

1. INTRODUCTION

As mentioned in chapter I, the second goal of this thesis is to build the flexible simulation model (FSM) in a modular fashion to provide the best foundation for extending the FSM to other types of production and distribution systems (PDS). To accommodate this objective, the methodology of structured systems analysis and development, using diagramming techniques and computer aided tools, were used. In order to understand the specifications, which are developed in chapter IV, this chapter provides a detailed explanation of this structured approach, which includes a description of its techniques and tools.

Historically, systems analysis and development has resulted in over-budget, inflexible systems, which have not necessarily met the objectives and requirements of the users [Yourdon and Constantine, 1978]. As the complexity and variety of problems facing the system analyst increase,
it becomes clear that traditional intuitive approaches to problem solving are inadequate.

The systems analyst traditionally has been thought of as interfacing with users and designers during the analysis phase, that is, only at the first stage of the system development life cycle. They have also had difficulty in building a tangible model of a system that can be communicated to users. This was due to the fact that the systems analysts' work, typically, have been expressed in volumes of English narrative, sprinkled with technical jargon and supported by program-level flowcharts. No specific or structured methods were used to illustrate how the system's modular functions fit together, and how the different components interact with one another. The result has been a document that has overwhelmed users and obscured the logical requirements of the system [McMenamin and Palmer, 1984].

Consequently, it can be recognized that a systematic approach which can overcome the above problems and account for the complexity of the problem must be utilized. This approach must accommodate any interactions between the system's components and also those between the system and it's environment. In addition, it must also provide a rational framework for addressing all aspects that are
relevant to the problem. Initially, this would appear to be an overwhelming task, but if comprehensive systematic approaches are utilized, solutions can be achieved that address the broader relevancies of the problem.

The structured analysis and development methodology is a technique that represents a broad-based systematic approach to problems [DeMarco, 1978], which can be used to achieve the second objective of this thesis. It is particularly useful when problems are complex and affected by many factors, and entail the creation of a problem model that corresponds as closely as possible in some sense to reality. This correspondence is needed to ensure that the analyst does not "end up" working on the wrong problem, and that the developed solution is for the right problem. Its usefulness increases with the problem complexity and the number of decision variables because it permits the system designer or analyst to take a broad overall view of the problem under consideration. The following section describes such an approach, its objectives, and its techniques and tools.

2. STRUCTURED ANALYSIS AND DEVELOPMENT, A SYSTEMS APPROACH

This structured approach was first used regularly in the data processing field to describe specific guidelines to be followed when writing a program - thus, structured
programming [Dahl, Dijkstra, and Hoare, 1972]. It was
developed to prevent the problems which are created when
each programmer, without specific guidelines and
techniques, developed his/her own haphazard strategy to
program systems. There was no standardization from project
to project or even from program to program. In general,
programmers spent most of their time doing tasks other than
programming, and the programs they produced frequently were
difficult to read and maintain [Weinwurm, 1970].

Structured Analysis was also developed to minimize the
difficulties that face the analyst, such as those mentioned
previously, in specifying system requirements and
specifications. It is structured in the sense that it
utilizes a set of tools and guidelines that have been
proven successful in other fields, most notably structured
programming and structured design [Yourdon and Constantine,
1978].

Structured Analysis involves the building of a model of
the system on paper. The model, being complex, has to be
partitioned into components that are organized for
comprehensible selective viewing and modification of the
system, as shown in Figure 3.1 [DeMarco, 1978]. That is, a
microscopic investigation of the system under study can be
made by subdividing each of the components into smaller
Figure 3.1 Graphical Representation of the System's Boundary and its Components.
component parts. This permits a detailed investigation of that component to determine possible design modifications. This micro-analysis of the system components is shown in Figure 3.2. Each component or level, in turn, is partitioned further until the activities to be carried out at that level can be fully described in terms of the fundamental constructs of programming.

Thus, a hierarchical system structure can be determined that permits a detailed analysis of the system at the various levels. This hierarchical structure is shown in Figure 3.3. This hierarchical structure permits the analysis of the system in terms of both higher- and lower-level systems. Furthermore, this structure provides the framework for analyzing the comprehensive aspects of the problem as well as the technical details that must be addressed and considered.

2.1 Primary Objectives of Structured Analysis [DeMarco, 1978 and Orr, 1981] (1)

(1) Accurately state and understand the users' requirements. Because it is fundamental, this goal is common to all methods of systems analysis. The statement of user requirements should be explicit, complete, precise, and specific.
Figure 3.2 Micro-Analysis of the System's Components.
Figure 3.3 Heirarchical System Structure.
(2) **Easily communicate the understanding of the system.** Arriving at a mutual understanding of users' requirements is achieved through interaction and communication. In practice, successful communication is a prerequisite to successful system development. It must occur among users, among analysts, and between users, analysts, and others involved in developing a system.

(3) **Prevent expensive mistakes.** In practice, the goal of a perfect statement of users' requirements is unlikely to be fully achieved. Nevertheless, the methods of systems analysis attempt to reduce the number of inconsistencies and errors as to minimize their impact on system development.

### 2.2 Secondary Objectives of Structured Analysis

[DeMarco, 1978 and Orr, 1981]

This section describes some of the secondary objectives that are necessary for meeting the primary objectives.

(1) **Decompose complex components into successively simpler ones.** That is, partition the system into lower level functions. These levels are also decomposed into simpler and simpler sub-levels. The beneficial effects of partitioning is that changes can be easily made and data can be easily altered without affecting other sections of
the specifications. At a very early stage of the project, the structured specifications provide a logical model of the processes and flow of data within the system. This model is easier to use and understand than traditional narrative specifications because it is concise, clear, and highly graphic.

(2) Minimize errors. Designers and programmers make fewer errors if the tools they use are well designed and organized. Where possible, techniques being employed should catch errors as soon as they are made, giving the designer or programmer immediate feedback.

(3) Control Complexity. It is vital to divide complex systems in an orderly fashion so that the humans can handle the complexity without errors. Control of complexity is achieved by minimizing the number of interactions among separate modules and standardizing the control structure.

(4) Achieve precise communication among people in a development team. One of the biggest sources of problems is mismatches in the work of different developers and programmers. The technique should minimize the need for complex interaction among team members and accomplish the interaction through formal representations, preferably computerized.
(5) Use diagramming techniques that are as clear as possible. Good diagrams are a great help in understanding complex problems.

(6) Achieve rigorous interfaces between separately developed modules. Most errors in very large projects are in program-program interfaces. Rigorous control of software interfaces requires more advanced structured techniques.

(7) Achieve sound data administration. Data administrations concerned with the correct structuring of data and the uniformity of data definition. When done well, this speeds up the analysis and design process.

(8) Achieve the maximum automation of system design with techniques that make the automatic generation of code possible. Structured techniques are steadily evolving into forms that permit automated checking, modification, and code generation. This automation of the design and programming process allows much higher quality code to be generated quickly.

The following sections illustrate the structured analysis techniques and the tools used for accomplishing the analysis task in a very constructive and systematic
manner.

3. STRUCTURED ANALYSIS TECHNIQUES

A system can be defined as a collection of components, connected by some type of interaction or interrelationship, which collectively respond to some stimulus or demand and fulfills some specific purpose or function [Coutinho, 1977]. In a system, each component responds to stimulation according to its intrinsic nature, but the actual stimulation it receives and its subsequent behavior is conditioned by the presence and interaction of the other system components. Therefore, the demands placed on a system call into play the individual behavior of the system components that collectively develop a synthesized composite behavior producing the system's response. In dealing with a system, it is possible to identify the following characteristics [Coutinho, 1977]:

1. There is some specific purpose or function that must be fulfilled or performed;

2. There are a number of components that can be identified as necessary ingredients of the problem. Furthermore, each component has a variety of attributes that implicitly, physically, and behaviorally are necessary for it's description;
3. The components are interrelated in some manner satisfying interface consistency between the components; and

4. There are constraints that restrict the system's behavior and the individual component response.

Hence, given that a system must function in a specific environment, which consists of the setting that contains or surrounds the system, and given that it has a purpose to accomplish, it is possible to describe what the system must do and what data are required. The system definition model (SDM), is such a description which remains true regardless of the technology used to implement the system. This model describes the function of each component of the system and how they interact with one another. It is also possible to describe a system as it might look on implementation with a particular technology. A model incorporating such a description is called the solution implementation model (SIM).

3.1 The System Definition Model (SDM)

The SDM consists of two parts: (1) Environmental Model and (2) Behavior-Response Model. The environmental model describes the environment in which the system operates and with which it must interact. It provides a description of
(a) the boundary between the system and its environment showing the interfaces between the two, and (b) the events that occur in the environment to which the system must respond. The behavior-response model describes the system’s behavior and response to events in the environment. It consists of two connected parts: (a) the data schema, which denotes graphically the data groupings required by the system and the relationship between data groups and (b) the transformation schema, which depicts graphically the layout of the transformations that operate on data flows within and outside the system boundary. In both these models, no assumptions are made regarding the technology to be used in the implementation stage.

As a description tool, the data schema is a graphics oriented abstraction on the use of nouns and verbs where the verbs describe the linkages between nouns. It shows the layout of information that must be remembered by the system for it to operate. The transformation schema is the active portion of the system that responds to events which occur in the environment. It models a system as a network of activities that transform input data into output data and produce messages.

3.2 The Solution Implementation Model (SIM)

The implementation model describes the system of
processors which are utilized to solve the problem defined by the essential model. Since the focus of this paper is on improving the productivity of the software development process, the methodology for constructing the implementation model will not be included. Details about the implementation model can be found in Ward & Mellor (1986).

4. STRUCTURED ANALYSIS TOOLS

In order to build a model of the system using the structured analysis approach defined above, special tools are used. Below is a summary describing these tool and the major notational symbols used. Most of the details related to the use and function of these tools and notational symbols will be clarified as they unfold, when a description of the model is viewed in the next chapter.

(1) A hierarchical set of Data Flow Diagrams (DFD) which graphically depicts the flows of data between the various activities that transform input data into output within the system. In other words, it portrays the partitioned system and its interfaces using multi-dimensional (levelled) diagrams to present the viewpoint of the data.
The major notational symbols used in drawing DFD are:

(a) Labelled arcs which represent data flow along a data path. Each arc shows where the data is originated from and what is its destination.

- The line represents a data path or interface between components. Each data flow is named for the data (Y) that travels along it.

(b) Circle which represent a data transformation or process that transforms input data into output.

- Each process is named using a naming convention, such as action verb (XYZ) and its direct object (usually the input or output data). B is an integer that represents the number of the transformation.
(c) Two parallel lines with a label in between represent a data base or data store/file. Each data store is named for its contents.

(d) A rectangular box, called a terminator, representing a source or sink of data, i.e. it is used to represent the various elements of the environment with which the system interacts. Each terminator is named after its origin or destination.
(e) Flows or arcs labelled with the prefix EV are used to signal the occurrence of an event; such flows have no data content.

- xyz represents an event.

(f) Circles labelled with the word "CONTROL" are used to represent control transformations. These processes have the task of controlling data transformations and, therefore, do not deal with data flows but rather with event flows.

- They are numbered with the letter "C". B represents the integer number(s) of the transformation(s) it controls.
(g) Unnamed arcs indicate the flow of data from or to data stores.

(2) Narrative Description: Is a description, in the form of English sentences and paragraphs, of how the data flows are transformed from input into output. It describes how the different components interact with one another. It also gives a detailed description of the function of each component and sub-components. Examples of how this may look can be seen in the following chapter.

(3) Data and Process Summaries (Data Dictionary): An ordered set of definitions of the terms used in the DFD to keep track of flows, data stores, data and control transformation processes and data elements.

(4) A structured methodology which uses the fundamental
constructs of programming for specifying the functional logic of a transformation primitive (a transformation which can no longer be decomposed or partitioned, or is no longer useful to partition any more). Sometimes referred to as "Structured English", the fundamental constructs of programming are structured English sentences used to describe (1) the activities (which transform input into output) performed by the primitives, (2) the conditions under which each activity takes place, and (3) the sequence by which the activities occur. The following is an example of how the fundamental constructs of programming may look:

IF (inventory level <= reorder point) THEN
    Order quantity = maximum inventory allowed - reorder point

ELSE
    IF (Inventory level > reorder point) THEN
        Order quantity = 0.

The above summary of the tools for structured analysis will be sufficient to understand the overall intent of the system. Most of the details related to partitioning and decomposition of data transformations will be clarified as they unfold, when a description of the partitioned system
is viewed along with the corresponding levelled diagram(s) in the next chapter. Details on the use of structured analysis are available from a number of references (DeMarco, 1978; Ward and Mellor, 1985; Orr, 1981).

The techniques and tools mentioned above can be used to perform the structured analysis process. However, they do not provide any means for (1) ensuring the consistency and accuracy of the data, (2) balancing* the data flow diagrams, (3) generating reports of the data, data stores, and data transformations whenever needed, (4) on-line documentation of the data, and (5) automating the clerical aspects of the analysis process. These capabilities are essential to ensure that (1) the developed specifications clearly define the function of each component, (2) the FSM corresponds as closely as possible to the real problem, (3) the definition of each event, transformation, data flow, and data store is accurate and correct, (4) the developed solution is for the right problem. They are also essential to (1) reduce the time it takes the analyst to perform the analysis process and document it and (2) drastically reduce the effort required by the analyst to illustrate how the

* Balancing is a process which checks whether the input and output data flows of an upper level diagram are consistent with the lower level diagrams.
system's modular functions fit together, and how the different components interact with one another. This effort is reduced by automating the documentation of the model specifications and all the clerical aspects of the analysis process.

Hence, it is clearly obvious that the means mentioned above are very important to increase the productivity of the analysis and development process. For that purpose, a computer-aided design package, known as Structured Architect (SA), was used to develop the system specifications and overcome the obstacles and problems, mentioned above and in previous sections, that may face the analyst.

Structured Architect was used because it (1) provides the necessary tools to perform a structured analysis of the system and develop it's specifications, ensure the consistency and accuracy of the data, and generate up-to-the-minute reports on the data, (2) improves the quality and productivity of the analyst's work by automating the clerical aspects of structured analysis, (3) automates the process of drawing and editing of the data flow diagrams, (4) accommodates multi-level diagrams and manages the analysis of large and complex systems, (5) allows extensive on-line documentation of the data, and (6) provides
extensive analysis and reporting such as balancing (defined previously) the data flow diagrams, and generating reports of the data, data stores, and data transformations. In addition, after the completion of the analysis and specifications development, the analysts and the end-user can participate in walkthroughs in which the developed specifications are tested for accuracy. It allows them to follow the data through the system; does it really follow the paths depicted on the diagrams? Also, it allows them to check the decomposition of the components and the data; are they consistent throughout the different levels?
structured development of the FSM specifications

The second goal of this thesis is to build the flexible simulation model (FSM) in a modular fashion to provide the best foundation for modifying the FSM to other types of production and distribution systems (PDS). Chapter III described the methodology of structured analysis and development which is used to accommodate this goal. This chapter, addresses this goal by describing how this methodology is used to develop the specifications of the FSM. The first section provides a list of the major activities performed before, during, and after the process of developing these specifications. The second section provides a set of data flow diagrams representing the model's architecture. This section also provides a detailed narrative description of the definition and function for every object depicted in the data flow diagrams.

1. specifications development process

The specifications development process involves three major phases. The first phase consists of acquiring information about the system. The second phase involves
analyzing this information in a structured fashion. The major step in this structured analysis are: (1) Generate data flow diagrams, (2) Partition complex diagrams, (3) Build the narrative description file, and (4) Create the data summary and process summary files. The third and final phase involves verifying the model.

1.1 Acquire Information About the System

Personnel who use or are affected by the system should be observed and interviewed to obtain a clear and thorough understanding of the functions and processes involved. Some of the techniques for acquiring the information needed are:

1. On-site Observation: Watch the user or the expert solve the problem on the job;
2. Problem discussion: Explore the kinds of data, knowledge, and procedures needed to solve specific problems;
3. Problem description: Have the user describe a prototypical problem for each component of the system.

1.2.a Generate Data Flow Diagrams

The information acquired during the first phase is used to generate a set of data flow diagrams which represent the
system. A data flow diagram graphically depicts the flow of data as it enters, passes through, and exists the system. Data flow diagrams consist of the notational symbols described in chapter three (arcs, circles, parallel lines, and boxes).

1.2.b Partition Complex Diagrams

Each process (transformation) represented within a data flow diagram may be decomposed into subordinate diagrams. The complete set diagrams depict all processes, sources, and destinations of data, files, databases, and data flows. A set of diagrams is composed of a top-level context diagram and a number of middle-level, intermediate-level, and lower-level diagrams.

The context diagram outlines the scope of the analysis and consists of only one process with the net input and output data flows and relevant sources and sinks. Figure 4.1 represents the context diagram of the integrated production planning and inventory control (IPPIC) system which will be discussed later. Middle-level diagrams depict the flow of data in successively greater detail when moving down from the context diagram. Middle-level diagrams (parent diagrams) can be decomposed further into lower-level diagrams (child diagrams), with each level reflecting more detail than the level above.
FIGURE 4.1 CONTEXT.
Those transformations that do not need to be decomposed further are called transformation primitives. The function of each transformation primitive is documented by structured functional logic (structured English sentences contained in the data dictionary described below). Details regarding this process can be found in DeMarco (1978).

1.2.c Build the Narrative Description File

This file gives a description of how the data flows are transformed from input into output. It describes how the different components interact with one another. It also gives a detailed description of the function of each component and its sub-components.

1.2.d Create the Data Summary and Process Summary Files (Data Dictionary)

This file, listed in Appendix B, contains the precise definitions for every object depicted in the data flow diagrams. For each object representing data flows, data stores, transformations, and terminators/sinks within a diagram, a general description is entered using the following convention:

1. Data flows. In addition to the description, origin/destination information and structure are provided. The structure contains the components
(elements) of the data. Data primitives are those data components that cannot be decomposed further. The following notation is used to show how simple data components are combined to form more complex components:

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>is equivalent to</td>
</tr>
<tr>
<td>+</td>
<td>and</td>
</tr>
<tr>
<td>[ ; ]</td>
<td>either/or</td>
</tr>
<tr>
<td>( )</td>
<td>iterations of</td>
</tr>
<tr>
<td>( )</td>
<td>optional</td>
</tr>
</tbody>
</table>

The above symbols are used to describe the structure and contents of both the data flows and data stores. Below is an example of how the above symbols are used to describe "Products Inventory":

\[
\text{Products-Inventory} = [\text{product name} : \text{product code}] \\
+ \text{product identification number} \\
+ \text{quantity in inventory} \\
+ (\text{product color, size, shape}) \\
+ (\text{shelf number})
\]

2. Data stores. In addition to the data flow information, this file specifies, for each data store, which transformations update the data store and
which transformations retrieve data from the data store. The symbols mentioned in (1) above are also used to define the structure of data stores. Examples of such definitions are provided in the data dictionary file in Appendix A.

3. Transformations (processes). In addition to the general description, it includes a list of the data entering and the data leaving the transformation, the parent process, and the child processes (if any). For each transformation primitive, the functional logic is provided.

1.3 Verify the Model

Finally, the analyst and the end-user can participate in walkthroughs. This process (walkthroughs) involves going through the developed specifications step by step, and closely studying them, to check for the consistency, completeness accuracy, and redundancy of the data flows.

Together, the analysts and the end-user, can check whether the data really follows the paths depicted on the data flow diagrams. Process description and data structure can also be tested for accuracy and consistency. If any errors are discovered, they can easily be corrected and modified.
2. Data Flow Diagrams: Partitioning and their Narrative Description

This section provides the set of data flow diagrams which represent the architecture of the IPPIC system. In order to understand the specifications developed, a detailed explanation (which is referred to as the narrative description) of the structured system definition model (SDM) is presented for each data flow diagram. As mentioned earlier, this narrative description outlines the objective or purpose of each of the components of the system. It also illustrates how each process or component of the system transforms data. The "IPPIC System" will hereafter be called "the system" for the sake of simplifying the discussion.

As mentioned in the previous chapters, the first step is to define the context of the environment in which the system is embedded in order to draw a proper boundary for the system. The context diagram, depicted in Figure 4.1 shows the system and its environment. This diagram is the highest level diagram wherein the overall functioning of the system is represented by a single transformation, IPPIC-OPERATIONS. The three terminators (sources/sinks), the USER, MARKET-DEMAND-SOURCE, and RAW-MATERIALS-SUPPLIER, represent the environment in which the system operates. Note that the various interactions between the system and
its environment are shown by the arcs (data flows) which represent the data transfers. These data flows will be defined throughout the following sections. Also note that the number of the transformation IPPIC-OPERATIONS is (0).

The notations used in Figure 4.1 and in the rest of the diagrams shown in this chapter were defined in chapter 3.

Let us first consider the environment (the 3 terminators) in which the system operates as shown in the CONTEXT diagram in Figure 4.1.

(CONTEXT):

Terminator 1: USER.

The user represents anyone who is using the system (end-user). Here it can be taken to represent the management in charge of the IPPIC system operations. REPORTS-REQUEST (data leaving the terminator) represents the request for the various reports from both the warehouses and the production facility. These reports include inventory reports from the warehouses, and production and inventory reports from the production facility. The reports are used by the end-user to evaluate the performance of the system.

REPORTS (data entering the terminator) represent the
set of reports produced by the warehouses **WAREHOUSES-REPORTS** and the production facility (DPMF-REPORTS, where DPMF represents discrete parts manufacturing facility) requested by the user. **WAREHOUSES-REPORTS** and **DPMF-REPORTS** are shown on diagram 0 in section D-0. The structure of these reports and others is provided in the data summary file in Appendix B.

**STANDARDS** in the data flow **STANDARDS-POLICIES** (data leaving the terminator) represent the specifications, bill of materials and costs for the classifications and coding of the products by code, name, and production operations. The term **POLICIES** refers to the inventory policies to be used by the warehouses for controlling the inventory of the products, and by the production facility for controlling the inventory of the raw materials and finished products. The various policies are:

1) Products inventory policy
2) Raw materials inventory policy
3) Scheduling policy

The **EV-TIME** data flow (leaving the terminator) represents an event flow to be used in the operations of certain fixed time-based production and inventory policies as will be seen in later sections.
The **DESIRED-CHANGES** data flow represents any changes to the system operations issued by the end-user. These may include changes in the production policies, inventory policies, products standards, cost parameters, etc.

**Terminator 2: MARKET-OR-FIELD.**

The data leaving the terminator, **MARKET-DEMAND**, represents the information about the market demand for products. This would include the type of product and quantity. The other data flow leaving the terminator, **SALES**, represents the actual sales of products at the market.

The data entering this terminator, **PRODUCTS-DISPATCHES**, represents the products shipped to the market from the warehouse to satisfy the needs (demand) of the consumers.

**Terminator 3: RAW-MATERIALS-SUPPLIER.**

The data entering the terminator is the **MATERIALS-ORDER**. This data represents the requisition sent to the raw materials supplier issued by the production facility. The orders provide a list of the raw materials needed by the production facility.

The **MATERIALS** data flow (leaving the terminator) represents the materials shipped to the production facility.
from the suppliers in response to an earlier MATERIALS-ORDER.

The following narrative description will be divided into structured sections. Each section represents one data flow diagram (DFD), with the section’s number and name representing the number and name of its corresponding data flow diagram. This division provides a better understanding of the data flow diagrams, and makes it easier to refer back and forth between the diagram and its narrative description. In addition, for data flows that come from a higher level DFD, a bracket notation will be used to identify the DFD that introduced these data flows. This notation looks as follows: [ID,P]. Where ID represents the identification of the DFD that introduced the data flow and P represents the page number for the definition of the DFD.
D-0 (IPPIC-OPERATIONS).

The context diagram presented a gross picture (a top overall view) of the system. Diagram 0 gives a more detailed picture of the system, IPPIC-OPERATIONS. Note that there are no terminators or sinks in this second level diagram.

Looking at this diagram, the IPPIC-OPERATIONS represented by a single process in the context diagram is broken down into two processes, namely WAREHOUSES-OPERATIONS and DPMF-OPERATIONS (where DPMF represents discrete parts manufacturing facility).

1. **Warehouses-Operations** represents the operations performed at the warehouses. These operations include forecasting the demand for products, dispatching the products to the market (PRODUCTS-DISPATCHES [CONTEXT, 89]), ordering and controlling the inventory of products (PRODUCTS-ORDERS), and producing the reports requested by the user (WAREHOUSES-REPORTS). These operations are detailed at lower level diagrams. The WAREHOUSES-INVENTORY-Policies data store contains the information about the inventory policies to be used by the warehouses, which is created by the user. These policies are chosen by the user (management).
1. Warehouses-Operations
   - products-dispatches
   - reports-request
   - sales
   - market-demand
   - warehouses-reports

2. DPMF-Operations
   - reports-request
   - EV-time
   - materials-order
   - materials
   - DPMF-reports

Products-orders
products-shipments
forecasted-demand

Scheduling-Sequencing-Policies

IPPIC-OPERATIONS (0).
2. DPMF-Operations represents the operations performed by the production facility. As defined previously, DPMF represents discrete parts manufacturing facility. These operations include shipping products to the warehouses (PRODUCTS-SHIPMENTS [D-0, 98]), ordering and controlling the inventory of raw materials (MATERIALS-ORDER [CONTEXT, 89]), generating reports for management (DPMF-REPORTS [D-0, 98]), and production scheduling and inventory control of the products. MATERIALS [CONTEXT, 89] are received in response to an earlier MATERIALS-ORDER. The DPMF-INVENTORY-POLICIES data store contains the information necessary for controlling the inventory of the products and the raw materials. The PRODUCTS-STANDARDS data store contains the data about the products' specifications, bill of materials, etc. SCHEDULING-SEQUENCING-POLICIES data store contains the information necessary for performing the production scheduling and sequencing, where scheduling is the assignment of jobs to time blocks and sequencing is the determination of the exact order in which the jobs in a given time block are to be produced. Again, the operations of this transformation are detailed at the lower level diagram to avoid redundancy.

From the diagram it can be seen how the two transformations are connected or interlinked by three data flows; namely PRODUCTS-ORDERS, PRODUCTS-SHIPMENTS, and FORECASTED-DEMAND. The WAREHOUSES-OPERATIONS transforma-
tion orders products (PRODUCTS-ORDERS) and receives (PRODUCTS-SHIPMENTS) from the production facility, and Vis-Versa.

If a look is taken at diagram 0 and the context diagram, it is found that many of the data flows are the same and the meaning of the data flows and data stores that are repeated is the same. Therefore, as the decomposition progresses, the meaning and contents of such data will not be repeated. The reader should refer to the data summary file (Appendix A) for more details about the data structure.

D-1 (WAREHOUSES-OPERATIONS).

This diagram decomposes Transformation 1 (WAREHOUSES-OPERATIONS) shown in diagram 0. As mentioned, this transformation represents the operations performed at the warehouses. Note that each vector (arc) associated with the WAREHOUSES-OPERATIONS process are shown on this immediate lower level diagram also. Again, these vectors have the same meaning on both diagrams and other lower level ones.

WAREHOUSES-OPERATIONS consists of 4 transformations that perform the operations mentioned above namely

1) Identify-Demanded-Product

2) Update-Files

3) Orders-Inv-Operations.
WAREHOUSES-OPERATIONS (1).
4) Forecast-Demand.

1) Identify-Demanded-Product:

Once the MARKET-DEMAND [CONTEXT, 89] for products is received at the warehouses, it is broken down into FORECASTABLE-DEMAND and UNFORECASTABLE-DEMAND. After receiving the data regarding the UNFORECASTABLE-DEMAND, this process accesses WAREHOUSES-PRODUCTS-FILES (this data store contains a list of the existing products and not their inventory levels) to check if the list includes a demand for new products that have not been ordered before. If any new products exist in that list, then two actions are taken:

1. information regarding the new products is stored in NEW-PRODUCTS-DEMANDED data store, and
2. a signal, EV-NEW-PRODUCT, is sent to the transformation UPDATE-FILES indicating that new products are under demand.

If the demand does not include orders for new products, the list DEMANDED-PRODUCTS is sent to ORDERS-INV-OPERATIONS.

2) Update-WH-Files:

The word "WH" in the title stands for warehouses. When
the signal EV-NEW-PRODUCT is received from process 1, this process accesses the data store NEW-PRODUCTS-DEMANDED. The information obtained from NEW-PRODUCTS-DEMANDED is then used to update the WAREHOUSES-PRODUCTS-FILES, i.e., to add the new product to the warehouses' files. These files contain updated lists of all the current products. These lists do not include inventory levels, codes, standards, etc.

3) Orders-Inv-Operations:

This transformation is responsible for generating reports, receiving and dispatching products, and inventory control. When an order for products is received (DEMANDED-PRODUCTS), this transformation dispatches the products (PRODUCTS-DISPATCHES [CONTEXT, 89]) to the market if they are available in inventory. If they are not available in inventory, PRODUCTS-ORDERS [DO, 98] are sent to the manufacturing facility. EV-TIME [CONTEXT, 89] indicates the actual date and time used for checking and updating the status of orders and inventory. The transformation accesses WAREHOUSES-INVENTORY-POLICIES [D-0, 98] to retrieve the information regarding the policy to be used for controlling the inventory of products. EV-TIME [CONTEXT, 89] is specifically used when this policy is the periodic-review policy. This will be clarified when this transformation is detailed further in section D-1.3.
PRODUCTS-SHIPMENTS [DO, 98] are received in response to an earlier PRODUCTS-ORDERS [DO, 98].

The data WAREHOUSE-REPORTS [DO, 98] are generated in response to a REPORTS-REQUEST [CONTEXT, 89]. The data store NEW-PRODUCTS-DEMANDED is accessed whenever a demand for products is received to retrieve the data needed about those new products.

4. Forecast-Demand:

This transformation is responsible for forecasting the demand for products. In order to perform the forecasting properly, this transformation uses SALES [CONTEXT, 89], FORECASTABLE-DEMAND, and PRODUCTS-DISPATCHES [CONTEXT, 89] data in addition to the data retrieved from WAREHOUSES- INVENTORY-POLICIES [DO, 98]. EV-TIME [CONTEXT, 89] is used as a timer which indicates the date and time as mentioned earlier. Once all the data needed are available, the FORECASTED-DEMAND [DO, 98] is generated. More details for this transformation are given in section D-1.4.

D-1.3 (ORDERS-INV-OPERATIONS).

As can be seen, ORDERS-INV-OPERATIONS is broken down into 3 different transformations namely RECEIVE-DISPATCH- PRODUCTS, INV-CONTROL-OPERATIONS, and GENERATE-WH-REPORTS.
ORDERS-INV-OPERATIONS (1.3).
1. Receive-Dispatch-Products:

This transformation performs the receiving and dispatching operations of products. Once a PRODUCTS-SHIPMENT is received, it stores the information about the received products in the WAREHOUSES-PRODUCTS-INVENTORY data store. This data store contains the information regarding the inventory level of each product. When orders for products (DEMANDED-PRODUCTS [D-1, 101]) are received, it accesses WAREHOUSES-PRODUCTS-INVENTORY. If the products are available, it generates PRODUCTS-DISPATCHES [CONTEXT, 89] and the respective information is sent to INV.-CONTROL-OPERATIONS. If the products are not available, it sends a signal (EV-PRODUCT-NOT-AVAIL.) to INV.-CONTROL-OPERATIONS indicating insufficient inventory status. Every time a dispatch is made or a shipment is received, this transformation also updates the data store WAREHOUSES-ORDERS-STATUS. This data store contains information regarding the status of the orders. This transformation is detailed in section D-1.3.1.

2. Inv.-Control-Operations:

This transformation is responsible for controlling the inventory of products at the warehouses. Such responsibilities include applying the inventory policy selected by the user to control inventory status, and placing orders for products when needed.
When the signal EV-PRODUCT-NOT-AVAIL. is received, PRODUCTS-ORDERS [DO, 98] are generated which indicates the products needed and their respective quantities. It accesses WAREHOUSES-PRODUCTS-INVENTORY to determine which products should be ordered. If products have been dispatched, the data regarding the DISPATCH-INFO. [D-1.3, 105] is sent to this transformation to check the inventory status of these products and issue replenishment orders if necessary. The information regarding these unfilled orders are then stored in WAREHOUSES-ORDERS-STATUS data store. It also accesses NEW-PRODUCTS-DEMANDED [D-1, 101] to check if new products have been ordered. If so, an order for those products is sent to the production facility (PRODUCTS-ORDERS [DO, 98]).

To determine which inventory policy to use, INV-CONTROL-OPERATIONS accesses the data store WAREHOUSES-INVENTORY-POLICIES [DO, 98] to obtain that information. This information contains the policy to be used and its parameters.

This transformation is detailed more in section D-1.3.2

3. Generate-Reports:

The only function of this transformation is to generate reports regarding the products' inventory levels and the status of orders at the warehouses. After receiving a REPORTS-REQUEST [CONTEXT, 89] it accesses the data store
WAREHOUSES-PRODUCTS-INVENTORY and the WAREHOUSES-ORDERS
STATUS. It also accesses the WAREHOUSES-INVENTORY-POLICIES
[DO, 98] to specify which of the inventory policies is
currently being implemented.

D-1.3.1 (RECEIVE-SHIP-PRODUCTS)

This diagram shows a more detailed picture of the
RECEIVE-SHIP-PRODUCTS transformation depicted in diagram
1.3. As mentioned, this transformation represents the
operations of shipping and receiving products at the
warehouses. As can be seen on the diagram, it consists of
five main processes namely CHECK-INVENTORY, DISPATCH-
PRODUCTS, UPDATE-INVENTORY, RECEIVE-FINISHED-PRODUCTS, and
UPDATE-ORDERS-STATUS.

1. Check-WH-Inventory:

Once an order for products is received (DEMANDED-
PRODUCTS [D-1, 101]), it accesses the data store WAREHOUSES-
PRODUCTS-INVENTORY [D-1.3, 105] to check the availability of
products in inventory. If the products are available, a
signal EV-PRODUCT-AVAILABLE is sent to the process DISPATCH-
PRODUCTS. Otherwise, the signal EV-PRODUCT-NOT-AVAIL
[D-1.3, 105] is sent to INV.-CONTROL-OPERATIONS (as shown
on the upper level diagram, 1.3) which indicates that
insufficient inventory of the product exists.
WAREHOUSES-PRODUCTS-INVENTORY

EV-product-available

EV-product-not-avail.

WAREHOUSES-ORDERS-STATUS

Demand-products

Dispatch-products

Dispatch-info.

Dispatch-info.

Updates-warehouse-Inventory

Received-products

Received-orders

Update-warehouse-Orders-Status

RECEIVE-DISPATCH-PRODUCTS (1.3.1).
2. Dispatch-Products:

Upon receipt of the signal EV-PRODUCT-AVAILABLE, this process dispatches the products to the market (PRODUCTS-DISPATCHES [CONTEXT, 89]). It then sends the information regarding the dispatches made (DISPATCH-INFO. [D-1.3, 105]) to the process UPDATE-WH INVENTORY.

3. Update-WH-Inventory:

This process updates the products' inventory levels at the occurrence of RECEIVED-PRODUCTS or DISPATCH-INFO [D-1.3, 105], or both. If products are dispatched, it accesses WAREHOUSES-PRODUCTS-INVENTORY [D-1.3, 105] and reduces the level of inventory by the amount dispatched. On the other hand, if products are received (RECEIVED-PRODUCTS) from the production facility, it accesses WAREHOUSES-PRODUCTS-INVENTORY and increases the level of inventory by the quantity received.

4. Receive-Finished-Products:

This process receives the PRODUCTS-SHIPMENTS [DO, 98] from the production facility and identifies the products on the received list. Then it sends the information regarding the received products to UPDATE-WH-INVENTORY, and the information regarding the corresponding orders (RECEIVE-ORDERS) to UPDATE-WH-ORDERS-STATUS.
5. Update-Orders-Status:

Whenever shipments have been received that correspond
to earlier orders, this process updates the status of these
orders in the WAREHOUSES-ORDERS-STATUS [D-1.3, 105] data
store. This data store contains information regarding the
status of orders that the warehouses sent to the production
facility.

D-1.3.2 (INV.-CONTROL-OPERATIONS).

This diagram decomposes INV.-CONTROL-OPERATIONS further
to show its several components (operations). Looking at
Diagram 1.3.2, this transformation is broken down into 4
main processes and one control process. As mentioned
earlier, this transformation controls the products' 
inventory levels and issues orders whenever necessary.

1. Identify-WH-Policy

With the occurrence of DISPATCH-INFO. [D-1.3, 105]
and/or DEMANDED-PRODUCTS [D-1, 101], this component
accesses the WAREHOUSES-INVENTORY-POLICIES [DO, 98] data
store to identify which of the policies is to be used for
controlling the products at the warehouses. Once that
information is obtained, it is sent to the CONTROL process.
If the policy to be used is the On-Demand policy (which
basically assumes zero inventory), then the signal
EV-ON-DEMAND is sent to the control process, which in turn
1. Identify WH Policy
   - EV-on-demand
   - EV-other
   - demanded-products

2. Apply WH On-Demand Policy
   - demanded-products

3. Apply WH Inventory Policy
   - EV-time
   - dispatch-info.

4. Process Orders for Products
   - needed-products
   - products-orders

CONTROL
1.3.2.2
1.3.2.3

New Products Demanded
EV-ENAB.-DISAB.

Warehouse-Products-Inventory

INV.-CONTROL-OPERATIONS (1.3.2)
enables the process APPLY-WH-ON-DEMAND-POLICY. Otherwise, the signal EV-OTHER is sent to the CONTROL process, which in turn enables the process APPLY-WH-INVENTORY-POLICY.

2. Apply-WH-On-Demand-Policy:

If this transformation is enabled by the CONTROL process, then it generates a list (NEEDED-PRODUCTS) for the DEMANDED-PRODUCTS [D-1, 101]. It also generates the list of NEEDED-PRODUCTS whenever products are not available (EV-PRODUCT-NOT-AVAIL [D-1.3, 105], which is received from CHECK-WH-INVENTORY on diagram 1.3.1) or new products have been ordered.

3. Apply-WH-Inventory-Policy:

This process applies the policy to be used for controlling the products' inventory levels. Once it is enabled by the CONTROL process, it accesses the WAREHOUSES-PRODUCTS-INVENTORY [D-1.3, 105] data store and decides whether to order products or not. If it is necessary to order products, then a list of the NEEDED-PRODUCTS is generated.

4. Process-Orders-for-PRODUCTS:

Once this transformation receives the list of NEEDED-PRODUCTS from either process 2 or 3, it prepares the orders, files them in the WAREHOUSES-ORDERS-STATUS [D-1.3,
data store, and the PRODUCTS-ORDERS [DO, 98] to the production facility.

D-1.3.2.3 (APPLY-WH-INVENTORY-POLICY)

This diagram presents the decomposition of the transformation APPLY-WH-INVENTORY-POLICY shown in diagram 1.3.2. This transformation is decomposed into three main operations, namely, IDENTIFY-WH-INV.-POLICY, APPLY-WH-PERIODIC-POLICY (where WH means warehouses), and APPLY-WH-CONTINUOUS-POLICY, and a CONTROL PROCESS.

1. Identify-WH-Inv.-Policy:

Once this process is enabled by EV-ENAB.-DISAB. [D-1.3.2, 112], it accesses the data store WAREHOUSES-INVENTORY-POLICIES [DO, 98] to identify which of the specific inventory policies to use for controlling the inventory of products. If the policy to used is the periodic policy, then the signal EV-PERIODIC is sent to the CONTROL process. The CONTROL process, in turn, enables APPLY-WH-PERIODIC-POLICY. Otherwise, the signal EV-CONTINUOUS is sent to the CONTROL process. The control process, in turn, enables APPLY-WH-CONTINUOUS-POLICY.

2. Apply-WH-Periodic-Policy:

As this transformation is enabled by the CONTROL process, it applies the periodic policy to control the
1 Identify-WH-Inv-Policy

EV-ENAB.-DISAB.

EV-Continuous

EV-periodic

C2 CONTROL-1.3.2.3.2-1.3.2.3.3

EV-ENAB.-DISAB.

EV-ENAB.-DISAB.

2 Apply-WH-Periodic-Policy

EV-time

needed-products

warehouse-Products-Inventory

3 Apply-WH-Continuous-Policy

needed-products

dispatch-info.
products’ inventory. At the end of each review period, which is determined by EV-TIME [CONTEXT, 89], the data store WAREHOUSES-PRODUCTS-INVENTORY [D-1.3, 105] is accessed. If any of the products have insufficient inventory levels, a list is generated for those products. This list contains the type of products and their respective quantities. The list of NEEDED-PRODUCTS [D-1.3.2, 112] is then sent to the production facility.

3. Apply-WH-Continuous-Policy:

If this transformation is enabled by the CONTROL process, it applies the continuous policy to control the inventory of products. On a continuous basis and/or when DISPATCH-INFO. [D-1.3, 105] occurs, the data store WAREHOUSES-PRODUCTS-INVENTORY [D-1.3, 105] is accessed to check for insufficient inventory levels. If any are detected, it generates a list of the NEEDED-PRODUCTS [D-1.3.2, 112] which contains the type of products needed and their respective quantities.

**D-1.4 (FORECAST-DEMAND)**

The FORECAST-DEMAND transformation shown in diagram 1 can be broken down into 4 lower level transformations.

1. Implement-Forecasting-Model:

This transformation implements the forecasting model to
FORECAST-DEMAND (1.4).
be used for predicting the demand for products. In order to function properly, it accesses the data stores MARKET-INVENTORY, WAREHOUSES-INVENTORY-POLICIES [DO, 98], and ANALYZED-DATA to obtain the necessary data. This transformation is signaled (EV-DATA-ANALYZED) to start once the data has been analyzed.

2. Analyze-Historical-Data:

This transformation analyzes the data in the HISTORICAL-DISPATCHES data store. This data contains the history of actual dispatches of products from the warehouses to the market. Once the data is analyzed, it updates the data store ANALYZED-DATA and signals IMPLEMENT-FORECASTING-MODEL to start. EV-TIME [CONTEXT, 89] is a trigger to start this transformation.

3. Update-Market-Inventory:

Upon the receipt of the data regarding the actual SALES [CONTEXT, 89] of products at the market and the PRODUCTS-DISPATCHES [CONTEXT, 89] from the warehouses to the market, this transformation updates the data store MARKET-INVENTORY. This data store contains the information regarding the inventory of products at the market.

4. Update-Historical-Data:

For every dispatch of products (PRODUCTS-DISPATCHES
from the warehouses to the market or FORECASTABLE-DEMAND [D-1, 101], this transformation stores the dispatch information in the data store HISTORICAL-DISPATCHES.

The following pages provide the narrative description of the data flow diagrams which detail the DPMF-OPERATIONS transformation mentioned earlier in section D-0.
D-2 (DPMF-OPERATIONS)

As mentioned earlier, this transformation represents the operations performed at the discrete parts manufacturing facility (DPMF). This diagram decomposes this transformation further to show the details of the operations performed. The three main processes as shown on diagram 2 are ORDERS-INV.-OPERATIONS, PRODUCTION-PLAN.-CONTROL, and RAW-MATERIALS-CONTROL.

1. Inv-Orders-Operations:

This transformation has three main functions: (1) it receives the orders from the warehouses and ships the respective products, (2) controls the inventory of products, and (3) generates the requested reports.

Once PRODUCTS-ORDERS [D-O, 98] are received, it checks if the products ordered are in inventory by accessing DPMF-PRODUCTS-INVENTORY data store. If products are not available, it issues PRODUCTION-ORDERS for those products. Otherwise, it produces PRODUCTS-SHIPMENTS to be sent to the warehouses.

To determine if insufficient inventory levels exist, this transformation accesses the data stores DPMF-INVENTORY-POLICIES [D-O, 98], to determine which policy to use, and DPMF-PRODUCTS-INVENTORY to obtain the necessary information about the inventory levels.
DPMF-OPERATIONS (2)
DPMF-REPORTS [D-0, 98] is generated in response to a request made earlier for reports. The generated reports include reports regarding the raw materials (MATERIALS-REPORTS), inventory and production reports (PRODUCTION-REPORTS), and the status of orders.

Whenever the signal EV-PRODUCTION-COMPLETED is received, indicating that the production of an earlier PRODUCTION-ORDER has been completed, it accesses DPMF-PRODUCTS-INVENTORY and generates PRODUCTS-SHIPMENTS.

Upon the receipt of the FORECASTED-DEMAND [D-0, 98], which indicates the type of products, quantity, and due date of each product that is predicted, this transformation processes this forecast and issues production orders for each product. This order would indicate exactly what is needed and when.

Whenever NEW-STANDARDS for the products are issued by the user, this transformation updates PRODUCTS-STANDARDS [D-0, 98] data store with the new changes. This data store contains the standards (specifications) of each product.

2. Production-Plan.-Control:

The word "Plan." in the title represents Planning. This transformation performs the steps required to control the production of the products. These steps range from grouping and sequencing jobs to the final step of production completion.
As it receives the PRODUCTION-ORDERS, this transformation checks if the raw materials needed are available in RAW-MATERIALS-INVENTORY. If they are, then it groups, sequences, and schedules the jobs for production. Once production is completed, the finished products are stored in DPMF-PRODUCTS-INVENTORY and a massage (EV-PRODUCTION-COMPLETED) is sent to process 1 as mentioned earlier. The same massage is also sent to RAW-MATERIALS-CONTROL to indicate to it that raw materials have been taken out of inventory. If the raw material needed was not available in inventory, the signal EV-MTL-NOT-AVAILABLE is sent to RAW-MATERIALS-CONTROL so it can order the material needed.

The signal EV-MTL-ARRIVED indicates triggers this transformation of the arrival or raw materials that was needed earlier. Upon that arrival, the necessary steps for performing the production operations is carried out.

To carry out the necessary production scheduling and sequencing operations, it accesses the data store SCHEDULING-SEQUENCING-POLICIES [D-0, 98] to obtain the necessary information. It also accesses PRODUCTS-STANDARDS [D-0, 98] to obtain the necessary information regarding the products' specifications.

This transformation also generates the reports regarding the status of production (PRODUCTION-REPORTS) at the DPMF. These reports are generated in response to an
earlier REPORTS-REQUEST [CONTEXT, 89].

3. Raw-Materials-Control:

This transformation mainly represent the operations involved in control the inventory status of raw materials and generating raw materials related reports. These reports include information about the inventory levels of the different raw materials, and the status of orders issued for raw materials.

For controlling the inventory, RAW-MATERIALS-CONTROL accesses the data stores DPMF-INVENTORY-POLICIES [D-0,98], to obtain the information regarding the policy to be used, and RAW-MATERIALS-INVENTORY to determine if any insufficient inventory levels exist. If this is the case, then a MATERIALS-ORDER [CONTEXT, 89] is issued for the materials needed.

MATERIALS [CONTEXT, 89] contains the list of the raw materials received from the suppliers. This list corresponds to an earlier MATERIALS-ORDER [CONTEXT, 89].

D-2.1 INV-ORDERS-OPERATIONS

This diagram provides a more detailed picture of ORDERS-INV.-OPERATIONS shown in diagram 2. As mentioned, this transformation represents the operations involved in controlling the inventory of products, processing the orders for products, and generating the requested reports.
INV-ORDERS-OPERATIONS (2.1).
Looking at diagram 2.1, this transformation is decomposed further into four main processes.

1. Identify-Ordered-Product:

Once the list of PRODUCTS-ORDERS [D-0, 98] and/or FORECASTED-DEMAND [D-0, 98] are received at the manufacturing facility, this process accesses DPMF-PRODUCTS-FILES (this data store contains a list of the existing products and their description, but does not include inventory levels, specifications, etc.) to check if the list includes an order for new products that have not been ordered before. If any new products exist in that list, then two actions are taken:

1. information regarding the new products is stored in NEW-PRODUCTS-ORDERED data store, and
2. a signal, EV-NEW-PRODUCT, is sent to the transformation UPDATE-DPMF-FILES indicating that new products are under demand.

Whether new products were on the list or not, the list of the products that exist (PRODUCTS-ORDERED) is sent to INV.-CONTROL-SHIPMENTS.

2. Update-DPMF-Files:

When the signal EV-NEW-PRODUCT is received from process 1, this process accesses the data store NEW-PRODUCTS-
ORDERED. The information obtained from this data store is then used to update the DPMF-PRODUCTS-FILE and PRODUCTS-STANDARDS [D-0, 98] data stores.

3. Shipments-Inv.-Control:

This transformation represents the several operations involved in shipping and controlling the inventory levels of products. Upon receipt of the list of PRODUCTS-ORDERED, it generates PRODUCTS-SHIPMENTS if they are available in DPMF-PRODUCTS-INVENTORY [D-2, 121]. If products are not available, it stores the corresponding orders in the DPMF-OUTSTANDING-ORDERS data store.

Whenever the signal EV-PRODUCTION-COMPLETED [D-2, 121] is received, the status of the corresponding orders is updated in the DPMF-OUTSTANDING-ORDERS data store, and the PRODUCTS-SHIPMENTS are generated. PRODUCTION-ORDERS [D-2, 121] are generated either to update inventory levels, or to satisfy orders that cannot be fulfilled from inventory.

4. Generate-DPMF-Reports:

The only function of this transformation is to generate reports regarding the production status, products' inventory levels, and the status of products orders in response to a REPORTS-REQUEST [CONTEXT, 89]. To generate these reports it accesses the data store DPMF-PRODUCTS-INVENTORY [D-2, 121] and the DPMF-OUTSTANDING-ORDERS. The
reports regarding production (PRODUCTION-REPORTS [D-2,121]) are obtained from GENERATE-DPMF-REPORTS [CONTEXT, 89] mentioned in section D-2.1.

D-2.1.2 (UPDATE-DPMF-FILES)

A more detailed picture of this transformation is shown here. As mentioned earlier, this transformation updates the available files containing data regarding the existing products. Below is a description of how these files are updated. Looking at diagram 2.1.2, it is decomposed into three main processes.

1. Update-PF-Products-LIST:

The term "PF" in the title represents production facility. If the signal EV-NEW-PRODUCT [D-2.1, 125] is received, then this process accesses the NEW-PRODUCTS-ORDERED [D-2.1, 125] data store to obtain the information regarding the new products. This information is used to update the DPMF-PRODUCTS-FILE [D-2.1, 125]. This data store contains a description of each product.

2. Determine-Engr.-Specs.:

Whenever the signal EV-NEW-PRODUCT [D-2.1, 125] is received, then this process accesses the NEW-PRODUCTS-ORDERED [D-2.1, 125] data store to obtain the information about the new products. It then determines the engineering
UPDATE-DPMF-FILES (2.1.2).
specifications of these products. The bill of materials
determined is stored in BILL-OF-MATERIALS [D-2.1.2, 129]
data store. The engineering specifications (ENGR.-SPECS.)
are also sent to DETERMINE-MFG.-PROCESSES.

3. Determine-Mfg.-Processes:

Once the engineering specifications (ENGR.-SPECS.) have
been determined, this process uses this information to
determine what manufacturing processes are needed to
manufacture the new products. Along with that, it
determines the processing times and jobs routings. It then
updates the data stores JOBS-PROCESSING-TIMES and JOBS-
ROUTINGS. Finally, the PRODUCTS-STANDARDS [D-0, 98] data
store is updated with the standards of the new products.

D-2.1.3 (SHIPMENTS-INV.-CONTROL)

As mentioned earlier, this transformation monitors the
inventory of products, issues orders, and generates the
shipments. Decomposing this transformation further, as
shown on diagram 2.1.3, it consists of five main processes.

1. Check-Avail.-of-Products:

Upon receiving the list PRODUCTS-ORDERED [D-2.1, 125],
this transformation accesses the DPMF-PRODUCTS-INVENTORY
[D-2, 121] data store to check the availability of
products. If products are available, the massage
EV-PRODUCTS-AVAIL is sent to RETRIEVE-AND-SHIP. Otherwise, the signal EV-PRODUCTS-NOT-AVAIL is sent to PROCESS-AS-BACK-ORDERS.

2. Retrieve-Ship-Products:

This process retrieves the products from DPMF-PRODUCTS-INVENTORY \([D-2, 121]\) and ships them to the warehouses (PRODUCTS-SHIPMENTS). This is done whenever EV-PRODUCTS-AVAIL. (which indicates that the products are available in inventory) or EV-PRODUCTION-COMPLETED \([D-2, 121]\) (which indicates that production has been completed for some earlier order) occur. Once the shipments are made, the signal EV-SHIPMENTS is sent to PRODUCTS-INV.-CONTROL, which indicates that shipments for products have been made.

3. Update-Orders-File:

This process updates the status of the orders for which shipments have been made. It accesses DPMF-OUTSTANDING-PRODUCTS-shipments \([D-2.1, 125]\) data store and determines which of the outstanding orders corresponds to the shipment made. Once that order is found, the status of that order is changed from outstanding to satisfied.

4. Process-as-Back-Orders:

If the products ordered are not available, this transformation receives the message EV-PRODUCTS-NOT-AVAIL.
Upon receiving that message, it tags the corresponding order as a back-order. The list of these BACK-ORDERS is then sent to PRODUCTS-INV.-CONTROL. EV-B.O. signals PRODUCTS-INV.-CONTROL that back orders have occurred and to take the necessary action, which will described in detail in section D-2.1.3.5 below.

5. Products-Inv.-Control:

This transformation controls the inventory of the products by applying the policy selected by the user. EV-TIME [CONTEXT, 89] indicates to this transformation the current time, which is needed in the case were the periodic policy is used. While PRODUCTS-SHIPMENTS is needed in the case were the continuous policy is used, and BACK-ORDERS is needed regardless of the type of policy used. EV-B.O signals PRODUCTS-INV.-CONTROL that some orders for products have been tagged as back-orders. If there are orders for new products, this process accesses the NEW-PRODUCTS-ORDERED [D-2.1, 125] data store to retrieve the needed information to generate PRODUCTION-ORDERS [D-2, 121] for these products. PRODUCTION-ORDERS are also generated for products with insufficient inventory levels and for any back-orders.

D-2.1.3.5 (PRODUCTS-INV.-CONTROL)

This diagrams shows a more detailed picture of the
PRODUCTS-INV.-CONTROL (2.1.3.5).
transformation PRODUCTS-INV.-CONTROL discussed in section D-2.1.3 above. As mentioned, it control the inventory of products at the production facility. Looking at diagram 2.1.3.5, this transformation is decomposed into four main processes and one control process.

1. Identify-PF-Policy:

With the occurrence of EV-SHIPMENTS [D-2.1.3, 131] and/or EV-B.O [D-2.1.3], this component accesses the DPMF-INVENTORY-POLICIES [D-0, 98] data store to identify which of the policies is to be used for controlling the products at the production facility. Once that information is obtained, it is sent to the CONTROL process. If the policy to be used is the On-Order (On-Demand) policy (which basically assumes zero inventory), then the signal EV-ON-DEMAND is sent to the control process, which in turn enables the process APPLY-PF-ON-ORDER-POLICY. Otherwise, the signal EV-OTHER is sent to the CONTROL process, which in turn enables the process APPLY-PF-INVENTORY-POLICIES.

2. Apply-PF-On-Order-Policy:

If this transformation is enabled by the CONTROL process, it generates a list of the NEEDED-PRODUCTS. The information used to generate this list comes from the data BACK-ORDERS [D-2.1.3, 131]. It also generates this list whenever new products have been ordered. In order to
obtain the necessary information, it accesses the data store NEW-PRODUCTS-ORDERED [D-2.1, 125].

3. Apply-PF-Inventory-Policies:

This process applies the policy to be used for controlling the products' inventory levels. When this process is enabled by EV-ENAB.-DISAB., it accesses the DPMF-PRODUCTS-INVENTORY [D-2, 121] data store and decides whether to order products or not. PRODUCTS-SHIPMENTS are also used to check the inventory levels if the policy is such inventory is checked every time a shipment occurs. Details regarding the inventory policies used and what inputs are needed for each is given in section D-2.1.3.5.3. If it is necessary to order products, then a list of the NEEDED-PRODUCTS is generated. EV-TIME is used to indicate the current time and date for use when implementing the inventory policies, which will also be detailed in section D-2.1.3.5.3 below.

4. Process-Production-Orders:

Once this transformation receives the list of NEEDED-PRODUCTS from either process 2 or 3, it prepares the orders, files them in the DPMF-OUTSTANDING-ORDERS [D-2.1, 125] data store, and sends the PRODUCTION-ORDERS [D-2, 121] to the PRODUCTION-PLAN.-CONTROL transformation. DPMF-OUTSTANDING-ORDERS contains the information regarding
orders that have received for products but not yet satisfied.

D-2.1.3.5.3 (APPLY-PF-INVENTORY-POLICIES)

This diagram presents the decomposition of the transformation APPLY-PF-INVENTORY-POLICY shown in diagram 2.1.3.5. This transformation is decomposed into three main operations, namely, IDENTIFY-PF-INV.-POLICY, APPLY-PF-PERIODIC-POLICY (where PF means warehouses), and APPLY-PF-CONTINUOUS-POLICY, and a CONTROL PROCESS.

1. Identify-PF-Inv.-Policy:

Once this process is enabled by EV-ENAB.-DISAB [D-2.1.3.5, 136], it accesses the data store DPMF-INVENTORY-POLICIES [D-0, 98] to identify which of the specific inventory policies to use for controlling the inventory of products. If the policy to be used is the periodic policy, then the signal EV-PERIODIC is sent to the CONTROL process. The CONTROL process, in turn, enables APPLY-PF-PERIODIC-POLICY. Otherwise, the signal EV-CONTINUOUS is sent to the CONTROL process. The control process, in turn, enables APPLY-PF-CONTINUOUS-POLICY.

2. Apply-PF-Periodic-Policy:

As this transformation is enabled by the CONTROL process, it applies the periodic policy to control the
APPLY-PF-INVENTORY-POLICY (2.1.3.5.3).
products’ inventory. At the end of each review period, which is determined by \text{EV-TIME} \text{[CONTEXT, 89]}, the data store DPMF-PRODUCTS-INV\text{ENTORY} \text{[D-2, 121]} is accessed. If any of the products have insufficient inventory levels, a list is generated for those products contains the type of products and their respective quantities. The list of NEEDED-PRODUCTS \text{[D-2.1.3.5, 136]} is then sent to the PRODUCTION-PLAN.-CONTROL transformation.

3. Apply-PF-Continuous-Policy:

If this transformation is enabled by the CONTROL process, it applies the continuous policy to control the inventory of products. On a continuous basis and/or when PRODUCTS-SHIPMENTS occurs, the data store DPMF-PRODUCTS-INV\text{ENTORY} \text{[D-2, 121]} is accessed to check for insufficient inventory levels. If any are detected, it generates a list of the NEEDED-PRODUCTS \text{[D-2.1.3.5, 136]} which contains the type of products needed and their respective quantities.

D-2.2 (PRODUCTION-PLAN.-CONTROL)

This diagram decomposes PRODUCTION-PLAN.-CONTROL further to show a detailed picture of the operations involved. As mentioned previously, this transformation represents the operations of planning and controlling the production. Looking at diagram 2.2, it consists of four main operations.
1. Identify-Mtl.-Needed:

This process identifies the raw materials needed to manufacture the products. As it receives the list of products to be manufactured (PRODUCTION-ORDERS [D-2, 121]), it accesses the data store BILL-OF-MATERIALS [D-2.1.2, 129] to identify the raw materials needed. Once this is done, the list of needed materials (MTL-NEEDED) is sent to the process CHECK-AVAIL.-OF-MTL.

2. Check-Avail.-of-Mtl:

The word "Avail." in the title refers to Available. As this process receives the list of MTL-NEEDED, it accesses the data store RAW-MATERIALS-INVENTORY [D-2, 121] to check if the raw materials needed are available in inventory. If they are, the massage EV-MTL-AVAILABLE is sent to the process PRODUCTION-OPERATIONS. Otherwise, the massage EV-MTL-NOT-AVAILABLE [D-2, 121] is sent to PROCESS-FILE-AS-BACK-ORDERS and RAW-MATERIALS-CONTROL.

3. Process-File-Back-Orders:

If the massage EV-MTL-NOT-AVAILABLE [D-2, 121] is received, this process files the corresponding orders as a back-order and stores that information in the data store DPMF-OUTSTANDING-ORDERS [D-2.1, 125]. Else, if the massage EV-MTL-ARRIVED [D-2, 121] is received, it retrieves the back-orders from the data store DPMF-OUTSTANDING-ORDERS and
accesses BILL-OF-MATERIALS [D-2.1.2, 129] to identify the corresponding materials needed. Finally, it sends the orders for which raw materials are now available (SELECTED-ORDERS) to PRODUCTION-OPERATIONS.

4. Production-Operations:

This transformation represents the operations involved in grouping and sequencing the jobs, and synchronizing the production flow in the facility. If the message EV-MTL-AVAILABLE, which correspond to PRODUCTION-ORDERS [D-2, 121], is received, this transformation accesses the data stores RAW-MATERIALS-INVENTORY [D-2, 121], PRODUCTS-STANDARDS [D-0, 98], and SCHEDULING-SEQUENCING-POLICIES [D-0, 98] for the information needed to perform the production operations. Once production is completed, the finished products data is stored in DPMF-PRODUCTS-INVENTORY [D-2, 121] and the message EV-PRODUCTION-COMPLETED [D-2, 121] is sent to RAW-MATERIALS-CONTROL. The PRODUCTION-REPORTS [D-2, 121] are generated in response to REPORTS-REQUEST [CONTEXT, 89].

D-2.2.4 (PRODUCTION-OPERATIONS)

This diagram shows a more detailed picture of PRODUCTION-OPERATIONS. As mentioned in section D-2.2 above, this transformation represents the operations involved in grouping and sequencing the jobs, and
1. Group-and-Assign-Job-Numbers
   - production-orders
   - selected-orders

2. Sequence-Jobs
   - grouped-jobs
   - jobs-sequence

3. Synchronize-Production
   - Grouped-Sequence-Jobs
   - Jobs-Routings
   - Jobs-Processing-Times
   - EV-production-completed
   - EV-time
   - Raw-Materials-Inventory
   - DPMF-Products-Inventory

PRODUCT-OPERATIONS (2.2.4)
synchronizing the production flow in the facility. Looking at diagram 2.2.4, it decomposed into three main processes.

1. **Group-and-Assign-Job-Numbers:**

   This process groups the jobs for production and assigns a number to each group, hence each job carries its group number. Once the PRODUCTION-ORDERS [D-2, 121] are received, this transformation groups the orders in jobs. If SELECTED-ORDERS [D-2.2, 140] are received, which correspond to earlier back-orders, these orders are also grouped with the production orders. To help determine which jobs (orders) to group, this transformation accesses the data store PRODUCTS-STANDARDS [D-0, 98] to obtain the necessary information. This information would include the specifications of each product and the operations required for manufacturing the product. The GROUPED-JOBS are then sent to the process SEQUENCE-JOBS. When the jobs have been sequenced, this transformation stores the jobs that have been grouped and sequenced in GROUPED-SEQUENCED-JOBS [D-2.2.4, 143].

2. **Sequence-Jobs:**

   Once the jobs have been grouped, this transformation accesses the data stores SCHEDULING-SEQUENCING-POLICIES [D-0, 98], JOBS-ROUTINGS [D-2.1.2, 129], and JOBS-PROCESSING-TIMES [D-2.1.2, 129] to obtain the necessary
information for sequencing the jobs on the machine centers. The SEQUENCED-JOBS are then sent to GROUP-AND-ASSIGN-JOB- NUMBERS.

3. Synchronize-Production:

This transformation represents the operations necessary for synchronizing the flow of production through the production facility. Once the message EV-JOBS-GROUPED [D-2.2.4, 143] is received, this process accesses the data store GROUPED-SEQUENCED-JOBS [D-2.2.4, 143] to perform the necessary operations. These operations involve identifying the bottleneck machines, scheduling the production on the bottlenecks, and controlling the production on the machines. The necessary information needed for performing the above task, this transformation accesses the data stores JOBS-ROUTINGS and JOBS-PROCESSING-TIMES [D-2.1.2, 129], and RAW-MATERIALS-INVENTORY [D-2, 121]. This will be detailed further in section D-2.2.4.3 below.

Once the production is completed, the finished products data is stored in DPMF-PRODUCTS-INVENTORY [D-2, 121], and the message EV-PRODUCTION-COMPLETED [D-2, 121] is generated and sent to the respective transformation(s). The event flow EV-TIME [CONTEXT, 89] is used as a timing clock for scheduling the jobs on the bottlenecks and controlling the release of jobs for production.
D-2.2.4.3 (SYNCHRONIZE-PRODUCTION)

The description that follows gives a more detailed picture of SYNCHRONIZE-PRODUCTION shown on diagram 2.2.4. As mentioned, this transformation represents the operations necessary for synchronizing the flow of production. Looking at diagram 2.2.4.3, it is decomposed into five main processes.

1. Identify-BN-Operations:

The word "BN" in the title represents bottleneck. This process identifies which of the machine centers, on which the jobs will be produced, are the bottleneck operations. Once it receives the message EV-JOBS-GROUPED [D-2.2.4,143], it accesses the data store GROUPED-SEQUENCED-JOBS [D-2.2.4, 143] and retrieves the information regarding those jobs. After it identifies the bottleneck operations, it stores this data in the BOTTLENECK-OPERATIONS-FILE and sends the message EV-BN-IDENTIFIED triggers SCHEDULE-BOTTLENECK to start.

2. Schedule-Bottleneck:

This process first accesses GROUPED-SEQUENCED-JOBS [D-2.2.4, 143] to determine which products are to be scheduled for production. Then, it classifies the sequencing problem depending on whether (1) the jobs have similar processing times; the data regarding the processing
SYNCHRONIZE-PRODUCTION (2.2.4.3).
times of each product is obtained from JOB-PROCESSING-TIMES [D-2.1.2, 129], (2) the bottleneck operation makes more than one part; it accesses JOBS-ROUTINGS [D-2.1.2, 129] and BOTTLENECK-OPERATION-FILE to obtain that information, (3) the bottleneck involves setup; it obtains this information from BOTTLENECK-OPERATION-FILE, (4) one bottleneck feeds another; it accesses JOBS-ROUTINGS and BOTTLENECK-OPERATION-FILE to obtain the necessary information, or (5) the jobs have dramatically different setup times; it obtains needed information from JOBS-PROCESSING-TIMES.

Then, the bottleneck is scheduled using specific methods according to its classification (which is detailed in section D-2.2.4.3.2). Once the schedule of the bottleneck is generated (BN-SCHEDULE), it is sent to START-PRE-BN-OPERATIONS. The information stored in JOBS-ROUTINGS is used to determine the optimum schedule for manufacturing the current jobs (orders).

3. Start-Pre-BN-Operations:

This process starts pre-bottleneck operations once the BN-SCHEDULE is received. The event flow, EV-TIME [CONTEXT, 89], entering the process represents the actual time (hour, day, week, etc.), which is used for controlling the release of jobs for production. It accesses GROUPED-SEQUENCED-JOBS [D-2.2.4, 143] to obtain the necessary information regarding the jobs to be produced. Once the operations
have been completed on the machine centers preceding the bottleneck, the finished jobs are stored in JOBS-QUEUE-I and the signal EV-PRE-BN-COMPLETED is sent to START-BN-OPERATION.

4. Start-BN-Operation:

The fourth process, START-BN-PRODUCTION, starts the production on the bottleneck as soon as it gets the signal EV-PRE-BN-COMPLETED. This transformation accesses the data store JOBS-QUEUE-I, retrieves the jobs data, and starts the production. JOBS-QUEUE-I is a stock buffer which needs to be placed in front of the bottleneck operation to prevent it from being starved due to disruptions in one of the operations that feed it. Once the production is done, it stores the finished jobs data in JOBS-QUEUE-II and the signal EV-BN-COMPLETED is sent to START-POST-BN-OPERATIONS.

5. Start-Post-BN-Operations:

This process starts the production on the machines following the bottleneck once the signal EV-BN-COMPLETED is received. Once the production is completed on these machines, the finished jobs are stored in DPMF-PRODUCTS-INVENTORY [D-2, 121] and the signal EV-PRODUCTION-COMPLETED [D-2, 121] is generated.
D-2.2.4.3.1 (IDENTIFY-BN-OPERATIONS)

This diagram decomposes IDENTIFY-BN-OPERATIONS to show its components. This transformation identifies which of the machines are the bottlenecks. As can be seen on diagram 2.2.4.3.1, it is decomposed into three main processes.

1. Initial-LP-Formulation:

   In the title, LP represents linear programming. This process formulates the initial linear programming model that represents the scheduling problem, and from which the bottlenecks are to be identified. It accesses the data store GROUPED-SEQUENCED-JOBS [D-2.2.4, 143] to obtain the information about the jobs to be produced for that period. The resulting LP model is then stored in MASTER-LP-FILE, and the massage EV-LP-FORMULATED is sent to SOLVE-LP.

2. Solve-LP:

   This process solves the linear programming model generated. Once it receives the massage EV-LP-FORMULATED, it accesses the data store MASTER-LP-FILE to obtain the information about the LP model. After obtaining the solution to the LP model, it is stored in the MASTER-LP-SOLUTION-FILE [D-2.2.4.3.1, 151] and the massage EV-LP-SOLVED [D-2.2.4.3.1, 151] is sent to DETERMINE-BN.
IDENTIFY-BN. OPERATIONS (2.2.4.3.1)
3. Determine-BN:

This process determines which of the operations are the bottlenecks. Once the massage EV-LP-SOLVED [D-2.2.4.3.1, 151] is received, it accesses the data store MASTER-LP-SOLUTION-FILE [D-2.2.4.3.1, 151] and obtains the solution tableau. It determines the bottleneck operations from this tableau. The information regarding the bottleneck operations is then stored in BOTTLENECK-OPERATION-FILE [D-2.2.4.3, 147] and the massage EV-BN-IDENTIFIED [D-2.2.4.3, 147] is sent to the respective transformation(s).

D-2.2.4.3.1.1 (INITIAL-LP-FORMULATION)

This diagram gives a more detailed picture of INITIAL-LP-FORMULATION shown in diagram 2.2.4.3.1. As mentioned above, this transformation formulates the linear programming model. As shown on diagram 2.2.4.3.1.1, it is decomposed further into five main processes.

1. Modify-Files:

This process makes the necessary modifications, issued by the user, to the respective files. Once the DESIRED-CHANGES [CONTEXT, 89] are received, the data stores REVENUES-EXPENSES-FILE, BILL-OF-MATERIALS [D-2.1.2, 129], GROUPED-SEQUENCED-JOBS [D-2.2.4, 143], and OPERATIONS-DATA-FILE are modified. It is not necessarily that all
1 Modifying Files
2 Formulate Objective Function
3 Formulate RHS-Vector
4 Formulate Tech-Matrix
5 Synthesize LP
these files will be modified, only those files corresponding to the desired changes are modified.

2. Formulate-Objective-Function:

Once the massage EV-JOBS-GROUPED [D-2.2.4, 143] is received, this process accesses the data store REVENUES-EXPENSES-FILE to formulate the objective function. The objective could be to minimize or maximize the function. After formulating the objective, it is (FORMULATED-OBJECTIVE) sent to SYNTHESIZE-LP.

3. Formulate-RHS-Vector:

Once the massage EV-JOBS-GROUPED [D-2.2.4, 143] is received, this process accesses the data stores REVENUES-EXPENSES-FILE, BILL-OF-MATERIALS [D-2.1.2, 129], GROUPED-SEQUENCED-JOBS [D-2.2.4, 143], and OPERATIONS-DATA-FILE to formulate the right-hand side vector of the matrix. The FORMULATED-RHS is then sent to SYNTHESIZE-LP.

4. Formulate-Tech.-Matrix:

Once the massage EV-JOBS-GROUPED [D-2.2.4, 143] is received, this process accesses the data stores GROUPED-SEQUENCED-JOBS [D-2.2.4, 143] and OPERATIONS-DATA-FILE to formulate the technology matrix. The FORMULATED-MATRIX is then sent to SYNTHESIZE-LP.
5. **Synthesize-LP:**

Upon the complete formulation of the objective function, the right-hand side, and the technology matrix of the programming model, this process synthesizes them to formulate the final linear programming model. Once the final model is formulated, it is stored in MASTER-LP-FILE [D-2.2.4.3.1, 151] and the signal EV-LP-FORMULATED [D-2.2.4.3.1, 151] is sent to the respective transformation(s).

**D-2.2.4.3.1.2 (SOLVE-LP)**

This diagram shows a more detailed picture of SOLVE-LP. As mentioned in the previous section, this transformation solves the developed linear programming model. As seen on diagram 2.2.4.3.1.2, it is decomposed further into three main processes.

1. **Revise-LP:**

This process revises the linear programming model. This revision is necessary when changes occur from period to period as new orders are received and production is completed. Once the signal EV-LP-FORMULATED [D-2.2.4.3.1, 151] is received, it accesses the data store MASTER-LP-FILE [D-2.2.4.3.1, 151] and revises the LP. Then it sends the REVISED-LP to SOLVE-REVISED-LP.
2. Restore-Old-Solution:

As this process receives the message EV-LP-FORMULATED [D-2.2.4.3.1, 151], it accesses the data store MASTER-LP-SOLUTION-FILE [D-2.2.4.3.1, 151] to obtain the old linear programming solution. This RESTORED-SOLUTION is sent to the SOLVE-REVISED-LP transformation. The old solution is restored to be used as an advance starting point to hopefully save time.

3. Solve-revised-LP:

This process solves the revised linear programming model (REVISED-LP received from REVISE-LP) using the RESTORED-SOLUTION. The solution to the revised model is then sent to the MASTER-LP-SOLUTION-FILE.

D-2.2.4.3.1.3 (DETERMINE-BN)

This diagram shows a more detailed picture of DETERMINE-BN. As mentioned in section 2.2.4.3.1, this transformation determines which of the operations involved are the bottlenecks. As can be seen in diagram 2.2.4.3.1.3, it is decomposed further into four main processes.

1. Classify-Slack-Variables:

Once this process is triggered by EV-LP-SOLVED [D-2.2.4.3.1, 151], it accesses MASTER-LP-SOLUTION-FILE
1. Classify Slack Variables
   - EV-LP-solved

2. Test Variables Value
   - vars-greater-than-zero

3. Classify as a BN
   - non-basic-variables
   - vars-equal-zero

4. Classify as a Non-BN

Master-LP-Solution-File

Non-BN-Operations-File

Bottleneck-Operation-File

DETERMINE-BN (2.2.4.3.1.3.)
[D-2.2.4.3.1, 151] to determine whether the slack variables are basic or nonbasic variables (note that a slack variable is assigned to each machine). If they are basic, then they are sent to TEST-VARIABLES-VALUE, otherwise they are sent to CALASSIFY-AS-A-BN.

2. Test-Variables-Value:

This process tests the values of the BASIC-VARIABLES received. If their value is greater than zero, then CLASSIFY-AS-A-NON-BN is notified. If their value is equal to zero, then CLASSIFY-AS-A-BN is notified.

3. Classify-as-a-BN:

If a variable has been classified as a nonbasic variable or a basic variable with a value greater than zero, then this process classifies the corresponding machine as a bottleneck operation. This data is then stored in BOTTLENECK-OPERATION-FILE.

4. Classify-as-a-Non-BN:

If a variable has been classified as a basic variable with a value greater than zero, then this process classifies the corresponding machine as a non-bottleneck operation. This information is then stored in NON-BN-OPERATION-FILE. The user can access this file whenever needed to obtain this information.
D-2.2.4.3.2 (SCHEDULE-BOTTLENECK)

This diagram decomposes SCHEDULE-BOTTLENECK further to show more detailed picture of it. As mentioned in section 2.2.4.3, this transformation schedules the production on the bottleneck operations using specific rules which depend on the type of scheduling problem involved. Looking at diagram 2.2.4.3.2, it is decomposed into six main processes.

1. Classify-Ordering-Problem:

   As this process receives the message EV-BN-IDENTIFIED [D-2.2.4.3, 147], it accesses the data stores BOTTLENECK-OPERATION-FILE, JOBS-ROUTINGS and JOBS-PROCESSING-TIMES [D-2.1.2, 129], and GROUPED-SEQUENCED-JOBS [D-2.2.4, 143] to determine what type of problem is involved. As can be seen on the diagram, five cases have been identified. The narrative description of the five cases is combined in one section below.

   (1) Sched-By-Due-Date: If the jobs have been recognized to have similar processing times (PTs-ARE-SIMILAR), then this process schedules the jobs for that period according to their due date.

   (2) Sched-By-Min-1-M/C-Make-Span: The "M/C" in the title represents machine. If it has been recognized that a bottleneck makes more than one part of a product (BN-MAKES-MORE-THAN-1-PART), then this process treats
SCHEDULE-BOTTLENECK (2.2.4.3.2).
the scheduling problem as one of sequencing products on a single machine to minimize the make span, i.e. minimize the time it takes to manufacture the product on the machines from begining to end.

(3) Solve-Travel-Sales-Prob.: The words "Travel" and "Prob." in the title represent "Travelling" and "Problem" respectively. If it has been identified that the bottleneck operation involves setup time (BN-INVOLVES-SETUP), then this process schedules the jobs using a travelling salesmen algorithm.

(4) Sched-By-Min-2-M/C-Make-Span: If it has been recognized that one bottleneck feeds another (ONE-BN-FEED-ANOTHER), then this transformation schedules the jobs through the two machines to minimize their make span.

(5) Sched-By-Remaining-PT: PT in the title represents processing time. If it has been recognized that the jobs' processing times are very different from one another (PTs-ARE-VERY-DIFFERENT), then this process schedules the jobs by their remaining processing time. Once the scheduling problem has been identified and solved as mentioned above, the bottleneck schedule (BN-SCHEDULE [D-2.2.4.3, 147]) is generated by the appropriate process and sent to the respective transformation(s).
D-2.2.4.3.3 (START-PRE-BN-OPERATIONS)

This diagram shows a more detailed picture of the START-PRE-BN-OPERATION depicted in diagram 2.2.4.3. As mentioned in section D-2.2.4.3, this transformation represents the operations involved in producing the jobs on the machines preceding the bottleneck. Looking at diagram 2.2.4.3.3, it is decomposed into three main processes and one control process.

1. Determine-T-to-Release-Jobs: (T = Release Time)

As this transformation receives the BN-SCHEDULE [D-2.2.4.3, 147], it accesses the data stores JOBS-PROCESSING-TIMES and JOBS-ROUTINGS [D-2.1.2, 129], and GROUPED-SEQUENCED-JOBS [D-2.2.4, 143] to determine the T to release jobs for production.

\[ T = \text{Date and time of when the job has to be done} - \text{total processing time on the post-BN operations} - \text{processing time on the BN} \]

The determined T is then sent to COMPARE-T-WITH-TIME.

2. Compare-T-With-Time:

This process compares the value of T and time indicated by EV-TIME [CONTEXT, 89] to determine if they are equal. If they are, the signal EV-TIMES-ARE-EQUAL is sent to the
Processing Times

1. Determine T-to-Release Jobs
   - BN-schedule
   - Jobs-Routings

2. Compare T-with Time
   - EV-time
   - EV-times are equal

3. Release Jobs-for Production
   - Grouped-sequenced Jobs
   - Raw Materials Inventory

C CONTROL 2.2.4.3.3

EV ENAB DISAB

EV-pre-BN-completed

START-PRE-BN-OPERATIONS (2.2.4.3.3).
CONTROL process.

3. Release-Jobs-For-Production:

Once the CONTROL process receives the massage EV-TIMES-ARE-EQUAL, it enables this transformation to release the jobs for production. To release the jobs, RELEASE-JOBS-FOR-PRODUCTION accesses the data stores RAW-MATERIALS-INVENTORY [D-2, 121] and GROUPED-SEQUENCED-JOBS [D-2.2.4, 143] to obtain the necessary data. As the jobs are produced on the pre-bottleneck machines, the signal EV-PRE-BN-COMPLETED [D-2.2.4.3, 147] is sent to the respective transformation(s) and the jobs’ data is stored in JOBS-QUEUE-I [D-2.2.4.3, 147]. This queue is a list of the jobs waiting to be produced on the bottleneck operation.

D-2.2.4.3.4 (START-BN-OPERATION)

This diagram shows a more detailed picture of the transformation START-BN-OPERATION shown in diagram 2.2.4.3. As mentioned in section D-2.2.4.3, this process represents the production of jobs on the bottleneck machines. Looking at diagram 2.2.4.3.4, it is decomposed further into two main processes.

1. Release-Jobs-From-I:

Once this process receives the massage EV-PRE-BN-COMPLETED [D-2.2.4.3, 147], it accesses the data store JOBS-
1. Release-Jobs-from-I

2. Process-BN-Operation

EV-pre-BN-completed

released-jobs

Jobs-Queue-I

EV-BN-completed

Jobs-Queue-II

START-BN-OPERATION (2.2.4.3.4)
QUEUE-I [D-2.2.4.3, 147] and releases the jobs for production. It then sends the RELEASED-JOBS to PROCESS-BN-OPERATION.

2. Process-BN-Operation:

As the RELEASED-JOBS are received, this transformation processes the jobs on the bottleneck operation. Once the jobs are finished, their data is stored in JOBS-QUEUE-II [D-2.2.4.3, 147] and the massage EV-BN-COMPLETED [D-2.2.4.3, 147] is sent to the respective transformation(s).

D-2.2.4.3.5 (START-POST-BN-OPERATIONS)

This diagram shows a more detailed picture of the transformation START-POST-BN-OPERATIONS depicted in diagram 2.2.4.3. As mentioned in section D-2.2.4.3, this transformation represents processing the jobs on the machine following the bottleneck. Looking at diagram 2.2.4.3.5, it can be seen that it is decomposed into two main processes.

1. Release-Jobs-From-II:

Once this process receives the massage EV-BN-COMPLETED [D-2.2.4.3, 147], it accesses the data store JOBS-QUEUE-II [D-2.2.4.3, 147] and releases the jobs for production. It then sends the RELEASED-JOBS to PROCESS-POST-BN-OPERATIONS.
2. Process-Post-BN-Operations:

As the RELEASED-JOBS are received, this transformation processes the jobs on the machines following the bottleneck. Once the jobs are finished, they are stored in DPMF-PRODUCTS-INVENTORY [D-2, 121] and the message EV-PRODUCTION-COMPLETED [D-2, 121] is sent to the respective transformation(s). Refer to the data dictionary for such details.

D-2.3 (RAW-MATERIALS-CONTROL)

This diagram shows a more detailed picture of the transformation RAW-MATERIALS-CONTROL depicted in diagram 2. As mentioned in section D-2, this transformation represents the operations involved in controlling the raw materials at the production facility. Looking at diagram 2.3, it can be seen that this transformation is decomposed further into six main operations.

1. Update-Inv.-Orders-Status:

Whenever MATERIALS [CONTEXT, 89] are received from the suppliers, this process (1) updates the RAW-MATERIALS-INVENTORY [D-2, 121] data store, (2) updates the status of the corresponding materials orders in the data store MATERIALS-ORDERS-STATUS, and (3) sends the RECEIVED-MATERIALS to GENERATE-MATERIALS-REPORTS, and (4) sends the signal EV-MTL-ARRIVED [D-2, 121] to the PRODUCTION-PLAN.
1. Update Inventory Status

2. Generate Materials Reports

3. Identify Raw Material Policy

4. Apply Inventory Policy

5. Apply on Demand Policy

6. Process Materials Order

---

RAW-MATERIALS-CONTROL (2.3)
CONTROL transformation.

2. Generate-Materials-Reports:

   This process generates reports regarding raw materials inventory and orders status (MATERIALS-REPORTS [D-2, 121]) whenever REPORT-REQUEST [CONTEXT, 89] is received from the user. To obtain the necessary information, it accesses the data stores RAW-MATERIALS-INVENTORY [D-2, 121] and MATERIALS-ORDERS-STATUS.

3. Identify-Raw-Mtl-Policy:

   With the occurrence of EV-PRODUCTION-COMPLETED [D-2, 121] and/or EV-MTL-NOT-AVAILABLE [D-2, 121], this component accesses the MATERIALS-INVENTORY-POLICIES data store to identify which of the policies is to be used for controlling the inventory of raw materials at the production facility. If the policy to be used is the On-Demand policy (which basically assumes zero inventory), then the signal EV-ON-DEMAND is sent to APPLY-ON-DEMAND-POLICY. Otherwise, the signal EV-OTHER is sent to APPLY-INVENTORY-POLICY.

4. Apply-Inventory-Policy:

   This process applies the policy to be used for controlling the raw materials' inventory levels. It accesses the RAW-MATERIALS-INVENTORY [D-2, 121] data store
and decides whether to order materials or not. If it is necessary to order materials, a list of the MATERIALS-NEEDED is generated. The decomposition of this transformation would be exactly the same as the one shown in diagram 2.1.3.5.3 with very minor changes in the names of the data flows.

5. Apply-On-Demand-Policy:

If this transformation is triggered by IDENTIFY-RAW-MTL-POLICY, it generates a list of the MATERIALS-NEEDED and sends it to PROCESS-MATERIALS-ORDERS.

6. Process-Materials-Orders:

Once this process receives the list of MATERIALS-NEEDED from either process 4 or 5, it prepares the orders and sends the MATERIALS-ORDER [CONTEXT, 89] to the raw materials suppliers.

D-2.3.1 (UPDATE-INV-ORDERS-STATUS)

This diagram shows a more detailed picture of UPDATE-INV-ORDERS-STATUS. As mentioned above, this transformation updates the status of inventory and orders. Looking at diagram 2.3.1, it can be seen that it is decomposed into two main transformations.
1. Update-Mtl-Order-Status:

Once this process receives RECEIVED-MATERIALS, it updates the status of the corresponding order in the MATERIALS-ORDERS-STATUS [D-2.3, 170] data store.

2. Update-Raw-Mtl-inventory:

Upon receipt of the raw MATERIALS [CONTEXT, 89] that correspond to an earlier order, this process updates the levels of inventory (RAW-MATERIALS-INVENTORY [D-2, 121]) with the received quantity and sends the signal EV-MTL-ARRIVED [D-2, 121] to the respective transformation(s).
CHAPTER V

CASE STUDY: PROBLEM DEFINITION

1. BACKGROUND

The Ohio Department of Transportation (ODOT) maintains a traffic sign production facility (Sign Shop) in Columbus, Ohio. The facility produces all the traffic signs installed on Ohio's rural roadway system which consists of U.S., State, and Interstate highways. As shown in Figure 5.1, there are twelve districts within Ohio which are responsible for installing and maintaining the traffic signs on the roads. These districts place orders for traffic signs with the Sign Shop. Each districts has its own warehouse (districts warehouses) where the traffic signs are stored.

As mentioned in chapter II, the flexible simulation model (FSM) consists of three different types of decision support models: (1) forecasting, (2) production scheduling and job sequencing, and (3) inventory models. For each type of models, there are a number of specific model alternatives from which the system designer can choose.
Figure 5.1. A Map of the Twelve Districts and the Location of the Sign Shop with Respect to Each District.
The goal is to select the particular combination of models which will enable the production and distribution system to be operated in an optimal fashion.

As will be explained in section 5, an investigation conducted by Yamada (1987) concluded that the demand for signs was found to be unforecastable, hence, forecasting could not be included in this case study as originally intended. Therefore, the FSM will be used to determine the optimal mix of production scheduling and inventory control models only.

2. DATA COLLECTION

Visits to the Sign Shop were conducted to become familiar with the processes required for manufacturing the different types of signs, and determine the specifications for the various signs. A video camera was used to record this information. The video tapes were then analyzed to obtain estimates of the times required for: (1) machine setup, (2) production, and (3) material handling.

Documented files of previous sign orders were also acquired. These files were used for the purpose of forecasting the demand for signs. They were also used to estimate the percentage of the Sign Shop capacity which was
3. SIGN SHOP (Production Facility)

As mentioned earlier, the Sign Shop manufactures traffic signs in response to orders received by the districts. At the time of this project, the Sign Shop did not utilize any models for forecasting the demand nor controlling inventories and scheduling production of signs in the shop. Basically, the signs are produced to order on a first-come-first-serve (FCFS) basis. Hence, no signs are maintained in inventory.

The following sections provide a description of the different types of signs manufactured at the Sign Shop. A description of the processes required for manufacturing the traffic signs and signs' structure is also provided.

3.1 Types of Traffic Signs

The Ohio Uniform Traffic Control Signs Chart groups the signs in terms of their function as shown in Table 5.1. Examples of these signs are shown in Figure 5.2.

3.2 Types of Production Operations

There are in total eleven different operations that are involved in manufacturing the different types of signs.
<table>
<thead>
<tr>
<th>Sign Function</th>
<th>Base Material of which sign is made</th>
<th>Color of Background</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warning</td>
<td>Aluminum</td>
<td>Yellow</td>
</tr>
<tr>
<td>Regulatory</td>
<td>Aluminum</td>
<td>White, Yellow, or Red</td>
</tr>
<tr>
<td>Construction &amp; Maintenance</td>
<td>Aluminum</td>
<td>Orange or White</td>
</tr>
<tr>
<td>Route Marker</td>
<td>Aluminum or Extruded Aluminum</td>
<td>White, Green, or Blue</td>
</tr>
<tr>
<td>Rest Area</td>
<td>Aluminum or Extruded Aluminum</td>
<td>Blue</td>
</tr>
<tr>
<td>Freeway &amp; Expressway</td>
<td>Extruded Aluminum</td>
<td>Green, Blue or Brown</td>
</tr>
<tr>
<td>Information</td>
<td>Aluminum or Plywood</td>
<td>White, Green or Blue</td>
</tr>
<tr>
<td>Destination &amp; Distance</td>
<td>Plywood</td>
<td>Yellow, Green, or Brown</td>
</tr>
</tbody>
</table>
Figure 5.2 Examples of Signs for each Category.
Figure 5.2 (Continued).
Figure 5.2 (Continued).
These operations are:

(1) **Shearing.** This operation involves cutting sheets of aluminum (raw material) down to a specific size.

(2) **Sawing.** This process involves sawing extruded aluminum panels and plywood sheets down to a specific size.

(3) **Hole-Punching.** For aluminum signs only, holes are punched in where the sign is to be bolted to the pole on the road.

(4) **Cornering.** This process reshapes the corners of aluminum signs from 90-degree angles to a curved corner.

(5) **Sanding.** After plywood sheets and extruded aluminum panels have been cut to the appropriate shape, sanding is done to ensure smoothness of the surfaces and corners. This is also done to aluminum signs after they have been sheared and hole-punched.

(6) **Sheeting and Trimming.** This process covers the sign with reflective material and trims off the
excess reflective material.

(7) Heat-Treating. This is done to aluminum signs after they have been covered with the self-adhesive reflective sheeting. This process ensures that the reflective material properly adheres to the aluminum surface.

(8) Hand-Lettering. A manual process used for fixing self-adhesive letters to plywood signs after they have been through the reflective sheeting process.

(9) Silk-Screening. This process uses silk screening methods to print the specific symbols and/or letters with the right color of paint.

(10) Button-Down. This process involves the drilling and riveting of button-reflectors symbols and/or letters to extruded aluminum signs.

(11) Drying. A process used for drying the signs after they have been silk-screened.

3.3 Sequence of Operations for the Different Signs

From the facts discussed above, it was recognized that the signs can be divided into three major categories
according the type of base material of the sign: (I) Sheet Aluminum, (II) Extruded Aluminum Pannel, and (III) Plywood.

3.3.1 The process for manufacturing rectangularly shaped aluminum signs consists of the following sequence of operations: (1) shearing, (2) hole-punching, (3) cornering, (4) sanding, (5) sheeting and trimming, (6) heat-treating, (7) silk-screening and, (8) drying. This sequence, which is graphically depicted in Figure 5.3., will be referred to as the main sequence. This sequence will change for other shapes as indicated below.

a) The first change is that an extra operation is required when the shape is triangular or pentagonal. This operation is sawing.

b) The second change is that one of the operations in the main sequence is replaced with another operation when the shape of the sign is octagonal or circular. For example, if the sign is circular, the sequence of operations (the main sequence mentioned above) would include sawing instead of cornering.

Figure 5.4 shows the sequence of operations for shapes other than rectangular,
Figure 5.3 Sequence of Operations for Aluminum Signs Which Have a Rectangular Shape.
Figure 5.4 Sequence of Operations for Aluminum Signs Which Have a Triangular or a Pentagonal Shape.
Figure 5.4 (Cont'd). Sequence of Operations for Aluminum Signs Which Have a Circular or an Octagonal Shape.
3.3.2 The sequence of operations required for manufacturing Extruded Aluminum signs:

Signs made out of extruded aluminum are rectangularly shaped. The sequence of operations for the different colors is shown in Figure 5.5. Note that the process of manufacturing signs that require yellow, brown, or blue as the color for their background involves an additional operation. This operation is applying the proper color of reflective sheeting to the overlay.

3.3.3 The sequence of operations required for manufacturing Plywood signs:

Signs made out of plywood require the same sequence of production operations, as shown in Figure 5.6: (1) sawing wood down to size, (2) drilling the holes, (3) sanding, (4) reflective sheeting, and (5) hand-lettering.

3.4 Structure of Signs

The structure of the signs, usually referred to as the "bill of materials (BOM)", is a list of the items, ingredients, or materials needed to produce the various signs. It lists all of the subassemblies, parts, and raw materials that go into a parent assembly, showing the quantity of each required to make an assembly. It shows how much of each raw materials is needed and also the order
Figure 5.5 Sequence of Operations for Extruded Aluminum Signs.
Figure 5.6 Sequence of Operations for Plywood Signs.
in which the materials must be supplied in order to manufacture a sign.

Due to the large number of different signs, the bill of materials for every sign will not be shown. Instead, a general bill of materials, showing the raw materials that go into manufacturing the different signs belonging to the three major groups (Aluminum, Extruded Aluminum, Plywood) is provided. This bill of materials is graphically depicted in Figure 5.7.

The same applies for extruded aluminum signs. That is, the amount of extruded aluminum panels, the fiber glass layovers, and the size of the sheeting required to manufacture a sign depends on the size of the sign. The requirements for reflective letters and symbols also depends on the information that need to be displayed.

The same also applies for plywood signs. The amount of plywood and reflective sheeting required depends on the size of the sign. In addition, the number of letters and symbols required depends on the information that needs to be displayed.
Figure 5.7 Bill of Materials for the Different Categories of Traffic Signs
4. FORECASTING

An analysis of the forecasting needs of the integrated production planning and inventory control (IPPIC) system was done by Yamada (1987). He investigated the utility of using univariate time series analysis methods to develop models for forecasting the demand of signs and resources which are used in their production. Yamada described the process which was utilized to develop the forecasting model, and the systematic procedure used to determine which member of the ARMA family of models is most appropriate. The ARMA family of models was used since Bowerman and O'Connell (1987) show that it possesses sufficiently flexibility to include the moving average various types of exponential smoothing models as special cases (e.g. single, double, triple, etc and Holt and Winters methods).

Yamada also described the implications of forecasting models for finished sign demands at the district warehouse on (1) the ability to anticipate future demand at the districts and (2) the way the Sign Shop should operate with respect to this anticipation.

Yamada used an F-test developed by Pundit and Wu (198 ) to conclude that only the demands for the different raw
materials were forecastable. Since demand was not necessary to include an evaluation of alternative forecasting models in the case study.

The typical approach in simulation analysis uses randomly generated demand. However, actual historical demand data was used instead of randomly generated demand. This was done for three reasons: (1) To increase the creditability of the model, (2) Sufficient historical demand data was not available to support the type of time series analysis required for discovering the most appropriate stochastic generating models for the demand for the various signs, and (3) Using historical demand data make it possible to use the flexible simulation model (FSM) as a prototype of the actual synchronized manufacturing control system.

5. Grouping of signs

Having recognized the different types of signs and the exact production operations required to manufacture each different type, the signs then could be grouped into "families" in terms of the similarity in their production operations. This grouping would allow, were possible, to group two or more jobs into one job and produce them as one batch. Through this grouping, considerable savings in setup
times could occur. These savings allow for the production of more signs, i.e. increasing productivity, than would be produced without the grouping of the orders. From the facts discussed in section 3 above, it was concluded that the signs should be grouped according to the following major categories (as shown in Figure 5.8):

5.1 Aluminum.

Signs that are made from aluminum are divided into three main subcategories: (1) color, (2) size and (3) shape. Reasons behind this division is explained in the subsections below.

5.1.1 Grouping in terms of color

All through this discussion the color of the reflective sheeting will be referred to as the color of the sign. Three major colors were recognized for the aluminum signs, Group 1. Yellow signs, Group 2. White signs, and Group 3. Orange signs.

Reasons for dividing them into groups depending on the color is because an additional setup is required when changing the production from one color to another at the reflective sheeting process. Since the signs within each of the above groups are of different shapes and sizes, each of these groups is divided further into subgroups in terms
Figure 5.8. Categories According to Which Signs Are Grouped.
of size and shape. Those aluminum signs with a shape other than rectangular are treated separately as a special case because of the slight difference in the sequence of operations as mentioned in section 2.3.

5.1.2 Grouping in terms of shape.

In section 3.3.1 it was seen that the main reason aluminum signs were grouped according to shape is the differences in the sequence of operations, and in the operations themselves. These shape categories are triangular, petagonal, circular, and octagonal.

5.1.3 Grouping in terms of size.

As for the size category, the shape categories mentioned above were sub-divided into different size categories according to the dimensions of the Width (W) or Height (H) as shown in Table 5.2. Reasons behind grouping the signs in those different size categories is because different sizes have different processing times and require different setups on the same machine.

5.2 Extruded Aluminum (Custom made signs).

Extruded aluminum is used to manufacture signs for which the message to be displayed by the signs differ from one location to the other. Extruded Aluminum signs are grouped in terms of the following: (1) the color of the
<table>
<thead>
<tr>
<th>Group #</th>
<th>Dimensions of W &amp; H (inches)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(W &amp; H) &gt; 36</td>
</tr>
<tr>
<td>2</td>
<td>(W &lt;= 36) or (H &lt;= 36) and (W &gt; 30) or (H &gt; 30)</td>
</tr>
<tr>
<td>3</td>
<td>(W &lt;= 30) or (H &lt;= 30) and (W &gt; 24) or (H &gt; 24)</td>
</tr>
<tr>
<td>4</td>
<td>(W &lt;= 24) or (H &lt;= 24) and (W &gt; 20) or (H &gt; 20)</td>
</tr>
<tr>
<td>5</td>
<td>(W &lt;= 20) or (H &lt;= 20) and (W &gt; 12) or (H &gt; 12)</td>
</tr>
<tr>
<td>6</td>
<td>(W &amp; H) &lt;= 12</td>
</tr>
</tbody>
</table>

Table 5.2. Size Categories for Aluminum Signs.
background of the sign and (2) the size of the sign. The signs are not grouped in terms of shape because all extruded aluminum signs are rectangular or square in shape.

5.2.1 Grouping in terms of the color of the background.

These signs are grouped as follows: (1) Group 1. Green background and (2) Group 2. Brown, Yellow, or Blue background. The signs were grouped in terms of the color of the background because, as mentioned in section 3.3, each group requires a different sequence of operations to be manufactured.

5.2.2 Grouping in terms of the size of the sign.

Each of the above groups is then divided into the six different size categories mentioned previously.

5.3 Plywood (Custom made signs).

Plywood is used to manufacture signs for which the message to be displayed by the sign differ from one location to the other. Plywood signs are grouped in terms of the color and size of the sign. Again, since all the signs are rectangular or square in shape, they are not grouped in terms of shape.
5.3.1 Grouping in terms of the color of the sign.

Plywood signs are grouped in terms of their color as follows: (1) Green, (2) Brown, and (3) Yellow.

All the above groups require the same sequence of operations, as mentioned in section 2.3. The only difference from one group to the other is obviously the color of the reflective sheeting required to cover the sign, which adds an additional setup when changing the production from one color to the other (as in the case of the aluminum signs).

5.3.2 Grouping in terms of the size.

Plywood signs are also grouped into the six different size categories mentioned previously in section 5.2.
The original objective of the case study was to investigate how different combinations of production scheduling, inventory control, and forecasting models would affect the overall system performance. As mentioned in section 5 of chapter V, an analysis of the forecasting needs for this case study was conducted by Yamada (1987). His investigation revealed that, with the available data, only the demand for raw materials was forecastable. Hence, the lack of sufficient data prevented the inclusion of the forecasting models as originally intended. Thus, the flexible simulation model (FSM) was used for the purpose of conducting an investigation for determining which combination of production scheduling and inventory control models would provide the best performance. This chapter presents the results obtained from the case study.

Two production sequencing rules for the Sign Shop and five inventory models for the Sign Shop and the districts' warehouses were selected in this study for the purpose of comparing the performance of different combinations of
these models. The numbers of these models are represented as follows:

(A) Inventory models for the districts’ warehouses:

(1) Periodic-Review.
(2) Order Up To \( I_{\text{MAX}} \).
(3) Continuous-Review.
(4) Base Stock.
(5) On-Demand (On-Order).

(B) Inventory models for the Sign Shop:

Same as (A) above.

(C) Production sequencing rules at the Sign Shop:

(1) Shortest processing time (SPT).
(2) Shortest slack time (SST).

This means that fifty different system combinations (\( 5 \times 5 \times 2 \)) of models are possible. The FSM was configured to represent each of the fifty model combinations and actual demand data was used as input.

Discussions with ODOT personnel revealed that the two most important measures are total inventory carrying costs and the percentage of orders satisfied from inventory (a measure of service). The penalty costs for not providing
adequate service are described below.

If an order cannot be satisfied from inventory, then, for critical signs, the order would have to be expedited in order to deliver it promptly. There are three major components associated with the cost of expediting an order: (1) interrupting the production schedule, which requires additional setups, (2) mileage costs for the special trip to be made for delivering the order, and (3) the cost of the driver making the trip.

Based on discussions with knowledgeable personnel the following estimates were obtained: (1) The average number of hours associated with interrupting the production equals 1.5 hours per order spent on additional setup, (2) Average trip length is 250 miles, (3) Average trip time is 6 hours, (4) Average time for loading and unloading the shipment is 2 hours, (5) The average number of orders per year is 4000 orders.

Assuming that labor rate equals $5.00 per hour for the driver and $10.00 per hour for the machine operator, and that the average mileage costs are $0.25 per mile, the following can be calculated:

(1) The number of orders to be expedited equals (Total number of orders for the year) multiplied by (The percentage of orders not satisfied from inventory).
(2) The cost of expediting per order =

\[
\text{mileage costs} = 250 \text{ miles} \times \$0.25 \text{ per mile} = \$62.50 \\
+ \text{driver costs} = (6+2) \text{ hours} \times \$5.00 \text{ per hour} = 40.00 \\
+ \text{setup cost} = 1.5 \text{ hours} \times \$10.00 \text{ per hour} = 15.00 \\
\text{------} \\
\text{total per order} = \$117.50
\]

Table 6.1 shows the performance of the system with respect to these two measures. Column 1 of this Table represents the fifty models combinations. Column 2 shows the total inventory carrying costs for both the warehouses and the Sign Shop per period. The percentage of orders satisfied from inventory are shown in column 3. Finally, column 4 shows the penalty costs of expediting orders plus the inventory carrying costs per year. These calculations are illustrated below. Note that the percentage of orders not satisfied from inventory equal one minus the value shown in column 3.

Examining Table 6.1 further, combination 5,3,1 (On-Demand inventory model at the warehouses, continuous-review inventory model at the Sign Shop, and the shortest processing time sequencing rule) performed best with respect to the expediting costs:

\[
\text{Expediting cost} = \text{total orders to be expedited} \times \text{expediting costs per order} \\
= [(4000) \times (1-0.4763)] \times \$117.50 \\
= \$246,139.00 \text{ per year}
\]
<table>
<thead>
<tr>
<th>Models</th>
<th>Total Inventory Carrying Costs ($/week)</th>
<th>% of Orders Satisfied from Inventory</th>
<th>Penalty Costs for not Providing Adequate Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,3,2</td>
<td>9953.67</td>
<td>0.3873</td>
<td>805,559.84</td>
</tr>
<tr>
<td>3,2,2</td>
<td>9865.88</td>
<td>0.2099</td>
<td>884,372.76</td>
</tr>
<tr>
<td>2,2,2</td>
<td>9631.22</td>
<td>0.1775</td>
<td>887,398.44</td>
</tr>
<tr>
<td>1,2,2</td>
<td>9663.35</td>
<td>0.1688</td>
<td>893,140.35</td>
</tr>
<tr>
<td>3,1,2</td>
<td>9833.55</td>
<td>0.2088</td>
<td>883,208.60</td>
</tr>
<tr>
<td>2,1,2</td>
<td>9631.22</td>
<td>0.1775</td>
<td>887,398.44</td>
</tr>
<tr>
<td>1,1,2</td>
<td>9639.65</td>
<td>0.1678</td>
<td>892,395.80</td>
</tr>
<tr>
<td>1,4,1</td>
<td>9942.95</td>
<td>0.3478</td>
<td>823,567.40</td>
</tr>
<tr>
<td>5,3,2</td>
<td>1500.81</td>
<td>0.4754</td>
<td>324,562.81</td>
</tr>
<tr>
<td>5,3,1</td>
<td>1495.10</td>
<td>0.4763</td>
<td>323,884.20</td>
</tr>
<tr>
<td>1,5,1</td>
<td>9107.85</td>
<td>0.0000</td>
<td>943,608.20</td>
</tr>
<tr>
<td>2,5,1</td>
<td>9072.43</td>
<td>0.0000</td>
<td>941,766.36</td>
</tr>
<tr>
<td>3,5,1</td>
<td>9518.65</td>
<td>0.0000</td>
<td>964,969.80</td>
</tr>
<tr>
<td>2,5,2</td>
<td>9087.77</td>
<td>0.0000</td>
<td>942,564.04</td>
</tr>
<tr>
<td>1,5,2</td>
<td>9124.78</td>
<td>0.0000</td>
<td>944,488.56</td>
</tr>
<tr>
<td>1,1,1</td>
<td>9643.73</td>
<td>0.1678</td>
<td>892,607.96</td>
</tr>
<tr>
<td>2,2,1</td>
<td>9635.30</td>
<td>0.1775</td>
<td>887,610.60</td>
</tr>
<tr>
<td>3,3,1</td>
<td>10333.32</td>
<td>0.4796</td>
<td>781,720.64</td>
</tr>
</tbody>
</table>

Table 6.1 Average Inventory Carrying Costs Per Period for Both the Warehouses and the Sign Shop, the Percentage of Orders Satisfied from Inventory, and Penalty Costs for not Providing Adequate Service.
<table>
<thead>
<tr>
<th>Models</th>
<th>Total Inventory Carrying Costs ($/week)</th>
<th>% of Orders Satisfied from Inventory</th>
<th>Penalty Costs for not Providing Adequate Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>4,4,1</td>
<td>10320.28</td>
<td>0.3769</td>
<td>829,511.56</td>
</tr>
<tr>
<td>5,5,1</td>
<td>00.00</td>
<td>0.0000</td>
<td>470,000.00</td>
</tr>
<tr>
<td>2,1,1</td>
<td>9635.30</td>
<td>0.1775</td>
<td>887,610.60</td>
</tr>
<tr>
<td>3,1,1</td>
<td>9833.44</td>
<td>0.2088</td>
<td>883,202.88</td>
</tr>
<tr>
<td>1,2,1</td>
<td>9667.43</td>
<td>0.1687</td>
<td>893,417.36</td>
</tr>
<tr>
<td>3,2,1</td>
<td>9865.88</td>
<td>0.2099</td>
<td>884,372.76</td>
</tr>
<tr>
<td>1,3,1</td>
<td>9963.06</td>
<td>0.3927</td>
<td>803,510.12</td>
</tr>
<tr>
<td>2,3,1</td>
<td>9950.22</td>
<td>0.3841</td>
<td>806,884.44</td>
</tr>
<tr>
<td>5,1,1</td>
<td>836.03</td>
<td>0.2193</td>
<td>410,402.56</td>
</tr>
<tr>
<td>5,2,1</td>
<td>865.48</td>
<td>0.2219</td>
<td>410,711.96</td>
</tr>
<tr>
<td>3,3,2</td>
<td>0333.32</td>
<td>0.4796</td>
<td>781,920.64</td>
</tr>
<tr>
<td>2,3,2</td>
<td>9943.39</td>
<td>0.3757</td>
<td>810,477.28</td>
</tr>
<tr>
<td>3,4,2</td>
<td>0300.81</td>
<td>0.4337</td>
<td>801,803.12</td>
</tr>
<tr>
<td>2,4,1</td>
<td>9962.90</td>
<td>0.3476</td>
<td>824,698.80</td>
</tr>
<tr>
<td>3,4,1</td>
<td>0309.34</td>
<td>0.4337</td>
<td>802,246.68</td>
</tr>
<tr>
<td>4,3,2</td>
<td>0348.23</td>
<td>0.4754</td>
<td>784,669.96</td>
</tr>
<tr>
<td>4,3,1</td>
<td>0355.15</td>
<td>0.4763</td>
<td>784,606.80</td>
</tr>
<tr>
<td>5,2,2</td>
<td>865.33</td>
<td>0.2219</td>
<td>410,704.16</td>
</tr>
<tr>
<td>4,2,2</td>
<td>9978.37</td>
<td>0.2219</td>
<td>884,563.00</td>
</tr>
<tr>
<td>4,2,1</td>
<td>9977.45</td>
<td>0.2219</td>
<td>885,574.40</td>
</tr>
</tbody>
</table>

Table 6.1 (Continued).
<table>
<thead>
<tr>
<th>Models</th>
<th>Total Inventory Costs ($/week)</th>
<th>% of Orders Satisfied from Inventory</th>
<th>Penalty Costs for not Providing Adequate Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,1,2</td>
<td>836.03</td>
<td>0.2193</td>
<td>410,402.50</td>
</tr>
<tr>
<td>4,1,2</td>
<td>9945.94</td>
<td>0.2193</td>
<td>884,117.88</td>
</tr>
<tr>
<td></td>
<td>9964.72</td>
<td>0.2193</td>
<td>885,094.44</td>
</tr>
<tr>
<td>2,4,2</td>
<td>10012.67</td>
<td>0.3640</td>
<td>819,578.84</td>
</tr>
<tr>
<td>1,4,2</td>
<td>9998.54</td>
<td>0.3648</td>
<td>818,468.08</td>
</tr>
<tr>
<td>5,4,2</td>
<td>1315.98</td>
<td>0.4070</td>
<td>347,140.96</td>
</tr>
<tr>
<td>5,4,1</td>
<td>1260.30</td>
<td>0.3781</td>
<td>357,813.00</td>
</tr>
<tr>
<td>4,4,2</td>
<td>10397.56</td>
<td>0.4083</td>
<td>818,772.12</td>
</tr>
<tr>
<td>5,5,2</td>
<td>00.00</td>
<td>0.0000</td>
<td>470,000.00</td>
</tr>
<tr>
<td>4,5,2</td>
<td>9523.65</td>
<td>0.0000</td>
<td>965,229.80</td>
</tr>
<tr>
<td>4,5,1</td>
<td>9506.99</td>
<td>0.0000</td>
<td>964,363.48</td>
</tr>
<tr>
<td>3,5,2</td>
<td>9522.45</td>
<td>0.0000</td>
<td>965,167.40</td>
</tr>
</tbody>
</table>

Table 6.1 (Continued).
Total inventory carrying costs for combination 5,3,1 =

$1,495.10 \text{ per week } \times 52 \text{ weeks } = $77,745.20

Therefore the penalty costs for combination 5,3,1 =

$323,884.20 \text{ per year.}

On the other hand, combinations where inventory was not maintained in at the Sign Shop, such as combination 4,5,2 (base stock inventory model at the warehouses, on-demand inventory model at the Sign Shop, and the shortest slack time sequencing rule) proved to be most costly, as anticipated, with a total penalty cost of $965,229.80 per year.

Based on the above analysis, if the objective is to minimize the costs of expediting orders and carrying inventory, then combination 5,3,1 is optimum. It should be noted that, under this combination, no inventory is maintained at the warehouses.
Chapter VI presented the results obtained from the case study. This chapter concludes this thesis with a summary of the research and the findings, and citing further research problems.

1. SUMMARY OF RESEARCH:

This thesis described the development of a flexible simulation model (FSM) which can be used to help design integrated production planning and inventory control (IPPIC) systems. Recognizing that there is a wide variety of production and distribution system (PDS), the current version of the FSM involves a production and distribution system which consists of a flow shop production facility and a single echelon of geographically decentralized warehouses. Examples of other types of PDSs are: (1) a job shop production facility with a single or multi echelon system of geographically decentralized warehouses and (2) two or more decentralized production facilities with a system of geographically decentralized warehouses.
The model consists of three different model banks: (1) forecasting models, (2) production sequencing models, and (3) inventory control models. Each bank consists of a number of specific model alternatives from which the IPPIC designer can choose.

In order to accommodate the objective of developing the FSM in a modular fashion, structured systems analysis and development methodology was used. These tools were used to (1) accommodate any changes that would be required when custom tailoring the FSM, and (2) provide the best foundation for the modifications that would be required for the above task. The thesis showed how this methodology was utilized to generate the specifications of the model.

2. SUMMARY OF FINDINGS:

(1) Studies and research made on structured systems analysis and development by knowledgeable sources such as James Martin (1985), Ward and Mellor (1985), and Thomas DeMarco (1978) provided evidence to the fact that the use of such methodology is a very effective and systematic approach to analyzing and designing systems. Such evidence, along with a list of some of the important advantages of using this methodology, was cited in chapter III. The author's experience from using this approach supports this evidence. In comparison to previous
assignments which were similar and structured systems analysis was not used, the author (1) experienced less errors during the development and program code phases, (2) was able to achieve sound data administration and rigorous interfaces between separately developed modules, and (3) was able to simplify complex system components for ease of understanding.

(2) The FSM demonstrated that the different elements of a production planning and inventory control system do have interdependencies which, together, control the overall performance of the system. Hence, the way by which the components of the system are interrelated should always be considered.

(3) The FSM can be used to achieve better designs of IPPIC systems. This, in turn, can achieve better system performance of production scheduling at the production facility, and inventory control at the production facility and the warehouses.

(4) An analysis of the results revealed that excessive inventory carrying costs were a result of the fact that the signs could remain in inventory for long periods of time. In order to reduce these excessive costs, it is suggested that instead of keeping inventory of finished signs, the
Sign Shop should stock them as semi-finished signs (i.e. maintain sub-level inventories of signs) in order to reduce excessive inventory costs. In other words, manufacture the sign up to a certain level where from that level on the semi-finished sign can be used to manufacture any of a number of different signs. This is possible because a large number of signs have the same size and shape, and are made from the same base material.

(5) The Sign Shop could also reduce excessive inventory costs by stocking only those signs with high or continual demand. In the case where a sign is not carried in inventory, the need for a short reaction time may necessitate the use of overtime or subcontracting.

(6) In order to maximize the flow of production through the Sign Shop, it was recognized that the flow through the reflective-sheeting and trimming operation should be maximized. This operation was identified to be the bottleneck operation (the bottleneck operation is defined to be the operation which does not have the capacity to produce at least the demand during a specified period).

(7) At the time this study was conducted, a procedure by which the inventory shortage costs of the signs where calculated did not exist. During the course of the study
it was found that some signs are more important to have available in stock than others. For example, it is always more important to have a "STOP" sign in inventory than a "route marker" sign. Reasons for this is that if a STOP sign was missing, it may cause tragic accidents to take place. Whereas, if a route marker is missing or not available in stock, it would not have the same affect. This suggests that a study should be made to categorize the signs in terms of how important it is to have a sign available in inventory. Then, for each category assign a shortage cost which corresponds to what actions would be taken in order to replenish the unavailable or missing signs.

(8) As mentioned in section 1.3, the average utilization of the machines was very low in comparison to their maximum utilization. An analysis of the data indicated that during some periods, due to the lack of, or very low demand, some of the machines were idle, or had very low utilization. It is suggested that, during those periods where demand is very low, the Sign Shop should schedule the production of (1) signs that are predicted to be in demand in future periods and (2) signs whose inventory level is lower than acceptable. This concept can be used to reduce the load on the Sign Shop.
(9) Utilizing the concept described in 11 above, labor efficiency can also be improved by (1) manufacturing signs that are predicted to be in demand in future periods during periods with low demand, (2) manufacturing signs whose inventory is lower than acceptable during periods with low demand, and (3) allocate labor at machine centers that have low demand to those with high demand.

3. FUTURE RESEARCH PROBLEMS

(1) The current version of the FSM considers only the case where the system consists of a flow shop production facility which services a single echelon system of geographically decentralized warehouses. More research is suggested to enhance the current system to consider other types of production and distribution systems such as the ones mentioned in chapter I. This task can be accomplished very efficiently by using the structured systems analysis and development approach described in chapter III.

(2) The inventory models used in the current version of the FSM implement a "pre-stocking" approach to control the inventory of signs. This approach focuses on the reorder quantities which should be put in inventory. Another approach that should be considered for controlling the inventories is known as "post-stocking". This approach
focuses on how much of the present inventory already in stock should be declared as surplus and disposed of. It would allow the districts to dispose any excess inventory to other districts that are in need for it. Such dispositions would reduce the inventory carrying costs at the district disposing the inventory. It also reduces the load on the Sign Shop because, without this post-stocking model, the districts which need the signs would have to order them from the Sign Shop instead.

(3) The developed simulation model does not consider sub-level inventories of products. The model would be a more powerful tool if further enhancements are done so it would include this feature. Under this approach, products are stored at a semi-finished stage. This is very effective when a semi-finished product can be used to produce a number of different products at their finished stage.

(4) It was suggested previously that when a sign is not carried in inventory, the need for a short reaction time may necessitate the use of overtime or subcontracting. This could be accomplished by, first, using the structured analysis techniques to modify the system's specifications to consider such a case. This modification would include building a new module and identifying its components and
the input/output data requirements. Once the different components and data requirements are completely and clearly defined, the program code can be written. This module can then be used to study the feasibility of the using overtime and subcontracting to satisfy the need for a short reaction time.

(5) Since the demand for signs greatly varied from zero to some high value, it suggested that the Sign Shop, during periods with low demand, schedule the production of (1) signs that are predicted to be in demand in future period and (2) signs whose inventory level is lower than acceptable. The current version of the system does not consider this task, but it can be easily modified by using the structured systems analysis techniques as explained in (4) above.

(6) The model can also be modified to include more production sequencing rules such as the ones mentioned in chapter IV. These rules include (1) sequence the jobs by their minimum machine make span if the bottleneck makes more than one part of the same product, (2) sequence the jobs using a travelling salesman algorithm if the bottleneck involves setup time, (3) if one bottleneck feeds another, then sequence the jobs through the two machines to minimize their make span, and other recognized and
applicable sequencing rules.

(7) The system developed in this research assumed that the process specification for manufacturing the products had already been determined and justified. Since it is very likely that process specifications can be considerably improved with the high advances in technology, to include the task of improving process specifications in the IPPIC operations could increase the productivity and performance of the overall system. This could accomplished by, first, using the structured analysis techniques to modify the system’s specifications to include such a task. This modification would include building a new module and identifying its components and the input/output data requirements. Once the components and data requirements are completely and clearly defined, the program code can be written. This module can then be used to study the feasibility of changing and modifying process specifications.

(8) The current version of the FSM does not have the flexibility to change the production sequencing rule from one period to another. It is recommended that an expert system be developed and embedded in the FSM to achieve this task. This expert system would be used to experimentally determine the system’s behavior when changing the rules
from one period to another. To possess the ability of adaptively changing the sequencing rule by period, an algorithm which allows the system to switch from one rule to another should be developed. The decisions made by the algorithm should be based upon the status of the shop in terms of some given criteria.

(9) The value of the FSM as a prototype tool will be especially enhanced if it were modified to generate a set of reports at the end of each period. This will permit the user to pilot test the entire system. When these pilot tests indicate that the system needs to be modified, the structured systems development method will make it possible to accomplish these modifications with minimal efforts.

(10) As mentioned in chapters I and V, it was assumed that the products were shipped from the production facility by a commercial carrier. As a result, the transportation capabilities were assumed to be adequate. It is suggested that the capability of the FSM be enhanced to analyze the possibilities of having a finite transportation fleet.
REFERENCES


Interfaces, pp. 81-96, November, 1975.


APPENDIX A

JOB SHOP LINEAR PROGRAMMING MODEL (JSLP)
In chapter II, it was mentioned that the linear programming (LP) model described applies only to one flow line type of facility (flow shop LP (FSLP)) and does not consider the case where multiple routes can be used to manufacture a product (job shop LP (JSLP)). For this case the LP model would have to be modified to consider the multiple routes by which products can be produced. The section below provides a description of this JSLP model and its notations.

Stated mathematically, the objective is to minimize

\[
Z = \sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{p=1}^{P} \sum_{s=1}^{S} \left[ C_{t,m,p} X_{t,m,p,s} + H_{t,p}(I_{t,p}) \right]
\]

The first term in the brackets represents the cost of producing \(X\) units of product \(p\) during period \(t\) on machine \(m\) at stage \(s\). The second term represents the cost of holding products in inventory.

There are three constraints which must be satisfied:

1) \(X_{t,p} + I_{(t-1),p} = D_{t,p} + I_{t,p}\)

This constraint states that the number of units to be produced of product \(p\) during period \(t\) plus the inventory \(I_{(t-1),p}\) of that product during period \(t-1\) must be equal to the demand \(D_{t,p}\) plus the inventory \(I_{t,p}\) of the
same product.

2) \[ \sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{p=1}^{P} \sum_{s=1}^{S} x_{tmps} (PT_{mps}) \leq MH_{tms} \]

The third constraint states that the total time required to produce \( X \) units of products during period \( t \) must be least equal to the machine hours available \( (MH_{tms}) \) for that period.

3) The fourth constraint states that, if a product \( P \) requires \( X \) units of product \( P1 \) and \( Y \) units of product \( P2 \) at stage \( i \) to produce \( Z \) units of \( P \) at stage \( k \), then

(1) the number of units to be produced of product \( P1 \) at stage \( i \) equals \( (X/(X+Y))Z \) and
(2) the number of units to be produced of product \( P2 \) at stage \( i \) equals \( (Y/(Y+Z))Z \).

To illustrate the above, assume that a facility manufactures paper where two stages are required for producing the final product as shown on the following page:
The above shows that six machines can be used to manufacture the final products via two different stages. It also shows the different possible routes by which the products can be manufactured. Assume that

1. $X_{t,m} = \text{product } 0$, on machine $m$, at stage 2, at time $t$
2. $X_{t1,m1} = \text{product } 1$, on machine $m$, at stage 1, at time $t$
3. $X_{t2,m1} = \text{product } 2$, on machine $m$, at stage 1, at time $t$
4. $X_{t,m} = \text{requires two units of } X_{t1,m1} \text{ and one unit of } X_{t2,m1}$.

Hence, the following constraints would be added in order to consider the different possible routes by which the products can be manufactured:
\[ \sum_{t=1}^{T} \sum_{m=1}^{M} x_{t,m1} = (2/3) \sum_{t=1}^{T} \sum_{m=1}^{M} x_{t,m2} \]
\[ \sum_{t=1}^{T} \sum_{m=1}^{M} x_{t,m1} = (1/3) \sum_{t=1}^{T} \sum_{m=1}^{M} x_{t,m2} \]

where

\[ C_{t,m} \] = production cost of product \( p \) during period \( t \) on machine \( m \) at stage \( s \),

\[ X_{t,m} \] = number of units to be produced of product \( p \) during period \( t \) on machine \( m \) at stage \( s \),

\[ HC_{t,p} \] = inventory holding costs of product \( p \) during time \( t \),

\[ I_{t,p} \] = inventory level of product \( p \) during period \( t \),

\[ D_{t,p,s} \] = demand of product \( p \) during period \( t \) at stage \( s \),

\[ PT_{t,m} \] = processing time required to produce product \( p \) on machine \( m \) at stage \( s \),

\[ MH_{t,m} \] = machine hours available on machine \( m \) during period \( t \) at stage \( s \),

\( T \) = the number of periods in the planning horizon,

\( K \) = the process number (machine centers),

\( P \) = number of product,
APPENDIX B

DATA AND PROCESS SUMMARY FILES
(I) DATA SUMMARY

Data store: ANALYZED-DATA

Description:
Contains an analysis of the data regarding the dispatches of products from the warehouses to the market and the forecastable demand of products.

Updated by
Analyze-Historical-Data

Referenced by
Implement-Forecasting-Model

Composition:
= product-ID
+ product-dispatches
+ forecastable-demand-of-product
+ demand-pattern

Data flow: BACK-ORDERS

Description:
Contains the list of orders received from the warehouses for which the production facility was not able to satisfy. In other words, those products were not available in inventory.

Origin
*** Off the diagram ***
Process-as-Back-Orders

Destination
Apply-PF-On-Order-Policy
Products-Inv.-Control
Composition:

= product-name
+ product-ID
+ ordered-quantity
+ warehouse-ID
+ date-ordered
+ date-wanted

Data store: BILL-OF-MATERIALS

Description:

This data store contains a list of the raw material, components, or units required to manufacture the products.

Updated by
Determine-Engr.-Specs.

Referenced by
Formulate-RHS-Vector
Determine-Mfg.-Processes
Identify-Mtl.-Needed
Modify-Files

Composition:

= product-name
+ product-ID
+ material-name
+ material-ID
+ quantity-needed
Data primitive: BN-INVOLES-SETUP

Description:
Indicates that the bottleneck operation involves a setup time.

Origin
Classify-Sequencing-Problem

Destination
Solve-Travel-Sales-Prob.

Data primitive: BN-MAKES-MORE-TAN-1-PART

Description:
Indicates that the bottleneck operation makes more than one part of the same product to be produced.

Origin
Classify-Sequencing-Problem

Destination
Sched-by-Min-1-M/C-Make-Span

Data flow: BN-SCHEDULE

Description:
Contains the information regarding the schedule of the bottleneck operation(s) generated by the respective transformations.

Origin
*** Off the diagram ***
Schedule-Bottleneck
Sched-by-Remaining-PT
Sched-by-Min-2-M/C-Make-Span
Solve-Travel-Sales-Prob.

Destination
Determine-T-to-Release-Jobs
Start-Pre-BN-Operations
*** Off the diagram ***
*** Off the diagram ***
*** Off the diagram ***
Sched-by-Min-1-M/C-Make-Span *** Off the diagram ***
Sched-By-Due-Date *** Off the diagram ***

Composition:

= machine-center-name
+ machine-center-number
+ the-list-of-jobs
+ time-to-release-each-job

Data store: BOTTLENECK-OPERATION-FILE

Description:

This data store contains the name and number of the bottleneck operations.

Updated by
Classify-as-a-BN Referenced by
Classify-Sequencing-Problem
Determine-BN Schedule-Bottleneck
Identify-BN.-Operations

Composition:

= machine-center-name
+ machine-center-number

Data flow: DEMANDED-PRODUCTS

Description:

These are the products ordered, which already exist and are not new products.

Origin
*** Off the diagram ***
Destination
Apply-WH-On-Demand-Policy
Identify-Demanded-Product

Composition:

= product-name
+ product-ID
+ quantity-demanded
+ date-wanted-by
+ warehouse-ID

Data flow: DESIRED-CHANGES

Description:

Represent the changes issued by the USER regarding policies, product specifications, cost parameters, operations data, materials, etc.

Origin

*** Off the diagram ***

User

Destination

Modify-Files

IPPIC-Operations

Composition:

= Inv-policies-changes
+ production-policy-changes
+ product-specification-changes
+ cost-parameters-changes
+ etc.
Data flow: DISPATCH-INFO.

Description:

This data flow contains the information regarding the products that has been dispatched from the warehouses to the market.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispatch-Products</td>
<td>Update-WH-Inventory</td>
</tr>
<tr>
<td>*** off the diagram ***</td>
<td>Identify-WH-Policy</td>
</tr>
<tr>
<td>*** off the diagram ***</td>
<td>Apply-WH-Inventory-Policy</td>
</tr>
<tr>
<td>*** off the diagram ***</td>
<td>Apply-WH-Continuous-Policy</td>
</tr>
</tbody>
</table>

Composition:

= product-ID
+ product-name
+ dispatch-quantity
+ warehouse-ID

Data flow: DISPATCHED-PRODUCTS

Description:

This data flow represents the products that have been dispatched to the market in response to some earlier demand.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>*** Off the diagram ***</td>
<td>Apply-WH-Continuous-Policy</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Apply-WH-Inventory-Policy</td>
</tr>
<tr>
<td>Receive-Dispatch-Products</td>
<td>Inv.-Control-Operations</td>
</tr>
</tbody>
</table>
Composition:

= product-name
+ product-ID
+ dispatched-quantity
+ warehouse-ID

**Data store:** DPMF-INVENTORY-POLICIES

**Description:**

These policies consists of two main categories, raw materials inventory policies and finished products inventory policies. Each category consists of the different types of policies (continuous and periodic) and the parameters needed to implement the policy. The policy to be implemented for controlling the inventory is selected by the user.

<table>
<thead>
<tr>
<th>Updated by</th>
<th>Referenced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Identify-PF-Inv-Policy</td>
</tr>
<tr>
<td></td>
<td>Identify-PF-Policy</td>
</tr>
<tr>
<td></td>
<td>Apply-PF-On-Order-Policy</td>
</tr>
<tr>
<td></td>
<td>Apply-PF-Inventory-Policy</td>
</tr>
<tr>
<td></td>
<td>Raw-Materials-Control</td>
</tr>
<tr>
<td></td>
<td>Orders-Inv.-Operations</td>
</tr>
<tr>
<td></td>
<td>DPMF-Operations</td>
</tr>
</tbody>
</table>

Composition:

= raw-materials-inventory-policy
+ finished-products-inv.-policy
**Data store: DPMF-OUTSTANDING-ORDERS**

Description:
Contains the information regarding the orders for products received from the warehouses and their status.

**Updated by**
- Process-as-Back-Orders
- Update-PF-Orders-File
- Shipments-Inv-Control
- Process-Production-Orders

**Referenced by**
- Process-as-Back-Orders
- Update-PF-Orders-File
- Generate-DPMF-Reports
- Shipments-Inv-Control

**Composition:**
- order-number
- product-name
- product-ID
- quantity-ordered
- order-status
- date-ordered
- date-wanted
- date-satisfied
- warehouse-ID

**Data store: DPMF-PRODUCTS-FILE**

Description:
This file contains the list of products that are presently manufactured at the production facility.

**Updated by**
- Update-Products-List

**Referenced by**
- Identify-Ordered-Product
Composition:

\[ \text{product-name} + \text{product-ID} + \text{product-description} \]

**Data store: DPMF-PRODUCTS-INVENTORY**

Description:

This data store contains the information regarding the inventory levels of the finished products available in inventory at the production facility.

**Updated by**

- Process-Post-BN-Operations
- Synchronize-Production
- Start-Post-BN-Operations
- Production-Operations
- Production-Plan.-Control

**Referenced by**

- Apply-PF-Periodic-Policy
- Apply-PF-Continuous-Policy
- Apply-PF-Inventory-Policy
- Check-Avail.-of-Products
- Retrieve-Ship-Products
- Generate-DPMF-Reports
- Shipments-Inv-Control
- Orders-Inv.-Operations
- Products-Inv.-Control

Composition:

\[ \text{product-name} + \text{product-ID} + \text{quantity-in-inventory} \]
Data flow: DPMF-REPORTS

Description:

These reports are generated so the user can make decisions regarding the inventory and production policies being implemented at the production facility. These reports provide information regarding the facilities performance with respect to (1) raw materials inventory levels, (2) finished products inventory levels, (3) work-in-process inventory, and (4) delivery lead times.

Origin

Generate-DPMF-Reports
DPMF-Operations
Orders-Inv.-Operations

Destination

*** Off the diagram ***
*** Off the diagram ***
*** Off the diagram ***

Component of:

reports

Composition:

raw-mtl-inv-reports
+ finished-products-inv-reports
+ production-status
+ work-in-process-inv-reports

Data flow: ENGR.-SPECS.

Description:

Contains the engineering specifications generated from DETERMINE-ENGR.-SPECS. for the new products.
Origin
Determine-Engr.-Specs.

Destination
Determine-Mfg.-Processes

Composition:

= product-name
+ product-ID
+ dimensions
+ color + material-name
+ material-ID
+ etc.

Event flow: EV-B.O.

Description:

Signals the respective transformation(s) that orders have been tagged as back-orders.

Origin
Process-as-Back-Orders

Destination
Products-Inv.-Control

*** Off the diagram ***

Identify-PF-Policy

Event flow: EV-BN-COMPLETED

Description:

This event flow signals the respective transformation(s) that the bottleneck operation has been completed.

Origin
*** Off the diagram ***

Process-BN-Operation

Destination
Release-Jobs-From-II

*** Off the diagram ***
Event flow: **EV-BN-IDENTIFIED**

**Description:**
This event flow signals the respective transformation(s) that the bottlenecks have been identified.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine-BN</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Identify-BN.-Operations</td>
<td>Schedule-Bottleneck</td>
</tr>
</tbody>
</table>

Event flow: **EV-CONTINUOUS**

**Description:**
This flow indicates that the inventory policy to be implemented for controlling the inventory of products is the continuous policy.

Event flow: **EV-DATA-ANALYZED**

**Description:**
This flow indicates that the data regarding the demand and the dispatches of products from the warehouses to the market have been analyzed, so that the forecasting model can be implemented to forecast the demand.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyze-Historical-Data</td>
<td>Implement-Forecasting-Model</td>
</tr>
</tbody>
</table>
Event flow: **EV-ENAB.-DISAB.**

Description:

This is an event flow that represents an ENABLING or a DISABLING action generated by a control process. With this control signal, the control process either enables or disables a transformation.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTROL-.....1-.....2</td>
<td>Apply-PF-Continuous-Policy</td>
</tr>
<tr>
<td>CONTROL-.....1-.....2</td>
<td>Apply-PF-Periodic-Policy</td>
</tr>
<tr>
<td>CONTROL-2.1.3.5.1-2.1.3.5.2</td>
<td>Apply-PF-Inventory-Policy</td>
</tr>
<tr>
<td>CONTROL-2.1.3.5.1-2.1.3.5.2</td>
<td>Apply-PF-On-Order-Policy</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Identify-WH-Inv-Policy</td>
</tr>
<tr>
<td>CONTROL-2.3.2.2.3</td>
<td>Apply-WH-Continuous-Policy</td>
</tr>
<tr>
<td>CONTROL-2.3.2.2.3</td>
<td>Apply-Periodic-policy</td>
</tr>
<tr>
<td>CONTROL-2.3.2.2-2.3.2.3</td>
<td>Apply-WH-Inventory-Policy</td>
</tr>
<tr>
<td>CONTROL-.....5.1-.....5.2</td>
<td>Process-Post-BN-Operations</td>
</tr>
<tr>
<td>CONTROL-.....4.1-.....4.2</td>
<td>Process-BN-Operation</td>
</tr>
<tr>
<td>CONTROL-3.2.5.3.-3.3</td>
<td>Release-Jobs-for-Production</td>
</tr>
<tr>
<td>CONTROL-2.3.2.2-2.3.2.3</td>
<td>Apply-WH-On-Demand-Policy</td>
</tr>
<tr>
<td>CONTROL-2.3.1.4</td>
<td>Update-WH-Inventory</td>
</tr>
</tbody>
</table>

Event flow: **EV-JOBS-GROUPED**

Description:

This flow signals the respective transformation(s) that the orders (jobs) have been grouped and ready to be scheduled for production.
Event flow: EV-LP-FORMULATED

Description:
This flow signals the respective transformation(s) that the linear program has been formulated.

Event flow: EV-LP-SOLVED

Description:
This flow signals the respective transformation(s) that the linear program has been solved.
Event flow: EV-MTL-ARRIVED

Description:

EV-MTL-ARRIVED (EV-material-arrived) signals the transformation PRODUCTION-PLAN-CONTROL that raw materials that was ordered have arrived.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update-Inv-Orders-Status</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Select-Respective-Orders</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Process-as-Back-Orders</td>
</tr>
<tr>
<td>Raw-Materials-Control</td>
<td>Production-Plan.-Control</td>
</tr>
</tbody>
</table>

Event flow: EV-MTL-AVAILABLE

Description:

This event flow signals the respective transformations that the materials required for manufacturing the products are available.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check-Avail.-of-Mtl</td>
<td>Production-Operations</td>
</tr>
</tbody>
</table>

Event flow: EV-MTL-NOT-AVAILABLE

Description:

This event flow signals RAW-MATERIALS-CONTROL that raw materials are not available in inventory for the production of some product(s).
**Event flow: EV-NEW-PRODUCT**

**Description:**

This event flow indicates that the products ordered are new orders which have never been ordered before.

**Event flow: EV-ON-DEMAND**

**Description:**

This flow indicates that the policy being implemented for controlling the inventory is the On-Demand policy.

**Event flow: EV-OTHER**

**Description:**

This flow indicates that the inventory policy used to control the inventory of products is not the On-Demand
policy. It is either the continuous inventory policy, or the periodic inventory policy.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify-PF-Policy</td>
<td>CONTROL-2.1.3.5.1-2.1.3.5.2</td>
</tr>
<tr>
<td>Identify-Raw-Mtl-Policy</td>
<td>Apply-Inventory-Policy</td>
</tr>
<tr>
<td>Identify-Inventory-Policy</td>
<td>CONTROL-2.3.2.2-2.3.2.3</td>
</tr>
</tbody>
</table>

**Event flow: EV-PERIODIC**

**Description:**

This event flow indicates that the policy to be implemented for controlling the inventory of products is the periodic policy.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify-PF-Inv-Policy</td>
<td>CONTROL-.....1-.....2</td>
</tr>
<tr>
<td>Identify-WH-Inv-Policy</td>
<td>CONTROL-2.3.2.2.3</td>
</tr>
</tbody>
</table>

**Event flow: EV-PRE-BN-COMPLETED**

**Description:**

This flow signals the respective transformation(s) that the processing of jobs on the machine centers that precede the bottleneck(s) have been completed.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>*** Off the diagram ***</td>
<td>Release-Jobs-from-I</td>
</tr>
<tr>
<td>Release-Jobs-for-Production</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Start-Pre-BN-Operations</td>
<td>Start-BN-Operation</td>
</tr>
</tbody>
</table>
**Event flow: EV-PRODUCT-AVAILABLE**

**Description:**

This is an event flow signaling the transformation DISPATCH-PRODUCTS that the products demanded are available in inventory. This event flow is generated from the transformation CHECK-WH-INVENTORY.

**Origin**

Check-WH-Inventory

**Destination**

Dispatch-Products

---

**Event flow: EV-PRODUCTION-COMPLETED**

**Description:**

EV-PRODUCTION-COMPLETED signals the respective transformation(s) that the production of jobs has been completed.

**Origin**

*** Off the diagram ***
Production-Plan.-Control
Process-Post-BN-Operations
Start-Post-BN-Operations
Synchronize-Production
*** Off the diagram ***
Production-Operations
*** Off the diagram ***
Production-Plan.-Control

**Destination**

Apply-Inventory-Policy
Raw-Materials-Control
*** Off the diagram ***
*** Off the diagram ***
*** Off the diagram ***
Retrieveship-Products
*** Off the diagram ***
Shipments-Inv-Control
Orders-Inv.-Operations
**Event flow: EV-PRODUCTS-AVAIL.**

**Description:**
This flow signals the transformation RETRIEVE-SHIP-PRODUCTS that the products ordered from the warehouses are available in inventory.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check-Avail.-of-Products</td>
<td>Retrieve-Ship-Products</td>
</tr>
</tbody>
</table>

**Event flow: EV-PRODUCTS-NOT-AVAIL.**

**Description:**
This event flow signals the respective transformation that the product demanded is not available in the warehouse's inventory.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive-Dispatch-Products</td>
<td>Inv.-Control-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Apply-WH-On-Demand-Policy</td>
</tr>
<tr>
<td>Check-WH-Inventory</td>
<td>*** Off the diagram ***</td>
</tr>
</tbody>
</table>

**Event flow: EV-SEQUENCED**

**Description:**
This flow represents a signal sent to the respective transformation(s) that the jobs have been sequenced.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence-Jobs</td>
<td>Group-and-Assign-Job-Numbers</td>
</tr>
</tbody>
</table>
**Event flow: EV-SHIPMENTS**

Description:

Signals the respective transformation(s) that shipments of products have occurred.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve-Ship-Products</td>
<td>Products-Inv.-Control</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Identify-PF-Policy</td>
</tr>
</tbody>
</table>

**Event flow: EV-TIME**

Description:

This is an event flow of the time which could be represented in terms of a date, week, day, hour, etc.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>*** Off the diagram ***</td>
<td>Inv.-Control-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Apply-Inventory-Policy</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Production-Plan.-Control</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Compare-T-with-Time</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Synchronize-Production</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Start-Pre-BN-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Apply-PF-Continuous-Policy</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Apply-PF-Periodic-Policy</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Apply-PF-Inventory-Policy</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Products-Inv.-Control</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Shipments-Inv-Control</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Orders-Inv.-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Raw-Materials-Control</td>
</tr>
</tbody>
</table>
Event flow: EV-TIMES-ARE-EQUAL

Description:

This flow indicates that the time at which the jobs have to be released for production (T) equals the current time.

Origin

Compare-T-with-Time

Destination

CONTROL-3.2.5.3-3.3

Data flow: FORECASTABLE-DEMAND

Description:

Represents the demand for products which can be forecasted.

Origin

originated as part of Market-Demand from the terminator Field-Or-Market

Destination

Forecast-Demand

Data flow: FORECASTED-DEMAND

Description:

Represents the forecasted demand for products.

Origin

Forecast-Demand

Destination

*** off the diagram ***

Implement-Forecasting-Model

*** off the diagram ***
*** off the diagram ***  
Orders-Inv-Operations  
*** off the diagram ***  
Identify-Ordered-Product  
Warehouses-Operations  
DPMF-Operations  

Composition:

= product-ID  
+ processing-times  
+ raw-materials-ID  
+ raw-materials-cost  
+ production-quantities

Data flow: FORMULATED-MATRIX

Description:

Represents the technology matrix formulated from FORMULATE-TECH.-MATRIX.

Origin  
Formulate-Tech.-Matrix  

Destination  
Synthesize-LP

Composition:

= product-ID  
+ processing-times  
+ raw-materials-ID  
+ raw-materials-cost  
+ production-quantities
**Data flow: FORMULATED-OBJECTIVE**

**Description:**

Represents the objective function of the linear program model formulated by FORMULATE-OBJECTIVE-FUNCTION.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formulate-Objective-Function</td>
<td>Synthesize-LP</td>
</tr>
</tbody>
</table>

**Composition:**

\[
\text{Composition:} \quad = \text{product-ID} \\
+ \text{production-cost} \\
+ \text{product-revenues} \\
+ [ \text{minimization-problem} \\
\quad : \text{maximization-problem} ]
\]

**Data flow: FORMULATED-RHS**

**Description:**

Represents the right-hand side vector of the linear program formulated by the process FORMULATE-RHS-VECTOR.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formulate-RHS-Vector</td>
<td>Synthesize-LP</td>
</tr>
</tbody>
</table>

**Composition:**

\[
\text{Composition:} \quad = \text{machine-constraints} \\
+ \text{demand-constraints} \\
+ \text{raw-materials-constraints} \\
+ \text{cash-flow-constraints}
\]

**Data flow: GROUPED-JOBS**

**Description:**

This data flow represents the orders that have been grouped.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group-and-Assign-Job-Numbers</td>
<td>Sequence-Jobs</td>
</tr>
</tbody>
</table>

**Composition:**

- job-number
- product-name
- product-ID
- production-quantity

**Data store: GROUPED-SEQUENCED-JOBS**

**Description:**

Contains the information regarding the grouped jobs that have been previously sequenced.

<table>
<thead>
<tr>
<th>Updated by</th>
<th>Referenced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group-and-Assign-Job-Numbers</td>
<td>Synchronize-Production</td>
</tr>
<tr>
<td>Modify-Files</td>
<td>Start-Pre-BN-Operations</td>
</tr>
</tbody>
</table>

Schedule-Bottleneck

Identify-BN.-Operations

Determine-T-to-Release-Jobs

Release-Jobs-for-Production

Classify-Sequencing-Problem

Formulate-Tech.-Matrix

Formulate-RHS-Vector
Initial-LP-Formulation

Composition:
= product-name
+ product-ID
+ job-number
+ sequence-number
+ route
+ production-quantity

Data store: HISTORICAL-DEMAND-DISPATCHES
Description:
Contains the historical data regarding the demand and products dispatches from the warehouses to the market.

Updated by
Update-Historical-Data

Referenced by
Analyze-Historical-Data

Composition:
= product-name
+ product-ID
+ historical-demand
+ historical-dispatches

Data store: JOBS-PROCESSING-TIMES
Description:
This data store contains the processing times of the products at each of the processing centers (machines).
Updated by

Determine-Mfg.-Processes

Referenced by

Schedule-Bottleneck

Determine-T-to-Release-Jobs

Classify-Sequencing-Problem

Sequence-Jobs

Synchronize-Production

Composition:

= product-name
+ product-ID
+ machine-center-ID
+ processing-time
+ setup-time

Data store: JOBS-QUEUE-I

Description:

Contains the upstream jobs that are waiting to be processed on the bottleneck.

Updated by

Release-Jobs-for-Production

Referenced by

Release-Jobs-from-I

Start-Pre-BN-Operations

Start-BN-Operation

Composition:

= product-name
+ product-ID
+ job-number
+ destination
Data store: JOBS-QUEUE-II

Description:
Contains the bottleneck's downstream jobs waiting to be processed on the machine centers following the bottleneck operation.

Updated by:
- Process-BN-Operation
- Start-BN-Operation

Referenced by:
- Release-Jobs-From-II
- Start-Post-BN-Operations

Composition:
= product-name
+ product-ID
+ job-number
+ destination

Data store: JOBS-ROUTINGS

Description:
Contains information regarding all the route(s) of each job.

Updated by:
- Determine-Mfg.-Processes

Referenced by:
- Sequence-Jobs
- Determine-T-to-Release-Jobs
- Classify-Sequencing-Problem
- Schedule-Bottleneck
- Synchronize-Production

Composition:
= product-name
Data flow: MARKET-DEMAND

Description:
The products demanded by the MARKET to satisfy the consumers' needs. MARKET-DEMAND is broken down into FORECASTABLE-DEMAND and UNFORECASTABLE-DEMAND as shown on data flow diagram 1 (WAREHOUSES-OPERATIONS).

Origin
*** Off the diagram ***
Demand-Source
*** Off the diagram ***

Composition: =
product-name
+ product-ID
+ quantity-demanded
+ date-demanded
+ date-wanted-by
+ source-ID-or-warehouse
+ products-description

Destination
Warehouses-Operations
IPPIC-Operations
Identify-Demanded-Product

Data store: MARKET-INVENTORY

Description:
Contains the data regarding the inventory of products
at the market.

Updated by
Update-Market-Inventory

Referenced by
Implement-Forecasting-Model

Composition:

= product-name
+ product-ID
+ inventory-level

Data store: MASTER-LP-FILE

Description:

Contains the master linear programming matrix formulated by INITIAL-LP-FORMULATION.

Updated by
Initial-LP-Formulation

Referenced by
Revise-LP
Solve-LP

Composition:

= LP-objective-function
+ LP-RHS-vector
+ LP-tech.-matrix

Data store: MASTER-LP-SOLUTION-FILE

Description:

Contains the master linear program solution created by SOLVE-LP.

Updated by
Test-Solution-Feasibility

Referenced by
Solve-LP
Solve-LP

Classify-Slack-Variables

Restore-Old-Solution

Determine-BN

Composition:

= objective-function-value

+ basic-variables

+ nonbasic-variables

+ values-of-variables

Data flow: MATERIALS

Description:

The raw materials shipment from the raw materials supplier in reply to an earlier order by the production facility.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>*** Off the diagram ***</td>
<td>Update-Inv-Orders-Status</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Raw-Materials-Control</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>DPMF-Operations</td>
</tr>
<tr>
<td>Raw-Materials-Supplier</td>
<td>IPPIC-Operations</td>
</tr>
</tbody>
</table>

Composition:

= raw-materials-name

+ raw-materials-ID

+ quantity-shipped

+ date-received

+ supplier-ID
Data flow: MATERIALS-ORDER

Description:
A list of the raw materials ordered by the production facility, which is sent to the raw materials supplier.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process-Materials-Order</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Raw-Materials-Control</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>DPMF-Operations</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>IPPIC-Operations</td>
<td>Raw-Materials-Supplier</td>
</tr>
</tbody>
</table>

Composition:

= raw-materials-name
+ raw-materials-ID
+ date-ordered
+ date-wanted-by
+ quantity-ordered
+ supplier-ID

Data store: MATERIALS-ORDERS-STATUS

Description:
This data store contains the information regarding the status of the raw materials orders.

Updated by
Update-Mtl-Order-Status
Update-Inv-Order-Status

Referenced by
Generate-Materials-Reports

Composition:

= order-number
Data flow: MATERIALS-REPORTS

Description:

This data flow represents the reports regarding the inventory levels of the raw materials at the production facility.

Origin

Generate-Materials-Reports
Process-Mtl-Order-Status
*** Off the diagram ***
Raw-Materials-Control

Destination

*** Off the diagram ***
Generate-DPMF-Reports
Orders-Inv.-Operations

Composition:

= materials-ID
+ materials-name
+ inventory-level
+ order-quantity
+ quantity-on-order
+ date-ordered
+ date-received
Data flow: NEEDED-PRODUCTS

Description:

This data flow contains the list of products that the warehouses need to either satisfy the demand for products, or maintain specific inventory levels.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply-Continuous-Policy</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Apply-Periodic-policy</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Prepare-and-Place-Orders</td>
</tr>
<tr>
<td>Apply-PF-Continuous-Policy</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Apply-PF-Periodic-Policy</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Apply-PF-Inventory-Policy</td>
<td>Process-Production-Orders</td>
</tr>
<tr>
<td>Apply-PF-On-Order-Policy</td>
<td>Process-Production-Orders</td>
</tr>
<tr>
<td>Apply-WH-Inventory-Policy</td>
<td>Process-Orders-for-Products</td>
</tr>
<tr>
<td>Apply-WH-On-Demand-Policy</td>
<td>Process-Orders-for-Products</td>
</tr>
</tbody>
</table>

Composition:

= product-name
+ product-ID
+ needed-quantity
+ date-wanted-by
+ warehouse-ID

Data store: NEW-PRODUCTS-DEMANDED

Description:

This data store contains the list of new products demanded. In other words, new products that have never
been ordered before. The list consists of the products' names, products' IDs, products' descriptions, and the quantity demanded.

**Updated by**
Identify-Demanded-Product

**Referenced by**
Update-Files
Apply-WH-On-Demand-Policy
Inv.-Control-Operations
Inv.-Orders-Operations
Update-Products-List-File

**Composition:**

= product-name
+ product-ID
+ product-description
+ quantity-demanded

**Data store: NEW-PRODUCTS-ORDERED**

**Description:**

This data store contains the information regarding the new products ordered. These are the products that have never been ordered before.

**Updated by**
Identify-Ordered-Product

**Referenced by**
Determine-Engr.-Specs.
Apply-PF-On-Order-Policy
Products-Inv.-Control
Update-PF-Products-List
Shipments-Inv-Control
Composition:

= product-name
+ product-ID
+ product-description
+ ordered-quantity
+ date-ordered
+ date-wanted

Data flow: NON-BASIC-VARIABLES

Description:

Represents the slack variables that do not exist in the basic column of the linear program solution matrix.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classify-Slack-Variables</td>
<td>Classify-as-a-BN</td>
</tr>
</tbody>
</table>

Composition:

= variables-ID
+ variables-value

Data flow: ONE-BN-FEED-ANOTHER

Description:

Indicates that the situation involves two bottleneck operations one directly following the other. In other words, one bottleneck feeds another.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classify-Sequencing-Problem</td>
<td>Sched-by-Min-2-M/C-</td>
</tr>
</tbody>
</table>
Data store: OPERATIONS-DATA-FILE

Description:
Contains information regarding the machine centers including processing times, setup times, capacity, power, etc.

Updated by: Modify-Files
Referenced by:
Formulate-RHS-Vector
Formulate-Tech.-Matrix

Composition:
= machine-center-name
+ machine-center-number
+ processing-times
+ setup-times
+ capacity
+ power
+ etc.

Data flow: PRODUCTION-REPORTS

Description:
PRODUCTION-REPORTS represent the production reports of the production facility. These consist of two main reports: (1) reports regarding the inventory levels of the finished products and (2) reports regarding the production flow, production lead times, work-in-process
inventory, delivery lead times, etc.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production-Operations</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Generate-DPMF-Reports</td>
</tr>
<tr>
<td>Production-Plan.-Control</td>
<td>Orders-Inv.-Operations</td>
</tr>
</tbody>
</table>

**Data flow: PRODUCTION-ORDERS**

**Description:**

PRODUCTION-ORDERS represents the production order issued within the production facility. This order is issued to satisfy some order from the warehouses and/or maintain specific levels of inventory.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>*** Off the diagram ***</td>
<td>Group-and-Assign-Job-Numbers</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Production-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Identify-Mtl.-Needed</td>
</tr>
<tr>
<td>Process-Production-Orders</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Products-Inv.-Control</td>
<td>*** Off the diagram ***</td>
</tr>
</tbody>
</table>

**Composition:**

= product-name
+ product-ID
+ quantity-to-be-produced
+ wanted-date
**Data flow: PRODUCTS-DISPATCHES**

Description:

The products shipped from the warehouses to the MARKET or FIELD in response to an earlier order.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive-Dispatch-Products</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Inv.-Orders-Operations</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Dispatch-Products</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Warehouses-Operations</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>IPPIC-Operations</td>
<td>Demand-Source</td>
</tr>
</tbody>
</table>

Composition:

= product-name
+ product-ID
+ date-dispatched
+ quantity-dispatched
+ destination-ID

**Data flow: PRODUCTS-ORDERED**

Description:

PRODUCTS-ORDERED contains a list of the products ordered which exist, but not necessarily in inventory.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>*** Off the diagram ***</td>
<td>Check-Avail.-of-Products</td>
</tr>
<tr>
<td>Identify-Ordered-Product</td>
<td>Shipments-Inv-Control</td>
</tr>
</tbody>
</table>

Composition:

= product-name
+ product-ID
+ quantity-orders
+ ordered-date
+ wanted-date
+ warehouse-ID

Data flow: PRODUCTS-ORDERS

Description:
The list of products ordered by the warehouses from the production facility. The products are ordered to satisfy some demand and/or maintain specific levels of inventory at the warehouses.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inv.-Control-Operations</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Inv.-Orders-Operations</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Identify-Ordered-Product</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Orders-Inv.-Operations</td>
</tr>
<tr>
<td>Prepare-and-Place-Orders</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Process-Orders-for-Products</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Warehouses-Operations</td>
<td>DPMF-Operations</td>
</tr>
</tbody>
</table>

Composition:
  = product-ID
  + quantity
  + warehouse-number
  + date-wanted-by
  + ( product-specifications )
Data flow: PRODUCTS-SHIPMENTS

Description:

The products shipped from the production facility to the warehouses in response to an earlier order.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orders-Inv.-Operations</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>DPMF-Operations</td>
<td>Warehouses-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Receive-Finished-Products</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Receive-Dispatch-Products</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Inv.-Orders-Operations</td>
</tr>
<tr>
<td>Retrieve-Ship-Products</td>
<td>*** Off the diagram ***</td>
</tr>
<tr>
<td>Shipments-Inv-Control</td>
<td>*** Off the diagram ***</td>
</tr>
</tbody>
</table>

Composition:

- product-name
- product-ID
- date-shipped
- quantity-shipped
- destination-ID

Data store: PRODUCTS-STANDARDS

Description:

Contains the specifications of each product manufactured at the production facility. Examples of such specifications are width, height, volume, color, etc. It also contains the type of material needed to manufacture each product.
Composition:

= product-ID
+ material-requirements
+ dimensions
+ color + special-features
+ etc.

**Data primitive: PTs-ARE-SIMILAR**

Description:

Indicates that the processing times of the jobs to be processed are similar to one another.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classify-Sequencing-Problem</td>
<td>Sched-By-Due-Date</td>
</tr>
</tbody>
</table>

**Data primitive: PTs-ARE-VERY-DIFFERENT**

Description:

Indicates that the processing times of the products to produced are very different from one another.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classify-Sequencing-Problem</td>
<td>Sched-by-Remaining-PT</td>
</tr>
</tbody>
</table>
**Data store: RAW-MATERIALS-INVENTORY**

**Description:**

This data store contains the information regarding the inventory levels of the different raw materials used at the production facility.

**Updated by**

- Update-Inv-Orders-Status
- Raw-Materials-Control

**Referenced by**

- Synchronize-Production
- Process-Mtl-Order-Status
- Generate-Materials-Reports
- Apply-Inventory-Policy
- Release-Jobs-for-Production
- Start-Pre-BN-Operations
- Production-Operations
- Check-Avail.-of-Mtl
- Production-Plan.-Control
- Raw-Materials-Control

**Composition:**

= materials-name
+ materials-ID
+ quantity-available

**Data flow: RECEIVED-ORDERS**

**Description:**

This data flow represents the orders that have been delivered by the production facility in response to earlier orders for products.
Data flow: RECEIVED-PRODUCTS

Description:

This data flow represents the products that the warehouses have received from the production facility in response to earlier orders for products.

Data flow: RELEASED-JOBS

Description:

This flow contains the data regarding the jobs released
for production on the machines.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release-Jobs-from-I</td>
<td>Process-BN-Operation</td>
</tr>
<tr>
<td>Release-Jobs-from-II</td>
<td>Process-Post-BN-Operation</td>
</tr>
</tbody>
</table>

Composition:
- job-number
- product-name
- product-ID
- production-quantity

**Data flow: REPORTS**

**Description:**

Represent the reports requested by the USER. These reports, generated from the warehouses and the production facility, allow the user to make decisions regarding the different standard policies being implemented. It provides the user with valuable information, which is used to check if changes in the system operations need to be made and/or necessary.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPPIC-Operations</td>
<td>User</td>
</tr>
</tbody>
</table>

Composition:
- warehouses-reports
- production-facility-reports
Data primitive: REPORTS-REQUEST

Description:

Is the request issued by the user for respective reports from both the warehouses and the production facility.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>*** Off the diagram ***</td>
<td>Generate-Reports</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Inv.-Orders-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Generate-Materials-Reports</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Process-Mtl-Order-Status</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Generate-BO-Status-Reports</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Production-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Process-as-Back-Orders</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Orders-Inv.-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Raw-Materials-Control</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Production-Plan.-Control</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Generate-DPMF-Reports</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>DPMF-Operations</td>
</tr>
<tr>
<td>*** Off the diagram ***</td>
<td>Warehouses-Operations</td>
</tr>
<tr>
<td>User</td>
<td>IPPIC-Operations</td>
</tr>
</tbody>
</table>

Data flow: RESTORED-SOLUTION

Description:

This flow contains the data regarding the restored linear programming solution.
Origin

Restore-Old-Solution

Composition:

= objective-function-value
+ basic-variables
+ nonbasic-variables
+ values-of-variables

Destination

Solve-Revised-LP

Data store: REVENUES-EXPENSES-FILE

Description:

Contains the data regarding the costs involved in producing the products and the revenues from selling them.

Updated by

Modify-Files

Referenced by

Formulate-RHS-Vector
Formulate-Objective-Function

Composition:

= product-name
+ product-ID
+ production-costs
+ revenues

Data flow: REVISED-LP

Description:

This flow contains the data regarding the revised linear programming model to be solved.
Origin: Revise-LP

Destination: Restore-Old-Solution

Composition:

\[ \text{revised-objective-function} + \text{revised-RHS-vector} + \text{revised-tech.-matrix} \]

Data flow: **SALES**

Description:

This flow contains the data regarding the actual sales for products at the market.

Origin: Market-or-Field

Destination: IPPIC-Operations

*** off the diagram ***

Destination: Warehouses-Operations

*** off the diagram ***

Destination: Forecast-Demand

*** off the diagram ***

Destination: Update-Market-Inventory

Composition:

\[ \text{product-name} + \text{product-ID} + \text{quantity-sold} \]

Data store: **SCHEDULING-SEQUENCING-POLICIES**

Description:

It contains the policies by which the production of products in the facility is to be scheduled. In other words, the production rules to be used for scheduling
the production of jobs. The production rule to be implemented is selected by the user.

Updated by
*** None ***

Referenced by
Sequence-Jobs
Production-Operations
Production-Plan.-Control
DPMF-Operations

Composition:
= scheduling-policy-name
+ scheduling-rule

Data flow: SELECTED-ORDERS

Description:
This data flow contains the list of back orders selected to be processed due to receipt of raw materials.

Origin
Process-File-Back-Orders
*** off the diagram ***

Destination
Production-Operations
Group-Assign-Job-Numbers

Composition:
= order-number
+ product-name
+ product-ID
+ date-ordered
+ date-wanted
Data flow: STANDARDS-POLICIES

Description:
These standards policies include the selection made by the USER regarding the warehouses inventory policies and the facility's products inventory policies, product standards, and production scheduling policies.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>IPPIC-Operations</td>
</tr>
</tbody>
</table>

Composition:
- = warehouses-inv-policies
+ facility-inv-policies
+ production-scheduling-policies
+ specifications-of-products

Data primitive: T

Description:
Contains the value of the time at which the jobs have to be released for production.

<table>
<thead>
<tr>
<th>Origin</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine-T-to-Release-Jobs</td>
<td>Compare-T-with-Time</td>
</tr>
</tbody>
</table>

Data flow: UNFORECASTABLE-DEMAND

Description:
Represents the demand for products which cannot be
forecasted.

Origin  Destination
originated as a part of
Market-Demand from the Identify-Demanded-Products
terminator Field-or-Market

**Data flow: VARS-GREATERTHAN-ZERO**

Description:

Indicates that the basic variable has a value greater than zero.

Origin  Destination
Test-Variables-Value  Classify-as-a-Non-BN

Composition:

= variables-ID

+ variables-value

**Data flow: VARS-EQUAL-ZERO**

Description:

Indicates that the basic variable has a value that is equal to zero.

Origin  Destination
Test-Variables-Value  Classify-as-a-BN

Composition:

= variables-ID

+ variables-value
Data store: WAREHOUSES-INVENTORY-POLICIES

Description:

This data store contains the several policies for controlling the inventory of products at the warehouses. Each policy consists of the different parameters needed to implement the policy. The policy to be implemented is selected by the user. The policies could be periodic, continuous, or both.

Updated by
User

Referenced by
Identify-WH-Inv-Policy
Identify-Inventory-Policy
Inv.-Control-Operations
Generate-Reports
Warehouses-Operations

Composition:

= inventory-policy-ID

+ policy-parameters for each product

Data store: WAREHOUSES-ORDERS-STATUS

Description:

This data store contains the information regarding the status of the orders sent out to the production facility. These orders could be to satisfy some market demand and/or maintain specific levels of inventory.

Updated by
Inv.-Control-Operations

Referenced by
Generate-Reports
Receive-Dispatch-Products

File-Orders

Process-Orders-for-Products

Update-WH-Orders-Status

Composition:

= order-number
+ date-issued
+ date-products-wanted
+ date-products-received

**Data store: WAREHOUSES-PRODUCTS-FILES**

Description:

This data store contains the list of products which exist at the warehouses but not necessarily in inventory. This list contains the products' name, products' IDs, and products' descriptions.

Updated by

Identify-Products-Description
Update-Products-List-File
Update-Files

Referenced by

Identify-Demanded-Product

Composition:

= product-name
+ product-ID
+ product-description
**Data store: WAREHOUSES-PRODUCTS-INVENTORY**

**Description:**

This data store contains the inventory level of each product at the warehouses.

**Updated by**

- Receive-Dispatch-Products
- Check-WH-Inventory
- Update-WH-Inventory

**Referenced by**

- Receive-Dispatch-Products
- Apply-WH-Continuous-Policy
- Apply-Periodic-policy
- Inv.-Control-Operations
- Generate-Reports
- Apply-WH-Inventory-Policy
- Check-WH-Inventory

**Composition:**

= products-name
+ products-ID
+ product-quantity-available
+ warehouse-ID

**Data flow: WAREHOUSES-REPORTS**

**Description:** These reports are generated so the user can make decisions regarding the policies being implemented at the warehouses.

**Origin**

- Generate-Reports
- Inv.-Orders-Operations
- Warehouses-Operations

**Destination**

*** Off the diagram ***
Component of:

reports

Composition:

= inventory-reports
+ orders-status-reports
Transformation primitive: ANALYZE-HISTORICAL-DATA

Process number: 1.4.2

Description:
Performs an analysis of the historical demand for products.

Input/Output Information:

Data Entering
EV-time

Data Leaving
EV-data-analyzed

Structured English:
On a timely basis DO
Access HISTORICAL-DISPATCHES-DATA and analyze the retrieved data

Parent:
Forecast-Demand

Transformation primitive: APPLY-PF-CONTINUOUS-POLICY

Process number: 2.1.3.5.2.2

Description:
Applies the continuous inventory policy to control the inventory of products at the production facility. This transformation is enabled by the control process if the policy selected for controlling the inventory was the continuous policy.
Structured English:

FOR (each occurrence of PRODUCTS-SHIPMENTS) DO

Access FINISHED-PRODUCTS-INVENTORY

Parent:

Apply-PF-Inventory-Policy

Process: APPLY-PF-INVENTORY-POLICY

Process number: 2.1.3.5.3

Description:

This transformation is enabled by the CONTROL process 2.1.3.5.1-2.1.3.5.2 if the policy selected by the user was the periodic or continuous policies. It applies the policy selected to control the inventory of products at the production facility on a continuous basis. It is also applied if a PRODUCTS-SHIPMENT occurred.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>products-shipments</td>
<td>needed-products</td>
</tr>
<tr>
<td>EV-ENAB.-DISAB.</td>
<td></td>
</tr>
<tr>
<td>EV-time</td>
<td></td>
</tr>
</tbody>
</table>
Parent:
Products-Inv.-Control

Child processes:
CONTROL-......1-......2
Identify-PF-Inv-Policy
Apply-PF-Continuous-Policy
Apply-PF-Periodic-Policy

Process: APPLY-INVENTORY-POLICY
Process number: 2.3.4
Description:
This process applies either the periodic inventory policy or the continuous inventory policy to control the inventory of raw materials at the production facility. When triggered with EV-OTHER, it accesses the RAW-MATERIALS-INVENTORY data store, applies the policy, and generates MATERIALS-NEEDED if necessary.

Input/Output Information:

Data Entering Data Leaving
EV-other materials-needed
EV-Time

Parent:
Raw-Materials-Control
Process: APPLY-ON-DEMAND-POLICY

Process number: 2.3.5

Description:

This process orders raw materials as the signal EV-ON-DEMAND is received. For this process, MATERIALS-NEEDED is generated as the materials are needed because the on-demand policy assumes zero inventory.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-On-Demand</td>
<td>materials-needed</td>
</tr>
</tbody>
</table>

Parent:

Raw-Materials-Control

Transformation primitive: APPLY-PF-ON-ORDER-POLICY

(PF = Production Facility)

Process number: 2.1.3.5.2

Description:

This transformation is enabled to apply the on-demand inventory policy, by the control process 2.1.3.5.1-2.1.3.5.2, if the policy to be implemented was the On-Demand policy. If BACK-ORDERS and/or NEW-PRODUCTS-ORDERED occurred, then this policy is used. This policy assumes zero inventory for the products.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-ENAB.-DISAB.</td>
<td>needed-products</td>
</tr>
</tbody>
</table>
back-orders

Structured English:

IF (the On-Order policy is used) AND
(a BACK-ORDER occurred) AND/OR (NEW-PRODUCTS-ORDER occurred) THEN
FOR (each product) DO
   NEEDED QUANTITY(product) = back-order quantity
   (if back-order)
OR
   NEEDED QUANTITY(product) = ordered quantity
   (if new product)

Parent:
Products-Inv.-Control

Transformation primitive: APPLY-PF-PERIODIC-POLICY

Process number: 2.1.3.5.2.1

Description:

This transformation applies the periodic inventory policy for controlling the inventory of the products at the production facility. It is enabled by the control process if the policy selected was the periodic policy.

Input/Output Information:

Data Entering                      Data Leaving
EV-ENAB.-DISAB.                     needed-products
EV-time
Structured English:

FOR (each occurrence of EV-ENAB.-DISAB.) DO
  FOR (every period signaled by EV-TIME) DO
    enable APPLY-PERIOD.-POLICY

Parent:

Apply-PF-Inventory-Policy

Transformation primitive:  APPLY-WH-CONTINUOUS-POLICY

(WH = Warehouses)

Process number: 1.3.2.3.3

Description:

This transformation applies the continuous inventory policy for controlling the inventory of products at the warehouses. Under this policy the inventory is continuously monitored. It authorizes the ordering of products by sending the data flow NEEDED-PRODUCTS to the transformation PROCESS-ORDERS-FOR-PRODUCTS.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>dispatch-info.</td>
<td>needed-products</td>
</tr>
<tr>
<td>EV-ENAB.-DISAB.</td>
<td></td>
</tr>
</tbody>
</table>

Structured English:

On a continuous basis and at the occurrence DISPATCHED-PRODUCTS DO

BEGIN

Access PRODUCTS-INVENTORY file
FOR (each product) DO

IF (amount in inventory)<(a specific level) THEN

set quantity needed = (maximum inv.) - (current inv. level)

ELSE

disable the process

END

Parent:

Apply-WH-Inventory-Policy

Process: APPLY-WH-INVENTORY-POLICY

Process number: 1.3.2.3

Description:

This transformation applies one of the inventory policies, selected by the user, to control the inventory of the products. This policy could be either periodic or continuous, depending on the selection made. To apply the policy, it accesses the PRODUCTS-INVENTORY file and decides which products need to be ordered.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-ENAB.-DISAB.</td>
<td>needed-products</td>
</tr>
<tr>
<td>dispatch-info.</td>
<td></td>
</tr>
<tr>
<td>EV-time</td>
<td></td>
</tr>
</tbody>
</table>
Parent:
Inv.-Control-Operations

Child processes:
Apply-WH-Continuous-Policy
Apply-WH-Periodic-Policy
Identify-WH-Inv-Policy
CONTROL-2.3.2.2.3

Transformation primitive: APPLY-WH-ON-DEMAND-POLICY

Process number: 1.3.2.2

Description:
This transformation applies the on-demand inventory policy to control the inventory of products at the warehouses. Basically, once a demand for products is received, it issues an order for those products. This policy maintains zero inventory of the products.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>demanded-products</td>
<td>needed-products</td>
</tr>
<tr>
<td>EV-ENAB.-DISAB.</td>
<td></td>
</tr>
<tr>
<td>EV-product-not-avail.</td>
<td></td>
</tr>
</tbody>
</table>

Structured English:
FOR (each occurrence of DEMANDED-PRODUCTS, OR (EV-PRODUC-NOT-AVAIL., where AVAIL. stands for available), OR
IF (policy used is On-Demand policy) DO
apply the On-Demand policy and order the demanded products

Parent:
Inv.-Control-Operations

Transformation primitive: APPLY-WH-PERIODIC-POLICY

Process number: 1.3.2.3.2

Description:
This transformation applies the periodic inventory policy for controlling the inventory of products. This policy is applied at the end of each review period when triggered by the EV-TIME event flow and enabled by the CONTROL process.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-ENAB.-DISAB.</td>
<td>needed-products</td>
</tr>
<tr>
<td>EV-Time</td>
<td></td>
</tr>
</tbody>
</table>

Structured English:
FOR (each period, when enabled) DO
BEGIN
Access PRODUCTS-INVENTORY file
FOR (each product) DO
IF (amount in inventory is less than a specific level) THEN
IF (inventory > reorder point) THEN
Set quantity needed = 0
ELSE

    set quantity needed = (maximum inv.) - (current inv. level)

END

Parent:

Apply-WH-Inventory-Policy

Transformation primitive: CHECK-AVAIL.-OF-MTL

Process number: 2.2.2

Description:

    Accesses the data store RAW-MATERIALS-INVENTORY and checks if the raw materials needed to manufacture the products are available in inventory.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>mtl-needed</td>
<td>EV-Mtl-not-available</td>
</tr>
<tr>
<td></td>
<td>EV-Mtl-available</td>
</tr>
</tbody>
</table>

Structured English:

FOR (each occurrence of MTL-NEEDED) DO

    Access RAW-MATERIALS-INVENTORY data store

    IF (material is available) THEN
        Schedule the production of the product(s)
    ELSE
        File the order as a back-order and order the raw materials
Parent:
Production-Plan.-Control

Transformation primitive: CHECK-AVAIL.-OF-PRODUCTS
Process number: 2.1.3.1

Description:
Checks if the products ordered are available in inventory.

Input/Output Information:

Data Entering          Data Leaving
products-ordered        EV-products-avail.
                        EV-products-not-avail.

Structured English:
FOR (each occurrence of PRODUCTS-ORDERED) DO
    Access FINISHED-PRODUCTS-INVENTORY data store
    FOR (each product ordered) DO
        IF (product is not available in inventory) THEN
            process the order as a back-order
        ELSE
            Retrieve product from inventory and ship it.

Parent:
Shipments-Inv.-Control

Transformation primitive: CHECK-WH-INVENTORY
Process number: 1.3.1.1
Description:

This transformation checks if the DEMANDED-PRODUCTS are available in inventory. If they are available, it sends a massage (EV-PRODUCT-AVAILABLE) to the transformation DISPATCH-PRODUCTS. Otherwise, it sends a massage (EV-PRODUCT-NOT-AVAIL.) indicating that the product is not available in inventory to the respective transformation(s).

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-product-available</td>
<td></td>
</tr>
</tbody>
</table>

Structured English:

FOR (each occurrence of DEMANDED-PRODUCTS) DO

BEGIN

Access file PRODUCTS-INVENTORY and check inventory for these products,

IF (the product is available in inventory) THEN

send a signal to dispatch the products,

ELSE

send a signal indicating that the product is not available

END

Parent:

Receive-Dispatch-Products
Process number: 2.2.4.3.1.3.3
Description:
Classifies the machine centers corresponding to the nonbasic variables as bottleneck machines.
Input/Output Information:

Data Entering
vars.-equal-zero
non-basic-variables

Data Leaving

Parent:
Determine-BN

Process number: 2.2.4.3.1.3.4
Description:
This transformation classifies the machines corresponding to the basic variables, with a value greater than zero, as non-bottleneck machines.
Input/Output Information:

Data Entering
vars.-greater-than-zero

Data Leaving

Parent:
Determine-BN

Transformation Primitive: CLASSIFY-SEQUENCING-PROBLEM
Process number: 2.2.4.3.2.1
Description:

This process classifies the problem of sequencing the jobs on the machine centers. In other words, it determines what type of sequencing problem is involved.

Input/Output Information

**Data Entering**  
EV-BN-identified

**Data Leaving**  
BN-involves-setup  
BN-makes-more-than-1-part  
one-BN-feed-another  
PTs-are-similar  
PTs-are-very-different

Parent:

Schedule-Bottleneck

**Transformation Primitive:** COMPARE-T-WITH-TIME

Process number: 2.2.4.3.3.2

Description:

This transformation compares the value of the time at which the jobs have to be released for production (T) with the current time.

Input/Output Information:

**Data Entering**  
EV-time  
T

**Data Leaving**  
EV-times-are-equal
Structured English:

FOR (each occurrence of T) DO

Compare the value of the time at which the jobs have to released for production (T) with the current time (indicted by EV-TIME).

IF ( T = TIME) THEN

send the message EV-TIMES-ARE-EQUAL

Parent:

Start-Pre-BN-Operations

Process: DETERMINE-BN

Process number: 2.2.4.3.1.3

Description:

This transformation determines which of the operations are the bottleneck operations. As shown on its child diagram, once the signal EV-LP-SOLVED is received, the data store MASTER-LP-SOLUTION-FILE is accessed. The first process involves classifying the slack variables to either basic or non-basic variables. The classification of the slack variables and their values determine which of the operations are the bottlenecks.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-LP-solved</td>
<td>EV-BN-identified</td>
</tr>
</tbody>
</table>

Parent:

Identify-BN.-Operations
Child processes:

Classify-as-a-Non-BN
Classify-as-a-BN
Test-Variables-Value
Classify-Slack-Variables

Transformation primitive: DETERMINE-ENGR.-SPECS.
Process number: 2.1.2.2

Description:
This transformation determines the engineering specifications of the new products (PRODUCTS) ordered such as dimensions, raw materials, color, etc.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-New-product</td>
<td>engr.-specs.</td>
</tr>
</tbody>
</table>

Structured English:

FOR (each signal EV-NEW-PRODUCT received) DO,
Access file NEW-PRODUCTS-ORDERED
FOR (all new products) DO
BEGIN
Determine the engineering specifications,
Update the BILL-OF-MATERIALS data store.
END

Parent:

Update-DPMF-Files
Transformation primitive: DETERMINE-MFG.-PROCESSES

Process number: 2.1.2.3

Description:
This transformation determines the manufacturing processes required to manufacture the new products. This would include the jobs' routes and processing times.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>engr.-specs.</td>
<td></td>
</tr>
</tbody>
</table>

Structured English:

FOR (each occurrence of EV-NEW-PRODUCT) DO
Access file NEW-PRODUCTS-ORDERED
FOR (all new products) DO
BEGIN
Determine the manufacturing processes required to produce the new products,
Update the respective files (JOBS-PROCESSING-TIMES, JOBS-ROUTE, and PRODUCTS-STANDARDS).

Parent:
Update-DPMF-Files

Transformation primitive: DETERMINE-T-TO-RELEASE-JOBS

Process number: 2.2.4.3.3.1

Description:
This transformation determines the time (T) to release the jobs to be processed on the machines. To obtain
the information needed to determine \( T \), it accesses the
data stores GROUPED-SEQUENCED-JOBS, JOBS-ROUTINGS, and
JOBS-PROCESSING-TIMES.

**Input/Output Information:**

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>BN-schedule</td>
<td>( T )</td>
</tr>
</tbody>
</table>

**Structured English:**

FOR (each occurrence of BN-SCHEDULE) DO

BEGIN

Access the data stores JOBS-PROCESSING-
TIMES and JOBS-ROUTINGS,

Calculate the time \( (T) \) to release the jobs for pro-
duction

END

Parent:
Start-Pre-BN-Operations

**Transformation primitive:** DISPATCH-PRODUCTS

Process number: 1.3.1.2

**Description:**

This transformation is in charge of dispatching the
products from the warehouses to the market. Once it
receives the signal EV-PRODUCT-AVAILABLE, it dis-
patches the respective products. The information
regarding the dispatched products (DISPATCH-INFO.) is
sent to the transformation UPDATE-INVENTORY.
Structured English:

FOR (each signal received (EV-PRODUCT-AVAILABLE)) DO
    process the dispatching of the available products and
    supply the dispatch information to the process that
    updates the inventory.

Parent:

Receive-Dispatch-Products

Process: DPMF-OPERATIONS

Process number: 2

Description:

This transformation represents the operations of the
production facility. These operations include raw
material inventory control, finished products inven-
tory control, production scheduling and job sequen-
cing. The decomposition of this transformation is
shown on diagram 2.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-time</td>
<td>products-shipments</td>
</tr>
<tr>
<td>materials</td>
<td>DPMF-reports</td>
</tr>
<tr>
<td>products-orders</td>
<td>materials-order</td>
</tr>
</tbody>
</table>
reports-request
forecasted-demand

Parent:
PPPIC-Operations

Child processes:
Orders-Inv-Operations
Production-Plan.-Control
Raw-Materials-Control

Transformation primitive: FORMULATE-OBJECTIVE-FUNCTION

Process number: 2.2.4.3.1.1.2

Description:
Formulates the objective function of the linear program. It accesses file REVENUES-EXPENSES-FILE to obtain the necessary data for formulating the objective function.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-jobs-grouped</td>
<td>formulated-objective</td>
</tr>
</tbody>
</table>

Structured English:

FOR (each occurrence of EV-JOBS-GROUPED) DO

BEGIN

Access REVENUES-EXPENSES-FILE,

Formulate the objective function (min. or max.)

END
Parent:

Initial-LP-Formulation

Transformation primitive: FORMULATE-RHS-VECTOR

Process number: 2.2.4.3.1.1.3

Description:

It accesses REVENUES-EXPENSES-FILE to obtain the cash flow data, BILL-OF-MATERIALS for the materials constraints, GROUPED-SEQUENCED-JOBS for the quantities to be produced of each product, and OPERATIONS-DATA-FILE for the machining centers constraints.

Input/Output Information:

Data Entering Data Leaving
EV-jobs-grouped formulated-RHS

Structured English:

FOR (each occurrence of EV-JOBS-GROUPED) DO

BEGIN


Formulate the right-hand side vector

END

Parent:

Initial-LP-Formulation
Transformation primitive: FORMULATE-TECH.-MATRIX

Process number: 2.2.4.3.1.1.4

Description:

It accesses the data stores GROUPED-SEQUENCED-JOBS for the quantities to be produced of each product and OPERATIONS-DATA-FILE for the processing and setup times on the machining centers for the different products. Once it retrieves this data, itformulates the technology matrix of the linear program model.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-jobs-grouped</td>
<td>formulated-matrix</td>
</tr>
</tbody>
</table>

Structured English:

FOR (each occurrence of EV-JOBS-GROUPED) DO

BEGIN

Access GROUPED-SEQUENCED-JOBS and OPERATIONS-DATA-FILE data stores,

Formulate the technology matrix

END

Parent:

Initial-LP-Formulation

Process: GENERATE-DPMF-REPORTS

Process number: 2.1.4

Description:

This process generates the reports required from the
production facility. These reports include production reports, orders reports, and inventory reports.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>materials-reports</td>
<td>DPMF-reports</td>
</tr>
<tr>
<td>production-reports</td>
<td></td>
</tr>
<tr>
<td>reports-request</td>
<td></td>
</tr>
</tbody>
</table>

Parent:

Orders-Inv-Operations

Process: GENERATE-MATERIALS-REPORTS

Process number: 2.3.2.

Description:

This transformation generates the reports requested by the user regarding the inventory levels of the raw materials, the status of orders, and the inventory policy being implemented for controlling the inventory of raw materials.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>reports-request</td>
<td>materials-reports</td>
</tr>
</tbody>
</table>

Parent:

Raw-Materials-Control

Transformation primitive: GENERATE-WH-REPORTS

Process number: 1.3.3
Description:

This transformation generates the reports requested by the user regarding the inventory levels of the products, the status of orders, and the inventory policy being implemented at the warehouses.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>reports-request</td>
<td>warehouses-reports</td>
</tr>
</tbody>
</table>

Structured English:

For each occurrence of REPORTS-REQUEST,

Access PRODUCTS-INVENTORY, ORDERS-STATUS, and INVENTORY-POLICIES data stores. Then, generate the warehouses reports from the retrieved information.

Parent:

Inv-Orders-Operations

Transformation primitive: GROUP-AND-ASSIGN-JOB NUMBERS

Process number: 2.2.4.1

Description:

This transformation groups the jobs depending on some specific criteria. Jobs are usually grouped according to shape, size, color, etc. First, it accesses the data store FACILITY-OUTSTANDING-ORDERS to obtain the information regarding the orders that need to be produced for that period. It then accesses the data stores SCHEDULING-POLICIES, and PRODUCTS-STANDARDS to retrieve
the data necessary for the process of grouping the jobs. Once the orders (jobs) are grouped, each is given a group number. Finally, the transformation stores the information about the grouped jobs in the data store GROUPED-JOBS and sends it to SEQUENCE-JOBS data transformation. After the jobs have been sequenced, the transformation retrieves this data from SEQUENCED-JOBS to check if a regrouping of the jobs is necessary, or if a better grouping is possible.

Input/Output Information:

Data Entering
EV-sequenced
production-orders
selected-orders

Data Leaving
EV-jobs-grouped
grouped-jobs

Structured English:

FOR (each occurrence of PROD.-ORDERS) DO
BEGIN
Access the data stores SCHEDULING-POLICIES,
PRODUCTS-STANDARDS, FACILITY-OUTSTANDING-ORDERS,
Group the outstanding orders into jobs, Assign numbers to the groups of jobs END FOR (each occurrence of EV-SEQUENCED) DO BEGIN Access the data stores SCHEDULING-POLICIES, PRODUCTS-STANDARDS, FACILITY-OUTSTANDING-ORDERS, Check if regrouping is necessary, or if better grouping is possible, IF (regrouping occurred) THEN Assign the new numbers
to the groups of jobs ELSE Keep the old numbers END

Parent:

Production-Operations

Process: IDENTIFY-BN.-OPERATION

Process number: 2.2.4.3.1

Description:

This transformation identifies the bottleneck operation(s) that may exist for that period. Once the bottlenecks are identified, the respective information is stored in BOTTLENECK-OPERATION-FILE, and a signal (EV-BN-IDENTIFIED) is sent to the transformation SCHEDULE-BOTTLENECK. The different processes involved in identifying the bottleneck operations are shown on its child diagram. The first process involves the formulation of a linear programming model (LP). The LP objective and constraints formulation will depend on the specific firm and situation. The second process solves the LP formulated. Finally, DETERMINE-BN determines which of the operations are the bottlenecks from the solution (as shown on DETERMINE-BN child diagram).

Input/Output Information:

Data Entering              Data Leaving
EV-jobs-grouped            EV-BN-identified

Parent:

Synchronize-Production
Child processes:

Determine-BN
Solve-LP
Initial-LP-Formulation

Transformation primitive: IDENTIFY-DEMANDED-PRODUCT

Process number: 1.1

Description:

This transformation is enabled by the occurrence of a UNFORECASTABLE-DEMAND. Once it receives this data, it checks to see if there are any new products ordered among the demanded products. To determine this, it accesses WAREHOUSES-PRODUCTS-FILES and compares the list of products ordered (DEMAND) with the list of products already existing in the warehouses files. If there are new products, it stores them in the NEW-PRODUCTS-DEMANDED data store.

Input/Output Information:

Data Entering

Data Leaving

demand

demanded-products

EV-New-product

Structured English:

FOR (each occurrence of UNFORECASTABLE-DEMAND) DO

BEGIN

Access file WAREHOUSES-PRODUCTS-FILES, and identify the demanded products,
IF (the demand contains any new products) THEN

Store the information about the new products in
NEW-PRODUCTS-DEMANDED data store AND Send a
signal (EV-NEW-PRODUCTS) to the transformation
UPDATE-FILES,

ELSE

Send the DEMANDED-PRODUCTS to the transformation
INV.-ORDERS-OPERATIONS.

Parent:

Warehouses-Operations

Transformation primitive: IDENTIFY-MTL.-NEEDED

Process number: 2.2.1

Description:

Upon the receipt of a production order (PROD.-ORDER),
this transformation access the data store BILL-OF-
MATERIALS to identify the raw materials needed to manu-
facture the respective products.

Input/Output Information:

Data Entering Data Leaving
production-orders mtl-needed

Structured English:

FOR (each occurrence of PROD.-ORDER (production-order))

DO

Access BILL-OF-MATERIALS data store

FOR (each product) DO
Identify the raw materials needed to manufacture the product

Parent:
Production-Plan.-Control

Process: IDENTIFY-ORDERED-PRODUCT
Process number: 2.1.1
Description:
This process identifies whether the ordered product is a new product or not. If new products have been ordered, which have not been manufactured at the production facility before, it stores this information in NEW-PRODUCTS-ORDERED.

Input/Output Information:

Data Entering Data Leaving
products-orders products-ordered
EV-New-product

Parent:
Orders-Inv-Operations

Transformation primitive: IDENTIFY-PF-POLICY
Process number: 2.1.3.5.1
Description:
Identifies the inventory policy (or policies) used to control the inventory of products at the production facility. Again, the policies to implemented are
selected by the user.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-B.O.</td>
<td>EV-other</td>
</tr>
<tr>
<td>EV-shipments</td>
<td>EV-On-Demand</td>
</tr>
</tbody>
</table>

Structured English:

FOR (each occurrence of EV-SHIPMENTS) OR
  (each occurrence of EV-B.O.) DO

Access the data store FACILITY-INVENTORY-POLICIES and
identify the inventory policy to be implemented

IF (policy is the On-Demand policy) THEN

Signal the control process to enable APPLY-ON-ORDER-POLICY

ELSE

Signal the control process to enable APPLY-INV.-POLICY

Parent:

Products-Inv.-Control

Process: IDENTIFY-PF-INV-POLICY

Process number: 2.1.3.5.3.1

Description:

This process identifies what inventory policy is to be used for controlling the inventory of products at the production facility.
Input/Output Information:

Data Entering  
EV-ENAB.-DISAB.

Data Leaving  
EV-periodic  
EV-continuous

Parent:
Apply-PF-Inventory-Policy

Process: IDENTIFY-RAW-MTL-POLICY
Process number: 2.3.3
Description:
This process identifies what inventory policy is used for controlling the inventory of raw materials at the production facility.

Input/Output Information:

Data Entering  
EV-production-completed  
EV-Mtl-not-available

Data Leaving  
EV-other  
EV-On-Demand

Parent:
Raw-Materials-Control

Process: IDENTIFY-WH-POLICY
Process number: 1.3.2.1
Description:
This transformation identifies which inventory policy is to be implemented for controlling the inventory of products at the warehouses. It accesses the data
store WAREHOUSES-INVENTORY-POLICIES to obtain the information needed. Once the policy is identified, it sends the respective massage to the CONTROL process 2.3.2.2-2.3.2.3.

Input/Output Information:

Data Entering:  
demanded-products  
dispatch-info.

Data Leaving:  
EV-other  
EV-On-Demand

Parent:  
Inv.-Control-Operations

Transformation primitive: IDENTIFY-WH-INV-POLICY

Process number: 1.3.2.3.1

Input/Output Information:

Data Entering:  
EV-ENAB.-DISAB.

Data Leaving:  
EV-continuous  
EV-periodic

Structured English:

FOR (each occurrence of EV-ENAB.-DISAB.) DO

Access the data store WAREHOUSES-INVENTORY-POLICIES

IF (the policy to be implemented is the periodic inventory policy) THEN

Signal the control process to enable APPLY-

PERIODIC-POLICY

ELSE

Signal the control process to enable APPLY-
Parent:

Apply-WH-Inventory-Policy

Transformation primitive: IMPLEMENT-FORECASTING-MODEL

Process number: 1.4.1

Description:

This process implements the forecasting model selected by the user to predict the demand for products.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-data-analyzed</td>
<td>forecasted-demand</td>
</tr>
</tbody>
</table>

Structured English:

FOR (each occurrence of EV-data-analyzed) DO

Access the data stores WAREHOUSES-INVENTORY-POLICIES, MARKET-INVENTORY, and DISPATCHES-DATA-BASE

Implement the forecasting model to generate the forecasted demand.

Parent:

Forecast-Demand

Process: INITIAL-LP-FORMULATION

Process number: 2.2.4.3.1.1

Description:

Represents the different processes involved in formulating the linear program for identifying the bottle-
necks. First, it checks for any changes desired by the user to any of the files accessed. The second process formulates the objective function. The second process formulates the right-hand side vector. Then, the technology matrix is formulated. Finally, the last process synthesizes the formulated matrices to formulate the linear program model.

Input/Output Information:

**Data Entering**
- EV-jobs-grouped

**Data Leaving**
- EV-LP-formulated

Parent:
- Identify-BN.-Operations

Child processes:
- Synthesize-LP
- Formulate-Tech.-Matrix
- Formulate-Objective-Function
- Formulate-RHS-Vector
- Modify-Files

Process: **INV.-CONTROL-OPERATIONS**

Process number: 1.3.2

Description:

This transformation represents the several operations performed to control the inventory levels at the warehouses. The decomposition of this transformation is shown in diagram 1.3.2.
Input/Output Information:

**Data Entering**
- dispatch-info.
- EV-time
- demanded-products

**Data Leaving**
- products-orders

**Parent:**
Inv-Orders-Operations

**Child processes:**
- Identify-WH-Policy
- Apply-WH-Inventory-Policy
- Process-Orders-for-Products
- CONTROL-2.3.2.2-2.3.2.3
- Apply-WH-On-Demand-Policy

**Process:** INV.-ORDERS-OPERATIONS

**Process number:** 1.3

**Description:**

This transformation represents the operations of receiving and dispatching products from the production facility and to the market, controlling the inventory of the products, and generating the reports requested by the user. The decomposition of this transformation is shown in diagram 1.3.

Input/Output Information:
**Data Entering**
- products-shipments
- reports-request
- demanded-products
- EV-Time

**Parent:**
- Warehouses-Operations

**Child processes:**
- Generate-WH-Reports
- Inv.-Control-Operations
- Receive-Dispatch-Products

**Process:** IPPIC-OPERATIONS

**Description:**
IPPIC-Operations is "the system". Diagram 1 shows the system, the three terminators, and the external events. Decomposing IPPIC-Operations yields the two main operations of the system as shown in diagram 2.

**Input/Output Information:**

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>demand</td>
<td>materials-order</td>
</tr>
<tr>
<td>EV-time</td>
<td>reports</td>
</tr>
<tr>
<td>desired-changes</td>
<td>products-dispatches</td>
</tr>
<tr>
<td>standards-policies</td>
<td></td>
</tr>
<tr>
<td>materials</td>
<td></td>
</tr>
<tr>
<td>reports-request</td>
<td>warehouses-reports</td>
</tr>
</tbody>
</table>
Parent:

CONTEXT

Child processes:

DPMF-Operations

Warehouses-Operations

Transformation primitive: MODIFY-FILES

Process number: 2.2.4.3.1.1.1

Description:

Modifies the respective files if there are any changes issued by the user. Once the data DESIRED-CHANGES are received, it updates the respective files for these changes.

Input/Output Information:

Data Entering

desired-changes

Data Leaving

Structured English:

FOR (each occurrence of DESIRED-CHANGES) DO

Update the data stores REVENUES-EXPENSES-FILE, BILL-OF-MATERIALS, GROUPED-SEQUENCED-JOBS, and OPERATIONS-DATA-FILE.

Parent:

Initial-LP-Formulation

Process: ORDERS-INV.-OPERATIONS

Process number: 2.1
Description:

This transformation represents the different operation that involve the inventory control of the products, the shipments of products to the warehouses in response to some order, updating products files, and generating the reports requested by the user.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>reports-request</td>
<td>products-shipments</td>
</tr>
<tr>
<td>EV-production-completed</td>
<td>DPMF-reports</td>
</tr>
<tr>
<td>EV-time</td>
<td>production-orders</td>
</tr>
<tr>
<td>production-reports</td>
<td>new-standards</td>
</tr>
<tr>
<td>products-orders</td>
<td></td>
</tr>
<tr>
<td>materials-reports</td>
<td></td>
</tr>
</tbody>
</table>

Parent:

DPMF-Operations

Child processes:

Generate-DPMF-Reports
Shipments-Inv-Control
Update-DPMF--Files
Identify-Ordered-Product

Transformation primitive: PROCESS-AS-BACK-ORDERS

Process number: 2.1.3.4

Description:

If the products ordered were not available in inven-
tory, this transformation receives the signal EV-PRODUCTS-NOT-AVAILABLE. Upon receipt of this signal, it tags those unsatisfied orders as BACK-ORDERS.

Input/Output Information:

Data Entering: EV-products-not-avail.  
Data Leaving: EV-B.O. back-orders

Structured English:

FOR (each occurrence of EV-PRODUCTS-NOT-AVAIL.) DO

Tag order as a back-order.

Parent:
Shipments-Inv-Control

Transformation primitive: PROCESS-BN-OPERATION

Process number: 2.2.4.3.4.2

Description:

Once this transformation is enabled by the CONTROL process, it starts processing the jobs received on the bottleneck operation. The finished jobs are stored in the data store JOBS-QUEUE-II, which is the downstream inventory of the bottleneck. For every job finished, it sends a signal (EV-BN-COMPLETED) to the respective transformation(s).

Input/Output Information:
Structured English:

FOR (each occurrence of JOBS and EV-ENAB.) DO

BEGIN

Process the jobs on the bottleneck,

Put the jobs finished in JOBS-QUEUE-II

END

Parent:

Start-BN-Operation

Process: PROCESS-MATERIALS-ORDER

Process number: 2.3.6

Description:

This process generates the orders of the raw materials needed.

Input/Output Information:

Data Entering Data Leaving
materials-needed materials-order

Parent:

Raw-Materials-Control

Transformation primitive: PROCESS-ORDERS-FOR-PRODUCTS

Process number: 1.3.2.4
Description:

This transformation processes the orders for products after it receives the information needed through (NEEDED-PRODUCTS). Once it places an order, it updates the ORDERS-STATUS file with that order. After the orders have been processed, they are then sent out to the production facility.

Input/Output Information:

Data Entering Data Leaving
needed-products products-orders

Parent:

Inv.-Control-Operations

Transformation primitive: PROCESS-POST-BN-OPERATIONS

Process number: 2.2.4.3.5.2

Description:

As it receives the jobs released, this transformation is enabled by the CONTROL process to start the production of the jobs on the machines following the bottleneck operation. The finished products are then stored in the FACILITY-PRODUCTS-INVETORY data store. A signal (EV-PRODUCTION-COMPLETED) indicating the completion of production is sent to the respective transformation(s).

Input/Output Information:
Structured English:

FOR (each occurrence of JOBS and EV-ENAB.) DO

BEGIN

Process the jobs on the machines,
Store the finished jobs in FINISHED-PRODUCTS-INVENTORY.

END

Parent:

Start-Post-BN-Operations

Transformation primitive: PROCESS-PRODUCTION-ORDERS

Process number: 2.1.3.5.4

Description:

Processes the orders for production once it receives the list of NEEDED-PRODUCTS. Once the orders have been processed, they are sent to the transformation in charge of the manufacturing operations.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>needed-products</td>
<td>production-orders</td>
</tr>
</tbody>
</table>

Structured English:

FOR (each occurrence of NEEDED-PRODUCTS) DO

BEGIN
Access FACILITY-ORDERS-STATUS,
FOR (each product) DO

Update the data store FACILITY-ORDERS-STATUS
Tag the orders as production orders and sends them
to the manufacturing transformation

END

Parent:
Products-Inv.-Control

Process: PRODUCTION-PLAN.-OPERATIONS
Process number: 2.2

Description:
This transformation (Manufacture-Control-Products) rep-
resents the production and inventory control oper-
ations of the products at the production facility.

Input/Output Information:

Data Entering
EV-time
reports-request
EV-Mtl-arrived
production-orders

Data Leaving
EV-production-completed
EV-Mtl-not-available
production-reports

Parent:
DPMF-Operations

Child processes:
Production-Operations
Check-Avail.-of-Mtl
Process: PRODUCTION-OPERATIONS
Process number: 2.2.4
Description:
This transformation represents the production operations at the production facility. These operations involve the grouping of jobs, sequencing and scheduling of jobs, controlling the production flow, etc.
Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>production-orders</td>
<td>production-reports</td>
</tr>
<tr>
<td>reports-request</td>
<td>EV-production-completed</td>
</tr>
<tr>
<td>EV-Mtl-available</td>
<td>selected-orders</td>
</tr>
</tbody>
</table>

Parent:
Production-Plan.Control

Child processes:
Sequence-Jobs
Synchronize-Production
Group-and-Assign-Job-Numbers

Process: PRODUCTS-INV.-CONTROL
Process number: 2.1.3.5
Description:
Controls the inventory levels of the products at the production facility.

Input/Output Information:

**Data Entering**
- EV-shipments
- EV-B.O.
- products-shipments
- EV-time
- back-orders

**Data Leaving**
- production-orders

Parent:
- Shipments-Inv-Control

Child processes:
- Identify-PF-Policy
- CONTROL-2.1.3.5.1-2.1.3.5.2
- Process-Production-Orders
- Apply-PF-Inventory-Policy
- Apply-PF-On-Order-Policy

**Process:** RECEIVE-DISPATCH-PRODUCTS

**Process number:** 1.3.1

**Description:**

This transformation is in charge of the receiving and dispatching operations of the products at the warehouses. Once a demand for products is received, it dispatches (PRODUCTS DISPATCHES) the products to the market if they are available in inventory. Also, once
it receives a PRODUCTS-SHIPMENTS from the production facility, it updates the PRODUCTS-INVENTORY and the ORDER-STATUS data stores. The information regarding the dispatched products (DISPATCHED-PRODUCTS) are sent to the transformation INV-CONTROL-OPERATIONS.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>products-shipments</td>
<td>dispatch-info.</td>
</tr>
<tr>
<td>demanded-products</td>
<td>products-dispatches</td>
</tr>
<tr>
<td></td>
<td>EV-product-not-avail.</td>
</tr>
</tbody>
</table>

Parent:
Inv-Orders-Operations

Child processes:
Update-WH-Orders-Status
Update-WH-Inventory
CONTROL-2.3.1.4
Receive-Finished-Products
Dispatch-Products
Check-WH-Inventory

Transformation primitive: RECEIVE-FINISHED-PRODUCTS

Process number: 1.3.1.4

Description:
This process is in charge of receiving the shipments of products (PRODUCTS-SHIPMENTS) from the production facility. Once it receives the shipment it sends a
signal (EV-SHIP.-RECD.) to C-CONTROL-2.3.1.4 and sends the information regarding the orders received (RECEIVED-ORDERS) to UPDATE-ORDERS-STATUS.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>products-shipments</td>
<td>received-orders</td>
</tr>
<tr>
<td></td>
<td>received-products</td>
</tr>
</tbody>
</table>

Structured English:

For each occurrence of PRODUCTS-SHIPMENTS,

BEGIN

Update the inventory with received products

Update the status of the orders that have been satisfied with the receipt of the shipment.

END

Parent:

Receive-Dispatch-Products

Transformation primitive: RELEASE-JOBS-FOR-PRODUCTION

Process number: 2.2.4.3.3.3

Description:

This transformation releases the jobs for production once it is enabled by the CONTROL process. It accesses the data stores GROUPED-SEQUENCED-JOBS for the jobs to be released, JOBS-ROUTINGS for the routes the jobs will take, and RAW-MATERIALS-INVENTORY for the materials needed to manufacture the products.
Structured English:

FOR (each enabling signal (EV-ENAB.)) DO

BEGIN

Access the data stores GROUPED-SEQUENCED-JOBS and

RAW-MATERIALS-INVENTORY,

Release the jobs for production.

END

Parent:

Start-Pre-BN-Operations

Transformation primitive: RELEASE-JOBS-FROM-I

Process number: 2.2.4.3.4.1

Description:

This process releases the jobs from the machines' upstream queue to be processed. It accesses the data store JOBS-QUEUE-I to release the jobs.
Access the data store QUEUE-I and release the jobs to the machines.

Parent:
Start-BN-Operation

Transformation primitive: RELEASE-JOBS-FROM-II
Process number: 2.2.4.3.5.1
Description:
This transformation accesses the JOBS-QUEUE-II and releases the jobs to be produced on the machines following the bottleneck. It signals the CONTROL process of the release of the jobs once it is done.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-BN-completed</td>
<td>EV-jobs-released</td>
</tr>
<tr>
<td></td>
<td>released-jobs</td>
</tr>
</tbody>
</table>

Structured English:
FOR (each occurrence of EV-BN-COMPLETED) DO
Access the JOBS-QUEUE-II and release the jobs for production.

Parent:
Start-Post-BN-Operations

Transformation primitive: RESTORE-OLD-SOLUTION
Process number: 2.2.4.3.1.2.2
Description:
Once the revised linear program is received, this transformation restores the old solution from the MASTER-LP-SOLUTION-FILE, and sends it to SOLVE-REVISED-LP.

Input/Output Information:

Data Entering
EV-LP-formulated

Data Leaving
restored-solution

Structured English:

FOR (each occurrence of REVISED-LP) DO

BEGIN

Access the data store MASTER-LP-SOLUTION-FILE,

Restore the old solution to the linear program model

END

Parent:

Solve-LP

Transformation primitive: RETRIEVE-SHIP-PRODUCTS

Process number: 2.1.3.2

Description:

Retrieves the products ordered from inventory upon availability and ships them to the warehouses.

Input/Output Information:

Data Entering
EV-production-completed
EV-products-avail.

Data Leaving
EV-shipments
products-shipments

Structured English:
FOR (each occurrence of (EV-PRODUCTS-AVAIL.)) OR (EV-PRODUCTION-COMPLETED) DO

Access FINISHED-PRODUCTS-INVENTORY data store and ship the product(s).

Parent:

Shipments-Inv-Control

Transformation primitive: REVISE-LP

Process number: 2.2.4.3.1.2.1

Description:

Revises the linear program model to check for feasibility of the model. It accesses the data store MASTER-LP-FILE to retrieve the information needed about the model. The flow EV-LP-FORMULATED signals this transformation to start.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-LP-formulated</td>
<td>revised-LP</td>
</tr>
</tbody>
</table>

Structured English:

FOR (each occurrence of EV-LP-FORMULATED) DO

BEGIN

Access the data store MASTER-LP-FILE,

Revise the linear program model

END
Parent:

Solve-LP

Process: SOLVE-TRAVEL-SALES-PROB.

Process number: 2.2.4.3.2.4

Description:

If it has been identified that the sequencing problem involves one bottleneck machine feeding another, then this transformation will schedule the production using the travelling salesman problem approach.

Input/Output Information:

Data Entering      Data Leaving
BN-involves-setup  BN-schedule

Parent:

Schedule-Bottleneck

Process: START-BN-OPERATION

Process number: 2.2.4.3.4

Description:

This transformation starts the production on the bottleneck operation once the message EV-PRE-BN-COMPLETED. Looking at the child diagram of this transformation, the first process involves releasing the jobs for production as soon as the bottleneck is available to process the jobs waiting in QUEUE-I. The second transformation processes the jobs on the bottleneck.
Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-Pre-BN-completed</td>
<td>EV-BN-completed</td>
</tr>
</tbody>
</table>

Parent:

Synchronize-Production

Child processes:

Process-BN-Operation
Release-Jobs-from-I

Process: SYNCHRONIZE-PRODUCTION

Process number: 2.2.4.3

Description:

This transformation starts and controls the production of the jobs. As shown at the lower level diagram, it first identifies and schedules the bottleneck operation. The next process involves the production of jobs on the pre-bottleneck operations. The forth process starts the bottleneck operation once the production is done on the preceding operations. Finally, when the production is done on the bottleneck operation, the fifth process controls the production of the jobs on the post-bottleneck operations.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV-jobs-grouped</td>
<td>EV-production-completed</td>
</tr>
</tbody>
</table>
EV-time

Parent:
Production-Operations

Child processes:
Start-Post-BN-Operations
Start-BN-Operation
Start-Pre-BN-Operations
Schedule-Bottleneck
Identify-BN.-Operations

Process: START-POST-BN-OPERATIONS
Process number: 2.2.4.3.5

Description:
This transformation starts the processing of the jobs on the machine centers following the bottleneck operation. Once the message EV-BN-COMPLETED is received, the first process releases the jobs for production from QUEUE-II. After the jobs have been released, the CONTROL process is signaled to enable the processing of the jobs.

Input/Output Information:

Data Entering
EV-BN-completed

Data Leaving
EV-production-completed

Parent:
Synchronize-Production
Child processes:
Process-Post-BN-Operations
Release-Jobs-From-II

Process: START-PRE-BN-OPERATIONS
Process number: 2.2.4.3.3

Description:
This transformation represents the different operations involved in starting and control the production on the machines that precede the bottleneck operations. As shown on its child diagram, the first process is to determine the time (T) to release the jobs for production. is to determine the time (T) to release the jobs for production. Once the current time (EV-TIME) is equal to T, the control process enables the transformation RELEASE-JOBS-FOR-PRODUCTION to release the jobs to the machine centers for production.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>BN-schedule</td>
<td>EV-Pre-BN-completed</td>
</tr>
<tr>
<td>EV-time</td>
<td></td>
</tr>
</tbody>
</table>

Parent:
Synchronize-Production

Child processes:
Release-Jobs-for-Production
Compare-T-with-Time
Transformation primitive: SYNTHESIZE-LP

Process number: 2.2.4.3.1.1.5

Description:

This transformation synthesizes the formulated objective function, right-hand side, and the technology matrix to formulate the final linear program model. It then stores the formulated linear program model in the data store FORMULATED-LP-MODEL.

Input/Output Information:

Data Entering

formulated-matrix
formulated-RHS
formulated-objective

Data Leaving

EV-LP-formulated

Structured English:

FOR (each occurrence of FORMULATED-OBJECTIVE, RHS VECTOR, and TECH.-MATRIX) DO

Synthesize the function objective, the right-hand side vector, and the technology matrix to formulate the final linear program model.

Parent:

Initial-LP-Formulation

Transformation primitive: TEST-SOLUTION-FEASIBILITY

Process number: 2.2.4.3.1.2.4
Description:

This transformation tests the feasibility of the new linear program solution. It then stores the new feasible solution in the data store MASTER-LP-SOLUTION-FILE.

Input/Output Information:

Data Entering       Data Leaving
new-LP-solution     EV-LP-solved

Structured English:

FOR (each occurrence of NEW-LP-SOLUTION) DO
BEGIN
Test the feasibility of the new solution to the linear program model.
Store the new solution in MASTER-LP-SOLUTION-FILE
END

Parent:
Solve-LP

Process: TEST-VARIABLES-VALUE
Process number: 2.2.4.3.1.3.2

Description:

Tests the values of the slack variables that were recognized as being basic variables. If the value of the variable is greater than zero, then the corresponding machine center is not a bottleneck operation. Otherwise, the machine is a bottleneck.
Input/Output Information:

Data Entering
  
  basic-variables

Data Leaving
  
  vars-greater-than-zero
  vars-equal-zero

Parent:

Determine-BN

Process: UPDATE-DPMF-FILES
Process number: 2.1.2
Description:

This process updates the production facilities files with the new products' data. This information is stored in DPMF-PRODUCTS-FILE data store.

Input/Output Information:

Data Entering
  
  EV-New-product

Data Leaving

Parent:

Orders-Inv-Operations

Child processes:

Determine-Mfg.-Processes
Determine-Engr.-Specs.
Update-PF-Products-List

Process: UPDATE-HISTORICAL-DATA
Process number: 1.4.4
Description:
Once the information regarding the PRODUCTS-DISPATCHES is received, this transformation updates HISTORICAL-DISPATCHES-DATA.

Input/Output Information:

Data Entering: products-dispatches

Data Leaving: 

Parent:

Forecast-Demand

Process: UPDATE-INV-ORDERS-STATUS

Process number: 2.3.1

Description:

Once raw materials are received, which correspond to an earlier order, the order status is changed from outstanding to satisfied, and the inventory levels are also updated.

Input/Output Information:

Data Entering: materials

Data Leaving: EV-Mtl-arrived

Parent:

Raw-Materials-Control

Process: UPDATE-MARKET-INVENTORY

Process number: 1.4.3

Description:
This transformation updates the market's inventory of products whenever SALES or PRODUCTS-DISPATCHES occur.

Input/Output Information:

Data Entering
sales
products-dispatches

Data Leaving

Parent:
Forecast-Demand

**Transformation primitive:** UPDATE-PF-ORDERS-FILE

Process number: 2.1.3.3

**Description:**

Updates the status of the facility's products' orders whenever a shipment (PRODUCTS-SHIPMENTS) occurs in correspondence to an earlier order.

Input/Output Information:

Data Entering
products-shipments

Data Leaving

Structured English:

\[
\text{FOR (each occurrence of PRODUCTS-SHIPMENTS) DO}
\]

Access FACILITY-ORDERS-STATUS data store

Set on-order(product,warehouse) =

on-order(product,warehouse) - shipment.

Parent:

Shipments-Inv-Control
Process: UPDATE-PF-PRODUCTS-LIST
Process number: 2.1.2.1
Description:

Updates the PRODUCTS-FILE data store with the information regarding the new products ordered.

Input/Output Information:

Data Entering          Data Leaving
EV-New-product

Parent:
Update-DPMF-Files

Process: UPDATE-WH-FILES
Process number: 1.2
Description:

This transformation updates the WAREHOUSES-PRODUCTS-FILES. Once it receives the signal EV-NEW-PRODUCTS from IDENTIFY-DEMANDED-PRODUCTS, it retrieves the new products from NEW-PRODUCTS-DEMANDED data store and updates WAREHOUSES-PRODUCTS-FILES with the respective information.

Input/Output Information:

Data Entering          Data Leaving
EV-New-product

Parent:
Warehouses-Operations
Transformation primitive: UPDATE-WH-INVENTORY

Process number: 1.3.1.3

Description:

Upon the receipt of products (RECEIVED-PRODUCTS), this process updates the level of inventory by the received amount. It also updates the inventory by the amount of products dispatched (DISPATCH-INFO.).

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>dispatch-info.</td>
<td>received-orders</td>
</tr>
</tbody>
</table>

Parent:

Receive-Dispatch-Products

Transformation primitive: UPDATE-WH-ORDERS-STATUS

Process number: 1.3.1.5

Description:

This transformation updates the status of the orders sent to the production facility for products. Once a shipment is received that corresponds to an earlier order, the status of that order is changed from outstanding to complete in the data store ORDERS-STATUS.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>received-orders</td>
<td></td>
</tr>
</tbody>
</table>

Structured English:

For each occurrence of RECEIVED-ORDERS,
Update the status of the order in the data store ORDERS-STATUS.

Parent:
Receive-Dispatch-Products

Process: WAREHOUSES-OPERATIONS
Process number: 1
Description:
This transformation represents the operations of the warehouses. These operations include processing orders for products from the production facility, dispatching products to the market, and controlling the inventory of products.

Input/Output Information:

<table>
<thead>
<tr>
<th>Data Entering</th>
<th>Data Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>products-shipments</td>
<td>products-dispatches</td>
</tr>
<tr>
<td>demand</td>
<td>products-orders</td>
</tr>
<tr>
<td>EV-time</td>
<td>warehouses-reports</td>
</tr>
<tr>
<td>reports-request</td>
<td></td>
</tr>
</tbody>
</table>

Parent:
IPPIC-Operations

Child processes:
Inv-Orders-Operations
Update-WH-Files
Identify-Demanded-Product
APPENDIX C

LISTINGS OF OUTPUT DATA FILES
The following is an example of the warehouses data file generated by the simulation model for a case study. The DISTRICTS represent the warehouses, SIGN represent the products, and DISPATCH represents the products shipped from the warehouses to the market.

(I) WAREHOUSES DATA FILE:

FILE: WAREHS DATA B  VM/SP CONVERSATIONAL MONITOR SYSTEM

** THE INVENTORY POLICY USED AT THE WAREHOUSES IS ORDER-UP-TO I(max) POLICY.  
** THE INVENTORY POLICY USED AT THE SIGN SHOP IS PERIODIC-REVIEW POLICY.  
** THE PRODUCTION RULE USED IS SHORTEST PROCESSING TIME.

<table>
<thead>
<tr>
<th>DISTRICT</th>
<th>SIGN</th>
<th>DEMAND</th>
<th>ORDER DUE</th>
<th>ONORDER</th>
<th>DATE</th>
<th>DATE</th>
<th>DISPATCH</th>
<th>INV.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CUM. ORDER DUE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>S-0</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>15</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>R-37R</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>24</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>D-84</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>33</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

INVENTORY CARRYING COSTS OF THIS PERIOD = 9324.35

PERIOD 2

<table>
<thead>
<tr>
<th>PERIOD</th>
<th>DISTRICT</th>
<th>SIGN</th>
<th>DEMAND</th>
<th>ORDER DUE</th>
<th>ONORDER</th>
<th>DATE</th>
<th>DATE</th>
<th>DISPATCH</th>
<th>INV.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>D-1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>31</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>N-50</td>
<td>5</td>
<td>5</td>
<td>13</td>
<td>40</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>N-50</td>
<td>3</td>
<td>3</td>
<td>13</td>
<td>35</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>D-1</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>14</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

INVENTORY CARRYING COSTS OF THIS PERIOD = 9324.35
The following is an example of the sign shop data file generated by the simulation model:

(II) SIGN SHOP DATA FILE:

FILE: SHOP DATA B VM/SP CONVERSATIONAL MONITOR SYSTEM

** THE INVENTORY POLICY USED AT THE WAREHOUSES IS ORDER-UP-TO I(max) POLICY.
** THE INVENTORY POLICY USED AT THE SIGN SHOP IS PERIODIC-REVIEW POLICY.
** THE PRODUCTION RULE USED IS SHORTEST PROCESSING TIME.

<table>
<thead>
<tr>
<th>DISTRICT</th>
<th>SIGN</th>
<th>DEMAND</th>
<th>ORDER DATE</th>
<th>DUE DATE</th>
<th>SHIPPED DATE</th>
<th>AMOUNT</th>
<th>INV. AFTER SHIPPED</th>
</tr>
</thead>
</table>

PERIOD 1

<table>
<thead>
<tr>
<th>10</th>
<th>S-0</th>
<th>3</th>
<th>1</th>
<th>15</th>
<th>10</th>
<th>3</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D-84</td>
<td>1</td>
<td>3</td>
<td>33</td>
<td>33</td>
<td>1</td>
<td>24</td>
</tr>
</tbody>
</table>

INVENTORY CARRYING COSTS OF THIS PERIOD = 1932.35

PERIOD 2

<table>
<thead>
<tr>
<th>2</th>
<th>D-1</th>
<th>1</th>
<th>10</th>
<th>31</th>
<th>10</th>
<th>1</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>N-50</td>
<td>5</td>
<td>13</td>
<td>40</td>
<td>13</td>
<td>5</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>N-50</td>
<td>3</td>
<td>13</td>
<td>35</td>
<td>13</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>D-1</td>
<td>7</td>
<td>14</td>
<td>30</td>
<td>14</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

INVENTORY CARRYING COSTS OF THIS PERIOD = 1932.35
The following is an example of the file containing the performance of the system data:

(III) RESULTS DATA FILE:

FILE: RESULTS DATA B  VM/SP CONVERSATIONAL MONITOR SYSTEM

** THE INVENTORY POLICY USED AT THE WAREHOUSES IS ORDER-UP-TO \( I(\text{max}) \) POLICY.
** THE INVENTORY POLICY USED AT THE SIGN SHOP IS PERIODIC-REVIEW POLICY.
** THE PRODUCTION RULE USED IS SHORTEST PROCESSING TIME.

* AVERAGE DELIVERY LEAD TIME = 5.89 DAYS
* VARIANCE FOR DELIVERY LEAD TIME = 12.35
* AVERAGE LATENESS OF JOBS FOR STOCKED PRODUCTS= 36.44 DAYS
* VARIANCE FOR LATENESS OF JOBS FOR STOCKED PRODUCTS=366.30
* AVERAGE LATENESS OF JOBS FOR NON-STOCKED PRODUCTS=60 DAYS
* VARIANCE FOR LATENESS OF JOBS FOR NON-STOCKED PRODUCTS= 1360
* AVERAGE INVENTORY CARRYING COSTS AT WAREHOUSES = 1020.98
* AVERAGE INVENTORY CARRYING COSTS AT WAREHOUSES = 9.66
* PERCENTAGE OF ORDERS SATISFIED FROM INVENTORY = 0.1775

<table>
<thead>
<tr>
<th>MACHINE</th>
<th>AVERAGE UTILIZATION</th>
<th>MAX. UTIL.</th>
<th>MIN. UTIL.</th>
<th>VARIANCE OF ACTIVE TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0716</td>
<td>0.520</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>2</td>
<td>0.0103</td>
<td>0.149</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>0.1587</td>
<td>0.783</td>
<td>0.00</td>
<td>0.03</td>
</tr>
</tbody>
</table>
LABOR EFFICIENCY

<table>
<thead>
<tr>
<th>MACHINE CENTER</th>
<th>PERCENTAGE SPENT ON PRODUCTION</th>
<th>PERCENTAGE SPENT ON SETUP</th>
<th>PERCENTAGE SPENT ON M/C SETUP AS A FRACTION OF THE ACTIVE TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1238</td>
<td>0.0363</td>
<td>0.1874</td>
</tr>
<tr>
<td>2</td>
<td>0.4015</td>
<td>0.1214</td>
<td>0.1827</td>
</tr>
<tr>
<td>3</td>
<td>0.2907</td>
<td>0.0360</td>
<td>0.0795</td>
</tr>
</tbody>
</table>

The following file represents the file generated from the linear programming model for identifying the bottleneck:

(IV) BOTTLENECK OPERATION FILE:

FILE: BOTTLEN DATA B  VM/SP CONVERSATIONAL MONITOR SYSTEM

PERIOD 1

BOTTLENECK OPERATIONS ARE:

--- NONE ---

PERIOD 2

BOTTLENECK OPERATIONS ARE:

--- NONE ---

PERIOD 3

BOTTLENECK OPERATIONS ARE:

6 (sheeting and trimming)

PERIOD 24

BOTTLENECK OPERATIONS ARE:

6 (sheeting and trimming)