

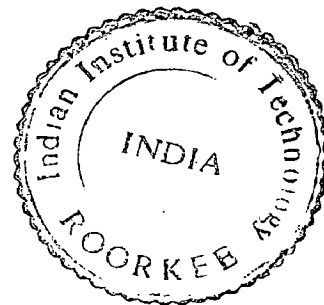
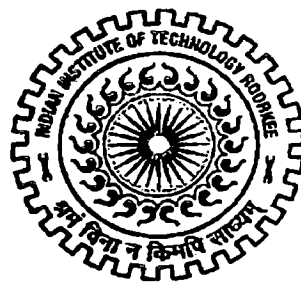
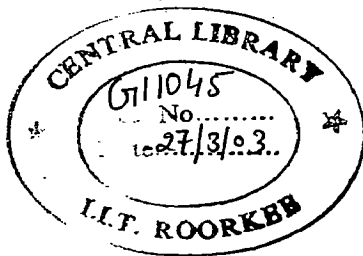
APPLICATIONS OF MATERIAL REQUIREMENTS PLANNING MODEL (MRP) IN WATER RESOURCES PROJECTS

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*
of
MASTER OF TECHNOLOGY
in
WATER RESOURCES DEVELOPMENT

By

NIHAR KANTI MISHRA



**WATER RESOURCES DEVELOPMENT TRAINING CENTRE
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
February, 2003**

12

CANDIDATE'S DECLARATION

I hereby certify that the work presented in this dissertation titled “APPLICATIONS OF MATERIAL REQUIREMENTS PLANNING MODEL (MRP) IN WATER RESOURCES PROJECTS” which is submitted in the partial fulfillment of the requirements for the award of the degree of **Master of Technology in Water Resources Development (Mechanical)**, of Indian Institute of Technology, Roorkee, is an authentic record of my own work carried out during the period from July 2002 to February 2003 under the supervision of Prof. Gopal Chauhan, WRDTC, and Associate Prof. Dr. Pradeep Kumar, Department of Mechanical and Industrial Engineering.

The matter embodied in this dissertation has not been submitted by me for award of any other degree or diploma.

Dated: 13.02.2003

ROORKEE

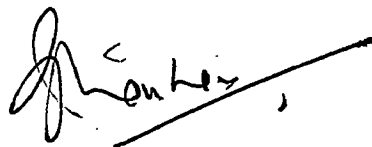
Nihar Kanti Mishra.
NIHAR KANTI MISHRA

This is to certify that the above statement made by the candidate is correct to the best of our knowledge and belief.



Dr. PRADEEP KUMAR

Associate Professor,
Department of Mechanical
& Industrial Engineering,
Indian Institute of Technology,
ROORKEE – 247 667,
INDIA.



PROF. GOPAL CHAUHAN

Professor,
Water Resources Development
Training Centre,
Indian Institute of Technology,
ROORKEE – 247 667,
INDIA.

ACKNOWLEDGEMENTS

It is my great pleasure to express my profound gratitude to **Prof. Gopal Chauhan**, Professor, Water Resources Development Training Centre, and **Dr. Pradeep Kumar**, Associate Professor, Department of Mechanical and Industrial Engineering, for their timely guidance, valuable suggestions, constant encouragement, moral support, and painstaking supervision which were my guiding forces in completing this dissertation successfully.

I am very much thankful to **Prof. U. C. Chaube**, Professor and Head, Water Resources Development Training Centre, for providing me all sorts of help right through my entire training period in Roorkee.

My sincere thanks and appreciation are also due to all other faculty members of Water Resources Development Training Centre for their valuable teaching, guidance, assistance, and encouragement during my entire study period in Roorkee.

The cooperation and help extended by all the staff members of Water Resources Development Training Centre, my fellow trainee officers, and friends in preparing and finishing this work is greatly acknowledged.

I am thankful to my parent department, "Department of Water Resources, Government of Orissa" for sponsoring me to this course in WRDTC. My sincere thanks also go specifically to Sri B. K. Pattanaik, IAS, Commissioner-Cum-Secretary, DOWR, Government of Orissa, and Er. B. B. Singhsamant, Engineer-in-Chief, DOWR, Government of Orissa in this regard.

I am deeply indebted to my father Late **Prof. Nrusingh Charan Misra**, mother **Smt. Sarat Kumari Mishra**, for their whole-hearted sacrifice, amazing tolerance, blessings, and good wishes in successful completion of this work.

Finally, I would like to sincerely thank my wife **Reena** and daughter **Avipsa**, for their love, sacrifice, and support in bringing out this work.

Nihar Kanti Mishra.
NIHAR KANTI MISHRA

CONTENTS

	Page
CANDIDATE'S DECLARATION	(i)
ACKNOWLEDGEMENTS	(ii)
CONTENTS	(iii)
ABBREVIATIONS	(vi)
LIST OF FIGURES	(vii)
LIST OF FORMS	(viii)
LIST OF TABLES	(ix)
SYNOPSIS	(x)
CHAPTER 1 INTRODUCTION	
1.1 MATERIALS MANAGEMENT	1
1.1.1 Materials for Water Resources Projects	3
1.2.MATERIAL REQUIREMENTS PLANNING MODEL	4
1.2.1MRP System Prerequisites and Assumptions	6
1.2.2.Regenerative System	6
1.2.3.Net Change System	7
1.3.SCOPE OF STUDY AND THESIS ORGANIZATION	7
CHAPTER 2 LITERATURE REVIEW	
2.1 GENERAL	9
2.2 MRP SYSTEM BASICS	11
2.2.1System Inputs	11
2.2.1.1.Master Production Schedule	13
2.2.1.2.Bill of Materials	15
2.2.1.3.Inventory Status Records	21
2.2.2 Lot Sizing Techniques	21
2.2.3.Time Phased Order Point Technique	23
2.2.4.Role of Safety Stock	23

2.2.5.	System Outputs	24
2.2.5.1.	Planned Order Receipts	24
2.2.5.2.	Planned Order Releases	26
2.2.5.3.	Exception Reporting	28
2.2.5.4.	Performance Control	29
2.3.	OBJECT-ORIENTED SOFTWARE	
	PARADIGMS	
2.3.1	Abstraction	30
2.3.2	Encapsulation	30
2.3.3	Modularity	30
2.3.4	Hierarchy	31
2.3.5	Typing	31
2.3.6	Concurrency	31
2.3.7	Persistence	31
CHAPTER 3	APPLICATIONS OF MRP TO WATER	
	RESOURCES PROJECTS	
3.1.	GENERAL	32
3.2.	MRP SYSTEM FOR WATER RESOURCES	32
	PROJECTS	
3.3.	SYSTEM CONTROLS	37
3.4.	COMPARISON WITH CURRENT INVENTORY	38
	MODELS USED IN W.R. PROJECTS	
3.4.1.	Limitationsof Fixed Order Size System	38
3.4.2.	Comparison with EOQ Model	39
3.5.	SYSTEM INTEGRITY	39
3.6.	TRAINING OF PROJECT PROFESSIONALS	40
CHAPTER 4	DESIGN OF MRP SOFTWARE	
4.1.	DESIGN BASICS	41
4.2.	METHODOLOGY AND	42
	ARCHITECTURE	
4.3.	DESIGN PROCESS	42
4.4.	OPERATING ENVIRONMENT	44
4.5.	LIMITATIONS OF THE SOFTWARE	44

CHAPTER 5	CASE STUDIES	
	5.1.GENERAL	45
	5.2.CASE 1	45
	5.2.1 Data Acquisition and Computations	45
	5.2.2. Results and Discussions	49
	5.3.CASE 2	50
	5.3.1. Data Acquisition and Computations	50
	5.3.2. EOQ Model	53
	5.3.3. MRP Model	53
	5.3.4. Results and Discussions	54
CHAPTER 6	CONCLUSIONS	
	6.1.GENERAL	66
	6.2.CASE STUDY 1	66
	6.3.CASE STUDY 2	67
	6.4.SCOPE FOR FUTURE RESEARCH	67
REFERENCES		68
APPENDIX:		71
SOFTWARE LISTING		72 - 117

ABBREVIATIONS

BOM	Bill of Materials
CRP	Capacity Requirements Planning
EGA	Enhanced Graphics Adapter
EOQ	Economic Order Quantity
EOT	Electric Overhead Travelling Crane
GUI	Graphical User Interface
JIT	Just In Time
MPS	Master Production Schedule
MRP	Material Requirements Planning
MRP II	Manufacturing Resources Planning
OOD	Object-Oriented Design
OOP	Object-Oriented Programming
VGA	Video Graphics Array Adapter

LIST OF FIGURES

Sl. No.	Figure No.	Description	Page
1	Fig. 1.1	Flow of Information in a Typical MRP System	5
2	Fig. 2.1	Typical Closed Loop MRP System	10
3	Fig. 2.2	Typical MRP System Inputs and Outputs	12
4	Fig. 2.3	Single Stage Planning Decision for a Horizon of One Period	14
5	Fig. 2.4	Flow Chart of a Standard MRP System	25
6	Fig. 4.1	MVC Architecture	43
7	Fig. 5.1	Inventory Levels for MRP Model for HSD Oil	52
8	Fig. 5.2	EOQ Model 1 for the Horizontal Girder	56
9	Fig. 5.3	EOQ Model 2 for the Rolled Steel Beams	57
10	Fig. 5.4	Inventory Levels for the MRP Model for the Hz. Girder	64
11	Fig. 5.5	Inventory Levels for the MRP Model for the Rolled Steel Beams	65

LIST OF FORMS

Sl. No.	Form No.	Description	Page
1	Form 2.1	Single Level Explosion Format	17
2	Form 2.2	Indented Explosion Format	17
3	Form 2.3	Summary Explosion Format	17
4	Form 2.4	Single Level Implosion Format	18
5	Form 2.5	Indented Implosion Format	18
6	Form 2.6	Summary Implosion Format	18
7	Form 2.7	Matrix Bill of Materials	20
8	Form 2.8	Comparative Bill of Materials	20
9	Form 2.9	Modular Bill of Materials	20
10	Form 2.10	Typical MRP Matrix	27

LIST OF TABLES

Sl. No.	Table No.	Description	Page
1	Table 3.1	EOQ and MRP Comparison	39
2	Table 5.1	Demand Rates for HSD Oil	47
3	Table 5.2	Master Production Schedule for HSD Oil	48
4	Table 5.3	Bill of Materials for HSD Oil	48
5	Table 5.4	Inventory Status Record for HSD Oil	48
6	Table 5.5	Planned Order Release for HSD Oil	49
7	Table 5.6	Inventory Levels for MRP Model for HSD Oil	51
8	Table 5.7	Inventory Levels in EOQ Model for Case 2	55
9	Table 5.8	Master Production Schedule for Level 1	58
10	Table 5.9	Bill of Materials for all Materials for Case 2	58
11	Table 5.10	Inventory Status Record for Level 1	58
12	Table 5.11	Inventory Status Record for Level 2	59
13	Table 5.12	Planned Order Release for Level 1	59
14	Table 5.13	Planned Order Release for Level 1	60
15	Table 5.14	Planned Order Release for Level 2	60
16	Table 5.15	Planned Order Release for Level 2	61
17	Table 5.16	Master Production Schedule for Level 2	61
18	Table 5.17	Master Production Schedule for Level 2	62
19	Table 5.18	Master Production Schedule for Level 2	62
20	Table 5.19	Inventory Levels in MRP Model for Case 2	63

SYNOPSIS

The management of materials or inventory concerns their flow to, within, and from the organization. The efficiency and efficacy of the flow can substantially influence costs and revenue generation and thus hold serious implications. Material Requirements Planning (MRP) is a computerized information processing system designed to plan and control the flow of materials. MRP is designed to release production and purchase orders in order to regulate the flow of raw materials, and in process inventories necessary to meet the production schedules for end items. The primary objectives of an MRP system are to determine gross and net requirements i.e. discrete period demands for each item of inventory, so as to be able to generate information needed for correct inventory procurement or production.

Now-a-days, the Water Resources Projects in most of the developing countries are mainly using the primitive methods or models in order to manage materials, though these projects involve huge material cost, which may be as high as 50% of the total capital outlay on an average.

Through this dissertation, extensive study has been carried out to develop an Object-Oriented software for the Material Requirements Planning (MRP) system, which can be used in the Water Resources Projects. This software is paperless, and it will operate in DOS mode, and also it has a user-friendly Graphical User Interface. Flow of a large number, and volume of materials can be planned and controlled through this software.

INTRODUCTION

1.1 MATERIALS MANAGEMENT

An organization is a management concept that encapsulates the structure, function, and objectives of any Water Resources Project. Materials Management is the basic part of any organization that produces products or gives services of economic value. The primary objective of the management of an organization is to minimize the expenditure by controlling its system functionalities. The cost of materials ranges from 10% to 80% of the capital expenditure depending on the type of the organization, and in case of Water Resources Projects, this cost percentage is usually more than 50%. In fact, the huge cost of materials used in the Water Resources Projects forces us to give a serious thought about it to develop an efficient and transparent planning and controlling tool for these materials, because in the present days, cost conscious organizations need to maintain minimum inventory level so as to have minimum holding and ordering costs, and to make efficient record system for the materials management department to avoid excess inventory and deadly stock-outs. The major goals of materials management are to minimize inventory investment, maximize customer service, and assure efficient but low cost operation of the organization, and through this dissertation we emphasize to achieve them by using a befitting planning and controlling tool named 'Material Requirements Planning'.

There are at least four reasons to hold inventories or materials for the future, and they can be stated as,

1. to provide a quick response to the customers,
2. to provide safety against uncertainties,
3. to increase operating efficiency of the organization, and
4. to take advantage of the unusual pricing opportunities.

The variety of materials used in various organizations pose a very difficult problem associated with the materials management. These materials can also be further classified into three types:

(a) Independent Demand Items:

Demand for materials is considered independent when no relationship exists between the demand for an item and any other item. Independent demand customarily exhibits a continuous and definable pattern but fluctuates because of random influences from the marketplace. The demand for independent items is subject to customer preferences and needs.

(b) Dependent Demand Items:

Demand is classified as dependent when a direct, mathematical relationship exists between the demand for an item and another 'higher level' or 'parent' item. Demand for dependent items is the result of the requirements generated for their use in the production of another item, as in the case of raw materials, parts, and subassemblies. Thus, the demand for the final product may be continuous and independent, while the demand for the subordinate items composing the product tends to be discrete, derived, and dependent.

(c) Mixed Demand Items:

A given inventory item may be subject to both dependent and independent demand. Such mixed demand arises in cases of parts used in current production as well as spare-part service.

Every organization dealing with inventories has an inventory policy that depicts the quantum of each material required either for purchasing, or for using, or for selling from time to time. The various inventory policies can be broadly classified as:

(1) Deterministic Models

In the deterministic models, which are applied to both uniform, independent demand and time varying discrete demand conditions, all the parameters and variables are known or can be calculated with certainty. The demand rate for units, and the related inventory costs are assumed to be known, and also the replenishment lead time is presumed to be constant, and independent of demand. The optimum inventory policies of these models are to determine economic lot sizes for the independent demand items, or dependent demand items, whether they are purchased from a supplier or produced in the organization itself because discrete demand patterns can occur in either independent or dependent demand items. But in

actual practice, these models have been failed in determining the correct quantity and correct time for inventories in use, production, and sell.

(2) Statistical Models

In statistical models, both the demand rate and the lead time are treated as random variables, and the average demand is assumed to be constant over time, and so the demand is stated in terms of probability distribution. Here for both the discrete and continuous demand distribution, the expected cost is minimized. Inventory of independent demand items can be categorized into working stock and safety stock or buffer stock. Working stock is that stock which is expected to be used in a given time period, but the safety stock is determined directly from the forecast. The random variations of the demand and lead time are absorbed by the safety stock. The safety stock is the difference between the stock available at which an order is triggered, or reorder point and the average demand during this replenishment period.

(3) Material Requirements Planning Model

Material Requirements Planning model is a computer based system designed to release production and purchase orders in order to regulate the flow of inventories in an organization. Through 'time phasing', it enables the organizations to maintain minimum levels of dependent demand items and at the same time the production schedules for the independent items can be met.

(4) Just in Time

This model attempts to virtually eliminate all costs that do not add value to the products. Continuous improvement and attention to any barrier to product flow is the main concept behind JIT. Inventory is considered an undesirable cost, and by lowering its level, quality and productivity obstructions can be eliminated.

1.1.1 Materials for Water Resources Projects

Depending on the type of Project, use of materials will vary. We can broadly classify these materials into three categories:

(a) Construction Materials:

Under this category, we can put cement, iron & steel, sand, and crushed stone.

(b) Generating Equipment Materials:

Under this category, we can take spare parts and accessories of turbines, hydro-generators, and control systems.

(c) Materials for Construction Plant and Equipment:

In this category, we can keep spare parts and accessories of heavy earth moving equipment, welding machines, pumps, EOTs, gates and valves, cable systems, air compressors, drilling machines, lathe machines, milling machines, grinding machines, shaping machines, plate bending machines, tunneling equipment, belt-conveyor systems, and pile-driving equipment. Also here we can put separately fuel and lubrication oil for related equipment; chemicals and explosives for tunneling; welding rods, acetylene, and oxygen gas for welding; and paints for paint shop.

The manufacturing activities that take place in a Water Resources Project (may not be in the site) can be the manufacturing of different types of gates, and steel ribs for tunnels.

1.2 MATERIAL REQUIREMENTS PLANNING MODEL

“Material Requirements Planning” is the new way of life in production and inventory management, displacing older models in general and statistical inventory control in particular⁽²¹⁾. An MRP system can be defined as a set of logically linked item inventory records, coupled with a computer program that maintains these records up to date (Fig. 1.1). This system enables organizations to develop realistic plans and to coordinate resources to ensure the availability of materials, components, and products for planned production and for customer delivery by maintaining minimum levels of dependent demand items, yet simultaneously promising that production schedules for the independent items can be met. It does so through proper timing of order placement and hence it is also known as “Time-Phased Requirements Planning”⁽²⁴⁾.

“Material Requirements Planning” systems entered the doors of materials management departments of the organizations since 1961 slowly and steadily. In the area of inventory and production management, the most successful innovation⁽²¹⁾ was

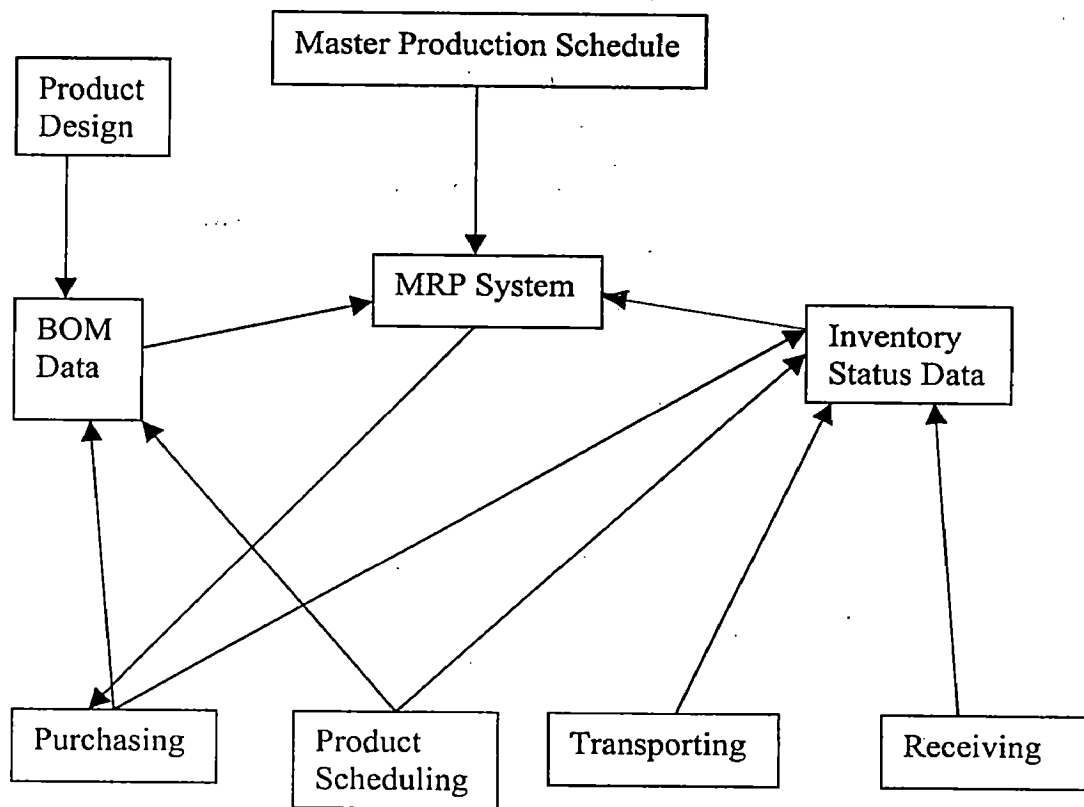


Fig. 1.1: Flow of Information in a Typical MRP System

the MRP system. Inventory and production management was traditionally giving emphasis on two terms quantity, and timing. In MRP systems, timing was more important and by introducing 'time phasing' concept to the MRP, the system became more refined⁽¹⁸⁾ and effective than ever before.

1.2.1 MRP System Prerequisites and Assumptions

Every constructed system has several prerequisites and they reflect certain fundamental assumptions. MRP system is no exception.

The prerequisites⁽²¹⁾ are,

- A 'Master Production Schedule' must exist,
- Ability of the 'Master Production Schedule' to be stated in terms of bill of material terms,
- Each and every inventory item must be unambiguously identified through a unique code or part number,
- A bill of material exists at planning period, and it must be set in a structured manner to reflect the design procedure of the products, and
- 'Inventory Status Record' for all items under the system's control must exist.

The assumptions⁽¹⁶⁾ adopted are,

- MRP system file data pertaining to 'Bill of Material' and 'Inventory Status Record' must be accurate, complete, and up to date for its effective and useful operation,
- Lead times for all inventory items are known and can be supplied to the system,
- Each and every inventory item goes into and out of stock,
- All of the components of an assembly are needed at the time of the assembly order release,
- Discrete disbursement and usage of component materials, and
- Process independence of manufactured items.

1.2.2 Regenerative System

In this type of MRP system, the material requirements for all items or components are recomputed periodically, usually weekly, based on the latest master production schedule requirements. Here the MPS is broken down into time

phased requirements for every individual item.

Under these types of systems, every end item stated in the MPS must be exploded; every bill of material must be retrieved; and the status of every active inventory item must be recomputed. The main aim is to finish the explosion process in one run of the MRP system as the MPS is periodically updated. During this run, the gross and net requirements for each inventory item are recomputed and its planned order schedule is recreated in a level by level fashion, starting from the top level.

The frequency of replanning is a critical variable in the use and design of an MRP system, and in this regenerative type, it is impractical to replan at a frequency higher than once per week. The solution to more frequent replanning is the net change approach.

1.2.3 Net Change System

Here, only additions and subtractions from the master production schedule are entered, and the change in material requirements is then calculated, usually at the end of each day, for only those items/components affected. This concept views the MPS as one plan in continuous existence, rather than as successive versions of it. The principle of net change also affects the inventory status record data, and data in this record must be maintained up to date to effect the daily net change. The concepts of 'Record Balance', and 'Interlevel Equilibrium'⁽¹⁶⁾ of the inventory status data characterize a net change MRP system. The inventory status record is said to be in balance when the on hand quantities correspond to existing projected requirements and scheduled receipts, and when the planned order releases are correctly calculated with respect to both quantity and timing. The principle of 'Interlevel Equilibrium' states that projected requirements for every item must correspond at all times to the quantities and timing of planned order releases of its parent items.

1.3 SCOPE OF STUDY AND THESIS ORGANIZATION

The ultimate developed tool in materials planning is MRP, and therefore, through this dissertation, extensive study has been carried out to develop a computerized MRP system for these projects. The main objectives of the present

work are,

1. To develop an efficient and transparent Material Requirements Planning system by using Object-Oriented software tools so that it can be used in the Water Resources Projects, and
2. To develop a user friendly GUI for the application using graphics tools for operational ease of the user.

Material Requirements Planning system is the latest model used in most of the organizations for efficient and transparent control of the flow of materials from, to, and within the organization. But it is seen that the Water Resources Projects in general are using the primitive models for planning and controlling the flow of huge materials involved in the project sites.

Through this dissertation, research has been done to develop a software for a typical MRP system which can be used in the Water Resources Projects, because the cost of any commercial MRP software available in the market is sky high. The software is designed in the OOP language, C++ because OOP is today's method of choice for software design, and C++ is the language of choice, which is designed to support OOP.

This dissertation documents the utility and development of an 'Material Requirements Planning' System for the Water Resources Projects. It is divided into six chapters.

Chapter 1 includes introduction, definitions, objectives and the scope of study. Chapter 2 describes the review of literature related to this topic. Chapter 3 discusses the applications of Material Requirements Planning system to the Water Resources Projects. MRP is also compared with the primitive inventory control systems. Chapter 4 describes the design procedure and methodology of developing an Object-Oriented software for the Material Requirements Planning system to be used in Water Resources Projects. Chapter 5 covers the case studies undertaken in order to see how MRP can be applied to a Water Resources Project. Chapter 6 presents the conclusions followed by scope for further study.

Information supplemental to this dissertation is included in appendices. Appendix A includes the 'Software Listing'.

LITERATURE REVIEW

2.1 GENERAL

The bottom level of the balance sheet of an organization depends on many factors. One of a major contributory factor to it, is the cost of materials involved. Before 1960, the flow of materials to, within, and from any organization were planned and controlled by conventional inventory and production control systems like,

1. Deterministic Models,
2. Statistical Models,
3. Conventional aggregate inventory management systems, and
4. ABC or VED inventory classification system.

Joseph Orlicky, 1961⁽¹⁶⁾ first innovated the MRP system for J. I. Case Company, Inc., USA, based on file structures and records, where the item records are logically linked with each other, coupled with a huge computer program, that maintains these records up to date, and with the advent of high speed computers having huge data processing capacity, the use of the above mentioned conventional systems became obsolete in organizations.

During early 1980s, closed loop MRP systems emerged as a practical scheduling system⁽⁶⁾⁽¹²⁾⁽²⁴⁾. But this closed loop MRP system (Fig. 2.1) was applied in a range of situations for production of high to medium volume standard products. The key features of this MRP system are to generate lower-level requirements, time phasing requirements, planning of order releases, and rescheduling of orders to meet realistic schedules. The closed loop system was extended to include feedback from and control of customer orders.

Closed loop MRP systems were specifically designed to meet the needs of low volume, job shop organizations during late 1980s, by controlling the engineering design changes in the bill of materials, as in Simon's MRP system⁽⁶⁾⁽¹²⁾.

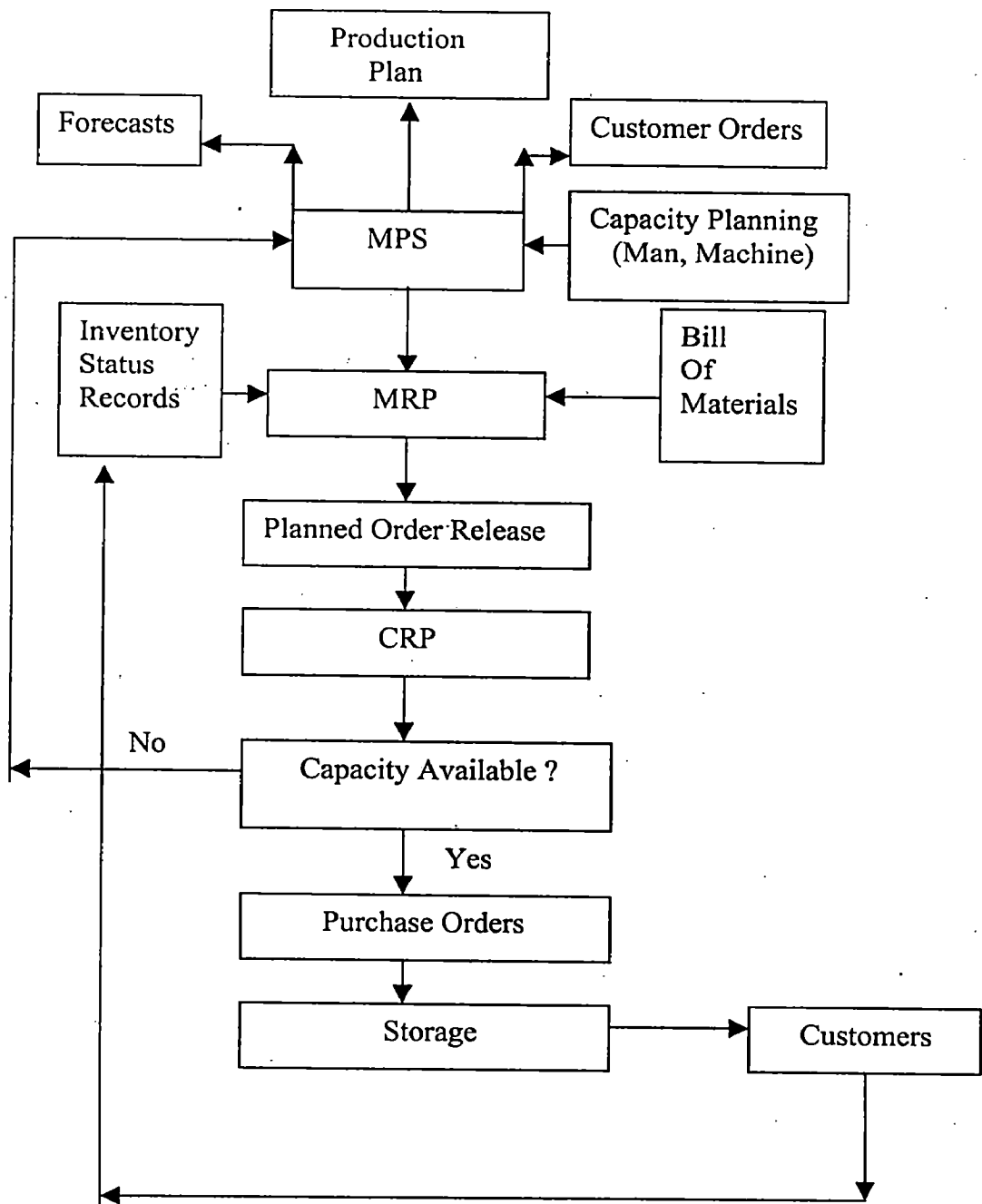


Fig. 2.1: Typical Closed Loop MRP System ⁽²⁴⁾

Timon Chih-Ting Du and Philip M. Wolfe, 2000⁽²⁷⁾ suggested to build an active, bucketless, and real time MRP system using hybrid architecture that includes an object-oriented database, fuzzy logic controllers, and neural networks, because an object-oriented database combines the advantages of object-oriented technologies and database technologies. The object-oriented technology provides: (1) superior complex data structure manipulation; (2) rich object structure (classes, types, hierarchy etc.); (3) potential object-oriented architecture; and (4) data reusability (inheritance mechanisms).

As per Lambrecht and Decalures^{'(27)} opinions, there are three reasons behind improper implementation of MRP systems at operational levels: (1) MRP ignores capacity constraints; (2) MRP cannot compete with the dynamism of the shop floor; and (3) safety stock or fixed lead time result in rigid implementation of the system.

In the MRP literature, the basics of Orlicky's MRP system remain unchanged, but MRP is applied in various situations with or without slight modifications suitable to the organization concerned.

2.2 MRP SYSTEM BASICS

The prime objective⁽¹⁶⁾ of an MRP system is to find gross and net requirements i.e. to determine discrete period demands for each item of inventory in order to generate information needed for correct inventory order action. The basic function of an MRP system is to compute net requirements from gross requirements so that the former should be covered by timely purchase orders or shop orders. The net requirements are then covered by the planned order releases, and the order quantities either match net requirements or are computed by using any one lot sizing techniques. The outputs of an MRP system can serve as valid inputs to other computer application systems in the production and inventory control, like purchasing systems, shop scheduling systems, dispatching systems, shop floor control systems, and capacity requirements planning systems.

2.2.1 System Inputs

The three basic inputs of an MRP system namely the 'Master Production Schedule', 'Bill of Materials', and 'Inventory Status Record' without which it cannot exist, are diagrammed in Fig. 2.2, and are discussed in the following paragraphs.

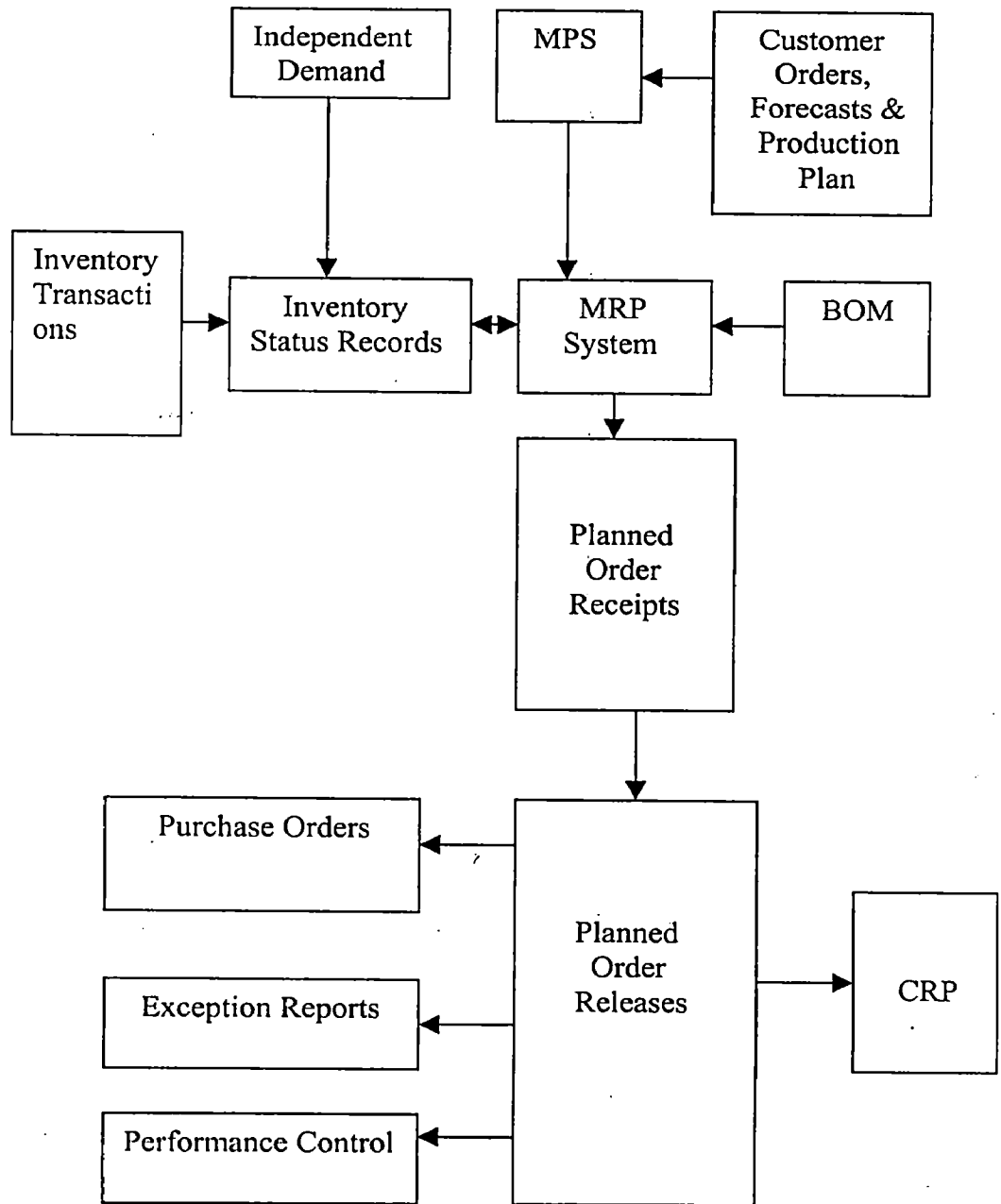


Fig. 2.2: Typical MRP System Inputs & Outputs

2.2.1.1 Master Production Schedule

This is stated in terms of end items, which may be either transportable products or highest level assemblies of some other products. The planning horizon to be covered by the MPS equals or exceeds the sum of procurement and manufacturing lead time for components of the products considered. MPS actually describes what materials we need, and when we need it.

MPS is a form of aggregate planning, which considers only finished products, but not the individual components that constitute those end products. The objective of the MPS is to develop an aggregate production schedule that will meet end item requirements, and at the same time minimize the incremental costs incurred. Some relevant costs may be the payroll costs, excess inventory costs, and cost of production rate changes. MPS as a total cost model ⁽⁸⁾ has been developed, where the above mentioned cost components vary with the changes in the decision variables of the model.

Here, MPS is represented by a single stage system (Fig. 2.3) in which the planning horizon is one period long. The state of the system at the end of the previous period is defined as: the aggregate work force size W_0 , the production rate P_0 , and the inventory level I_0 . The ending state conditions for one period become the initial conditions for the upcoming period. A forecast of the end item requirements for the upcoming period comes through some decision process that set the size of the work force and production rate for the upcoming period. So, the inventory at the ending state conditions will be⁽⁸⁾:

$$I_1 = I_0 + P_1 - F_1, \text{ where}$$

F_1 is forecasted sales,

I_1 is inventory level in the end of period one, and

P_1 is production rate in the end of period one.

The MPS statement of the quantity of each item required is put into the MRP system each week or more frequently. The MRP program references the inventory status record, and the bill of material corresponding to the item in the MPS, and thus material requirements are generated. The MPS actually drives the MRP system and, consequently, the entire production and inventory management system ⁽¹⁸⁾.

The MPS is derived from the aggregate production plan based on demand forecasts, customer orders, and capacity limitations. The MPS is divided into time periods called time buckets. Each product has its own MPS, and there is also an

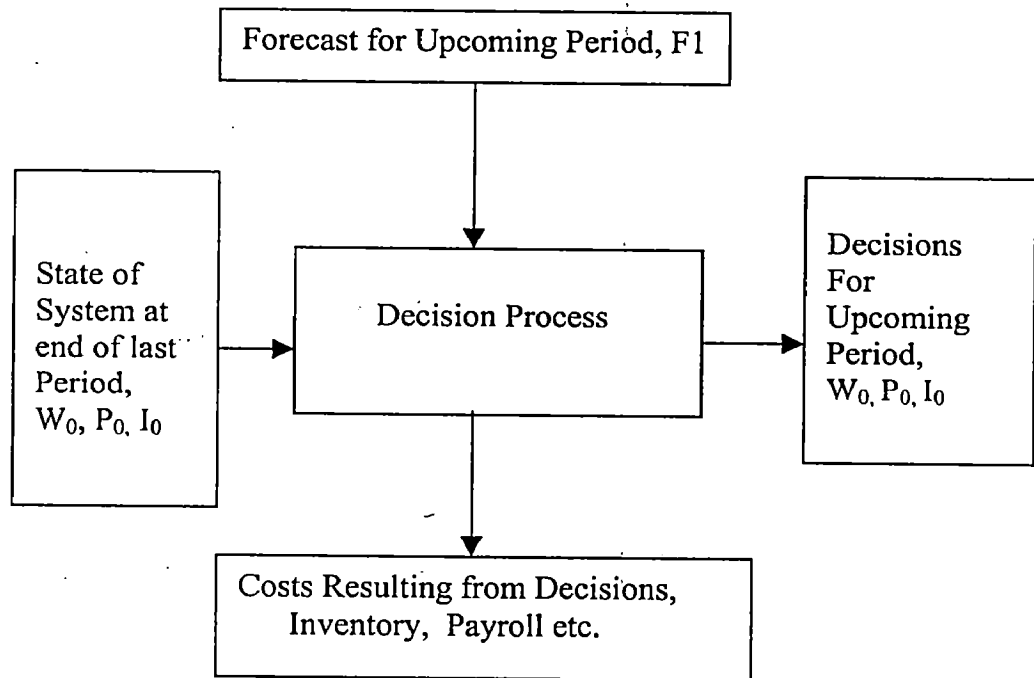


Fig. 2.3: Single Stage Planning Decision for a Horizon of One Period

overall MPS that synthesizes the requirements for all products or a group of products that share facilities ⁽¹⁵⁾.

The MRP parts explosion process assumes that the MPS is capacity feasible. In closed loop MRP systems, the orders generated from the parts explosion process are put into a capacity requirements planning routine to evaluate their feasibility. If sufficient capacity is not available, the MPS should be revised or capacity should be added until the MPS can be accomplished. The MPS must project a realistic plan of production that is leveled to accommodate available capacity ⁽²³⁾.

The MPS is developed from end item forecasts, but it is not exactly the same as forecast. MPS and forecast may differ because,

- (a) The forecast may exceed plant capacity,
- (b) Inventory level may increase or decrease, and
- (c) Organization concerned may decide to operate uniformly, using safety stock against demand fluctuations.

2.2.1.2 Bill of Materials

It contains the relational information on all materials, components, raw materials, subassemblies required to produce each end item. BOM is used to derive the quantities of dependent components required to produce end items. Moreover, the BOM is a structured list of dependent demand items which describes the sequence of steps in manufacturing the product ⁽²⁴⁾, and hence the BOM is alternately called 'Product Structure Tree'. These dependent demand items are placed in levels in the BOM, representing the way they are actually placed in the manufacturing process. The BOM should be updated as the products are redesigned, new products are added, and product sequencing or assembly is changed. Information on every component at every level of the BOM must include a unique part number for the item, an item description, the quantity used per assembly, the next higher assembly in the structure list, and the quantity used per end item. Each stage in the manufacturing process of converting material into product is equivalent to a level of 'Product Structure Tree'.

BOM is essential for the exact computation of gross and net requirements. While determining net requirements for a low level inventory item, we calculate the quantity that exists under its own identity; quantity of it, if it is behaving as a component item of any other parent item; and quantity of it, if it is one of the parents

of any other parent item. The computation of net requirements proceeded in the direction from top to bottom of the product structure tree in a level by level⁽¹⁶⁾ fashion. The net requirement on the component item level can be computed only after the net requirement on the parent item level is determined. The downward progression from one product level to another is called explosion, and the vice versa is called implosion. The former is to establish all lower level component scheduling, and the latter is required for identification of the parent item generating the requirement, when scheduling problem exist at the component level ⁽²⁴⁾.

Various formats or ways in which the BOM is displayed are:

(1) Single Level Explosion Format:

It displays the components used at a specific level of assembly (Form 2.1).

(2) Indented Explosion Format:

It lists components on all lower levels by indentation, signifying levels, under their respective parents, i.e. the products are displayed in the manner in which they are manufactured (Form 2.2).

(3) Summary Explosion Format:

This format lists all items which go into an end item along with their total quantities, i.e. it includes all components of an end item, irrespective of level. Summary explosion bill helps during purchasing of proper quantity of items (Form 2.3).

(4) Single Level Implosion Format:

This displays the assemblies that directly use a component at the next higher level, i.e. it indicates all the immediate parent items of an item (Form 2.4).

(5) Indented Implosion Format:

It indicates the usage of a component to all higher levels, and it is also valuable in finding the parent that generated the component requirement (Form 2.5).

(6) Summary Implosion Format:

This shows all higher level items which contain the component along with the total quantity used in each (Form 2.6).

(7) Matrix Bill Of Materials:

This format is used to identify and group the common parts found on the models in a family of items. This format is useful for products having many common components among products (Form 2.7).

Assembly	Component No.	Qty. per Assembly	Description
----------	---------------	-------------------	-------------

Form 2.1: Single Level Explosion Format

Part Number			Description	Quantity Per Assembly	Unit of Measure
Level 1	2	3			

Form 2.2: Indented Explosion Format

Assembly	Component and Subassemblies	Description	Quantity Required	Unit of Measure
----------	-----------------------------	-------------	-------------------	-----------------

Form 2.3: Summary Explosion Format

Part Number	Assembly Used On	Quantity Per Assembly	Description
-------------	------------------	-----------------------	-------------

Form 2.4: Single Level Implosion Format

Component Part Number	Assembly Used On	Quantity	Description
-----------------------	------------------	----------	-------------

Form 2.5: Indented Implosion Format

Component Part Number	Assembly Used On	Quantity Required	Description
-----------------------	------------------	-------------------	-------------

Form 2.6: Summary Implosion Format

(8) Comparative Bill Of Materials:

This bill defines a special product in terms of a standard product (especially new or unique products), and specifies which components are to be added, and which components are to be deleted (Form 2.8). This format cannot be used in making forecasts, and is not used in MRP system ⁽²⁴⁾.

(9) Modular Bill Of Materials:

These are used for complex products that have various configurations, and are made from a number of common parts. This BOM is stated in modules from which the final product is assembled (Form 2.9). The process of modularizing breaks down the bills of products into lower level modules. The concept of modularity ⁽²⁴⁾ has two purposes, (i) to separate combinations of optional product features by facilitating forecasting, and (ii) to isolate common parts from unique parts, by minimizing inventory investment in components common to optional units. Owing to modularity, BOM is abolished at the top of the 'Product Structure Tree', and lower level components are promoted to end item status for MRP purposes.

(10) M-Bill:

It is the collection of all the individual modular BOM that are selected by a customer, which states the options necessary to build a specific end item. This bill though not a direct part of the MRP system, can be integrated with the MRP system by defining the items to be assembled against the final assembly schedule ⁽²⁴⁾.

(11) S-Bill:

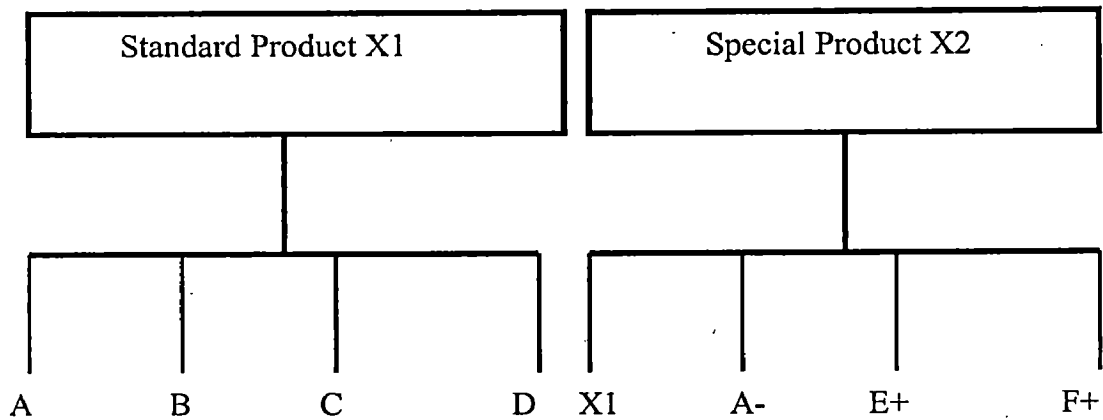
While modularizing the bills, many components become end items, which will increase the number of items to be forecasted and identified in the MPS. So, a set of components in any related group is assigned an imaginary part number to form a S-Bill (Form 2.10). These artificial numbers are not actually assembled, but are used in forecasting and MPS operations.

(12) K-Bill:

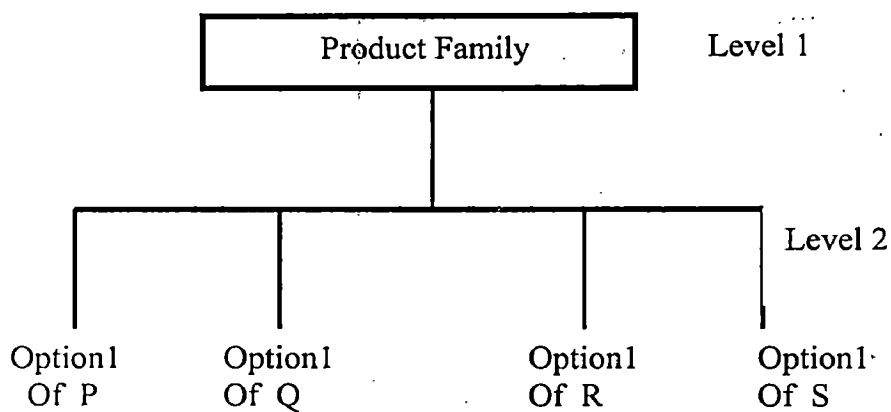
While building a product, a large number of loose parts are frequently used. Without identifying each of the items individually, they are grouped together under an imaginary part number called a kit ⁽²⁴⁾. So, with one number an entire group of components having similar features can be properly scheduled.

Component Part Number	Description	Unit Of Measure	Product		
			One	Two	Three

Form 2.7: Matrix Bill Of Materials



Form 2.8: Comparative Bill Of Materials



Form 2.9: Modular Bill Of Materials

2.2.1.3 Inventory Status Records

This record comprises the individual item inventory records containing the status data (on hand and on order) required to find the net requirements in a time phased format, by checking what inventory will be available to meet the production schedule. This record must be kept up to date by posting inventory transactions (stock receipt, issue, scrap etc.) accurately. These transactions update item status, which is then modified in the MRP computation process.

Inventory status record also contains the 'planning factors'⁽¹⁶⁾ like lead time, safety stock, lot sizing algorithms, scrap allowances, item description, demand history etc., and if one of these factors will change, then the item status will change.

The data in this record must be kept up to date and accurate otherwise the whole MRP system will go wayward.

2.2.2 Lot Sizing Techniques

The main purpose of this section is to know the methods of evaluating lot size for ordering materials within an MRP framework. Usually MRP has been implemented in a deterministic environment. The pattern of the demand in an MRP system may follow constant or variable distribution. The constant demand pattern implies an identical demand is needed over a known planning horizon. The variable demand pattern indicates demand variability over a planning horizon similar to some probabilistic distributions. The assumptions⁽¹⁶⁾ on which the lot sizing decisions are based, are as follows:

- (i) Demand is deterministic and time varying,
- (ii) The product structure is multi-level, and demand for a product in one level depends on the demand of a product on another level,
- (iii) The lead time is fixed and known,
- (iv) The replenishment rate is finite,
- (v) Shortages are not allowed, and
- (vi) Lot splitting is not allowed.

The lot sizing heuristics proposed in an MRP framework for both single level and multi-level (assembly type) systems are:

- ❑ Lot For Lot (LFL) ⁽²⁴⁾,
- ❑ Economic Order Quantity (EOQ) ⁽²⁴⁾,
- ❑ Modified Economic Order Quantity (EOQ₂) ⁽²⁰⁾,
- ❑ Periodic Order Quantity (POQ) ⁽¹⁶⁾,
- ❑ Least Unit Cost (LUC)⁽¹⁶⁾,
- ❑ Least Total Cost (LTC)⁽¹⁶⁾,
- ❑ Modified Least Cost (LTC₂)⁽¹⁶⁾,
- ❑ Part Period Algorithm (PPA)⁽¹⁶⁾,
- ❑ Silver-Meal Algorithm (SMA)⁽¹⁶⁾,
- ❑ Silver-Meal Algorithm II⁽²⁴⁾, and
- ❑ Groff-Marginal Algorithm (GMA)⁽¹⁶⁾.

In this dissertation, the first three lot sizing techniques are discussed.

(a) Lot For Lot Technique:

Here, a lot is created for each period in which there is demand. The holding cost remains constant, and for certain future demand, zero inventory is carried from one period to another. LFL technique is used under assumption that the ordering cost is low and the cost of inventory is high.

(b) Economic Order Quantity:

This technique was developed to minimize the cost for constant and continuous demand. The Wilson Camp formula for determination of the economic order quantity is,

$$EOQ = (2 * R * C / H)^{1/2}, \text{ where,}$$

R = average annual demand in units,

C = ordering cost per order, or set-up cost per batch,

H = inventory holding cost per unit per annum.

EOQ method is not optimal in MRP environment in the sense that the assumption of constant demand is not met in a discrete demand environment.

(c) Modified Economic Order Quantity:

Mitra et al. ⁽²⁰⁾ developed the modified economic order quantity formula for discrete demand situations with a view to balance the two opposing costs, i.e. inventory carrying cost and ordering / set-up cost. The EOQ₂ determines the

lot sizes such that no period has items produced in two lots. Mitra et al ⁽²⁰⁾ adjusted the economic order quantity (EOQ) to cover demand during an integral number of periods, to find EOQ_2 . Here demand is accumulated until cumulative demand approaches EOQ.

The modified economic order quantity is denoted as,

$$Q_1 = \sum_{i=k}^n di, \text{ where } Q_1 \text{ exceeds EOQ, and}$$

$$Q_2 = \sum_{i=k}^{n-1} di, \text{ where } Q_2 \text{ is less than EOQ, where,}$$

Q_1 and Q_2 are order quantities,

k = Time period in which the order quantity is going to be determined, i.e. the starting point,

di = Demand in period i , and

n = Total number of periods.

Then the modified order quantity is either Q_1 or Q_2 , depending on whichever is nearer to the EOQ. If EOQ is exactly half way between Q_1 and Q_2 , then the order quantity is chosen as Q_2 .

2.2.3 Time Phased Order Point Technique

It is an approach which allows the 'time phasing' concept of the MRP system to be used for planning and controlling independent demand items. The demand for such items have to be forecasted, and their supply would be controlled by means of order points.

Actually no time phased order point is computed by the MRP system, only the replenishment order is issued by the system by lead time offset, when the quantity on hand goes down below the safety stock, i.e. the system deals with the independent demand items exactly the same way it treats any item for which safety stock is specified.

2.2.4 Role of Safety Stock

While the primary function of safety stock is protection against the uncertainty of demand, i.e. forecast error, a secondary function is to compensate for uncertainty of supply. When safety stock is planned at the item level, the MRP system attempts to

conserve its quantity and to protect it from being used up, so that this quantity should always be on hand. By the way, safety stock is carried by the system and never used in the whole process, and create undesirable overstated material requirements.

Component-item demand is certain with respect to the master production schedule, and so safety stock has no place in the estimate of material requirements planning of it.

So, if required, safety stock can enter the MRP system through the MPS for independent demand end item requirements planning, and planning and carrying safety stock should be limited to the purchased items only.

Mather (1977)⁽²⁾ has recommended the use of safety stock in conjunction with firm planned orders. Bannerjee (1979)⁽²⁾ indicated that safety stock do not produce significant results under different scheduling policies (lot sizing). Candace Yano and Robert Carlson (1987)⁽²⁾ indicated that it is economical to use safety stock as protection against demand variation, along with infrequent rescheduling.

However, there is a production priority problem with excessive safety stock even if it is admissible. Excessive work-in-process component inventory destroys priorities through increased queue time and can cause missed schedules ⁽¹⁶⁾⁽¹⁸⁾. In short, MRP system should have neither 'too little' nor 'too much' safety stock.

2.2.5 System Outputs

The outputs of a typical MRP system, which were depicted in Fig. 2.2, are discussed in the following paragraphs.

2.2.5.1 Planned Order Receipts

In MRP computation process, the gross material requirements are converted to net requirements by subtracting the available inventory from it, which is called 'netting'. Then the actual order quantity for an item is matched to a suitable lot size by using appropriate lot sizing algorithm, or it may be equal to the net requirement quantity. This order quantity is known as 'Planned Order Receipt'⁽¹⁶⁾⁽²⁴⁾, i.e. new orders for the particular item scheduled for future release. The detail flow diagram depicting the MRP computation process for a typical system is shown in Fig. 2.4.

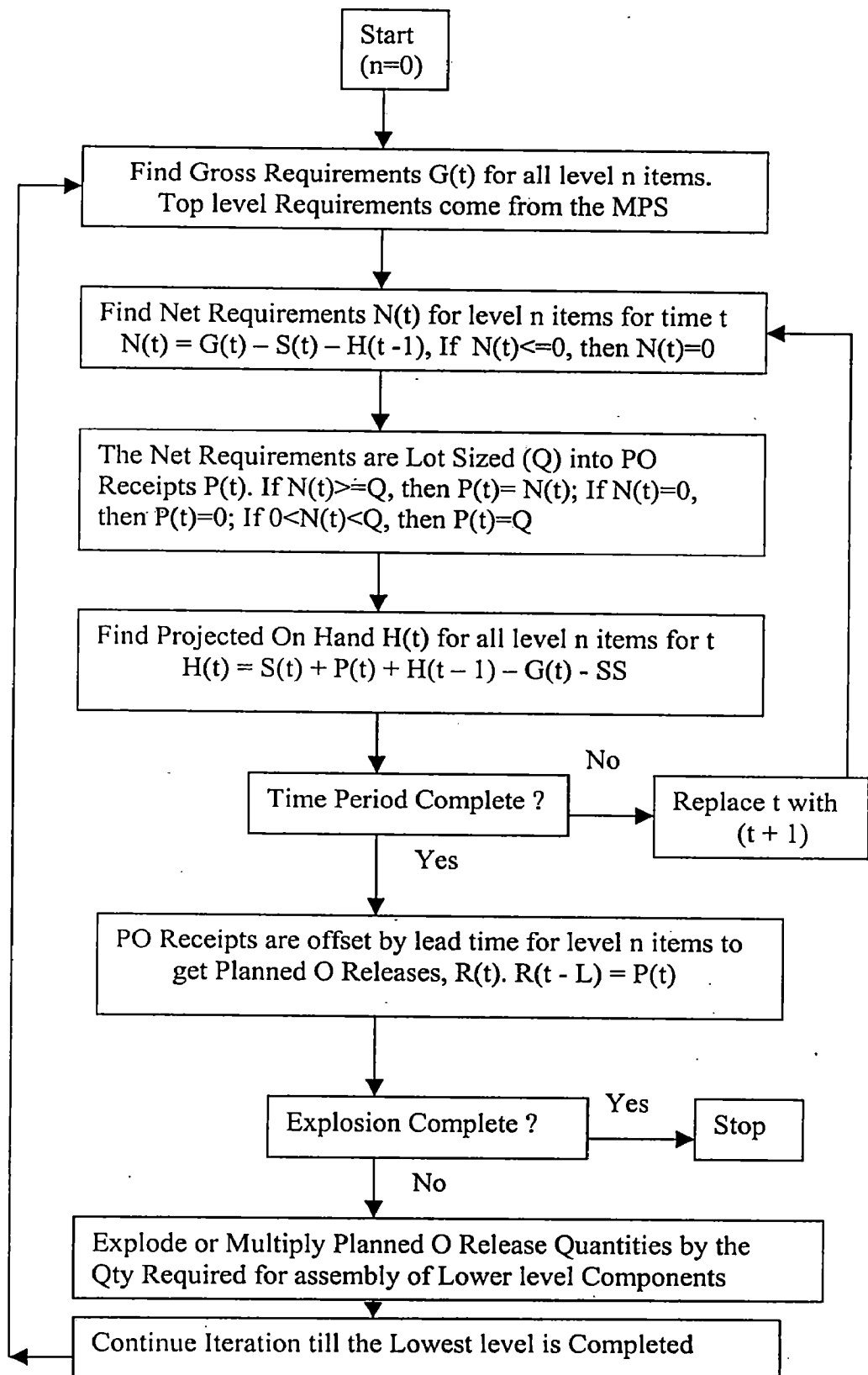


Fig. 2.4: Flow Chart of a Standard MRP System ⁽²⁴⁾

2.2.5.2 Planned Order Releases

The timing of planned orders is based on scheduling order releases for the purchase or the manufacture of component parts in a sequence which promises their availability in tune to the total production process. Usually these orders are planned much before to permit sufficient time for scheduled completion of the final product without unnecessary queue of the materials. MRP time phases these orders by offsetting the lead time (expressed in shop calendar units), i.e. by subtracting the lead time value from the shop calendar date of 'Planned Order Receipt' or order completion.

In a shop calendar, weeks are usually given two-digit designations (from 00 to 99), and working days three-digit designations (from 000 to 999) ⁽¹⁶⁾, and one week is equal to five working days.

For purchasing components, the lead time is the time interval between the placement of the purchase order and its entry into the inventory system, and for manufacturing items, it is the interval between the release of the work order and its completion. In an MRP environment, the planned lead time may constitute the operations such as queue (waiting to be worked on) time, running (machining, fabrication, etc.) time, set-up time, queue (waiting for transportation) time, inspection time, and move time. The formula to calculate the planned lead time for MRP purposes is,

$$L = 2 * N + 6, \text{ where,}$$

L = Lead time in working days, and

N = Number of Operations.

Planned lead time is sometimes added with 'safety lead time' or 'safety time' for completion of an order in advance of its real date of need, and here the MRP system will plan both the order release and order completion dates, and the respective scheduled receipt period would be moved back by the 'safety time'. This 'safety time' is similar to 'safety stock' in compensating the demand fluctuations. In actual practice, this 'safety time' will create unnecessary work-in-process inventory ⁽¹⁶⁾.

Usually, all components to an assembly are planned to be available before the start date, and so the 'Planned Order Receipt' is moved back by the lead time to output the 'Planned Order Release', and usually the output is shown in a typical horizontal format as shown in Form 2.10.

Item Code:	<u>MRP MATRIX</u>					Date:----/-----/---
Lead Time:						
Safety Stock:	Weeks					
	PD	1	2	3	4	5
Projected Requirements:						
Scheduled Receipts:						
Quantity On Hand:						
Net Requirements:						
Planned Order Receipts:						
Planned Order Releases:						

Form 2.10: Typical MRP Matrix

These 'Planned Order Releases' provide the quantity of the item and the relevant time period when the work orders are to be released to the workshop or purchase orders placed with the suppliers. When this order is released, it changes from being 'planned' or 'open' to being 'scheduled' or 'on order'.

The primary purposes of 'Planned Order Release' are to generate material requirements at the next lower level, and to project capacity requirements ⁽²⁴⁾.

2.2.5.3 Exception Reporting

Outputs from an MRP system can be in the form of user's choice like reports, or individual messages displayed in the computer screen. MRP system can keep order priorities up to date by planning and replanning order due dates. It attempts to make the due date and need date coincide at the time of planned order release, by computing and recomputing net requirements, such that the operations proceed as planned while inventory investment is minimized. The due date is defined as the present date associated with the order, i.e. the expected date of order completion, and the need date signifies the date when the order is actually needed ⁽¹⁶⁾. MRP is a priority planning and controlling system, and it gives importance to the valid open-order due dates, which establishes the relative priority of the current order. The various exception reports and messages produced from the MRP system can be like:

- Projected or gross requirement input is beyond the planning horizon,
- Number of digits in the projected requirement quantity field exceeds limit,
- 'Planned Order Receipt' moved back by the lead time but misplaced,
- Number of digits in the 'Planned Order Receipt' quantity exceeds limit,
- Number of digits in the net requirement quantity field exceeds limit,
- Number of digits in the 'Planned Order Release' quantity exceeds limit,
- Number of digits in 'Scheduled Receipt' quantity overflows to the 'On Hand Quantity' field,
- Due date of the planned order is outside the planning horizon,
- Past-due projected requirement quantity has been included in the current period,
- Component / item part number does not exist,
- Actual receipt quantity exceeds 'Scheduled Receipt' quantity,
- Order quantity released exceeds the 'Planned Order Release' quantity,
- Delay, or expedite, or cancel an order,

- Mismatch of timing between demand and supply, and
- Launch a new order.

MRP system will simply print messages specifying exactly where the changes are needed but the final decision to effect changes depends on the management people.

2.2.5.4 Performance Control

Management monitors the performance of inventory planners, vendors, the workshop, buyers, and financial matters of the organization, by reviewing the outputs of an MRP system.

If the inventory data are stored along with the cost component, the on hand quantity plus the planned order receipt quantity are costed out, and summarized by item group to get the inventory investment level forecasts.

Open purchase orders recorded under valid due date can also be converted to a purchase commitment report in a similar fashion.

2.3 OBJECT-ORIENTED SOFTWARE PARADIGMS

The concept of an object is central to anything object-oriented. Object-Oriented Programming (OOP) language supports object-oriented software design methods. OOP is an implementation method in which programs are organized as cooperative collections of objects, and each of that object represents an instance of some class, and whose classes are all members of a hierarchy of classes united through inheritance relationships⁽⁹⁾. Object-Oriented Design (OOD) is to refer to any method that leads to object-oriented decomposition. For everything object-oriented, the conceptual framework is the 'Object Model'⁽¹⁾.

Materials managers should react quickly and effectively to the changing scenarios of the activities going on inside or outside their organizations. These changes are driven by an increasingly competitive global economy that forces acquisition of new equipment and the introduction of new or improved processes. Embracing these changes can be difficult for materials managers if they rely strictly on the traditional systems. Systems which are flexible enough to allow organizations to quickly react to the changes occurring in their enterprises need to be developed. Not only must these tools be flexible, but also they must have sufficient clarity of documentation that the mechanics of making these changes is relatively simple to

perform. Object-oriented technology has been proven to have the right 'texture'⁽⁴⁾ in this regard. Systems designed based on object-oriented concepts are modular, reusable, and less costly to maintain. The elements of the 'Object Model' are,

- (a) Abstraction,
- (b) Encapsulation,
- (c) Modularity,
- (d) Hierarchy,
- (e) Typing,
- (f) Concurrency, and
- (g) Persistence.

2.3.1 Abstraction

It is the essential characteristics of an object that distinguish it from all other kinds of objects, and so provide finely defined conceptual boundaries, relative to the user⁽⁹⁾. An abstraction shows the outside view of the object, and as such separate the object's observable behavior from its implementation. In object-oriented design, the main problem is to decide the correct set of abstractions for a given domain. The abstractions are responsible for the preliminary design decisions in designing an 'Object Model', and they should precede the decisions about their implementations.

2.3.2 Encapsulation

Encapsulation is the process of dividing the elements of an abstraction that constitute its structure and behavior⁽¹⁾⁽⁹⁾. Thus encapsulation clearly separates the interface of an abstraction and its implementation. Encapsulation is achieved through 'data hiding', which is the process of hiding the structure of an object, and the implementation of its functions.

2.3.3 Modularity

In OOP languages, classes and objects form the logical structure of a system, and these abstractions are placed in modules to develop the physical architecture of the system. In other words, modules are the physical containers, in which the logical design of the classes and objects are declared. So, modularity can be defined as the property of a system that has been decomposed into a set of cohesive and loosely coupled modules or compartments⁽⁹⁾.

2.3.4 Hierarchy

Simply, hierarchy is an ordering of abstractions. In a complex system, there are two types of hierarchies,

- (a) “is a”⁽¹⁾ hierarchy represents the class structure of the system, and the suitable proposition of this type of hierarchy is inheritance, because inheritance defines the relationship among classes, and
- (b) “part of”⁽⁹⁾ hierarchy represents the object structure of the system, which is built upon the concept of level of abstraction.

2.3.5 Typing

Abstractions are expressed by typing so that the programming language in which these abstractions are implemented can be made to enforce design decisions. Strong typing prevents mixing abstractions. Typing can be defined as the enforcement of the class of an object, such that objects of different types may not be interchanged.

2.3.6 Concurrency

Every computer program has at least one ‘thread of control’⁽²⁶⁾, i.e. a single process responsible for occurrence of independent dynamic actions within the system. But a system involving concurrency may have many such threads. Concurrency is a property, which distinguishes active and inactive objects, because in object-oriented design, each object represents one ‘thread of control’.

2.3.7 Persistence

Every object in a software system takes up some space and it survives for a specified period of time. Persistence can be defined as the property of an object through which the object continues to exist after its creator ceases to exist, and/or the location of the object moves from the address space in which it was created⁽⁹⁾⁽²⁶⁾.

APPLICATIONS OF MRP TO WATER RESOURCES PROJECTS

3.1 GENERAL

The construction works of any Water Resources Project is highly time consuming, and the materials required for the purpose of the construction varies depending on the type of the project. Usually before start of any construction work, complete survey is done, construction schedule is prepared, and effort is made to fulfill that schedule. This schedule is prepared year wise and actual work is performed month wise by dividing each year to twelve months. The requirements of the materials will depend on the actual work to be done, and as per the estimates prepared.

3.2 MRP SYSTEM FOR WATER RESOURCES PROJECTS

For a typical Water Resources Project, the construction works may be divided in a year wise fashion as follows:

Year	Construction Works
1 st Year	Preparatory Works; Purchase of Construction Plant and Equipment; Construction of Machine shop, Weld shop, Paint shop; Storing Facilities for various materials like, Cement, Steel, Fuel, Lub. Oil, other spare parts of Machineries, and Heavy equipment; River/Canal Diversion Works, and its Concreting Works; Spillway Excavation, and its Concreting Works; Intake Excavation, and its Concreting Works.

2 nd Year	River/Canal Diversion Works; Construction of Cofferdams; Main Dam foundation Excavation, and Grouting Works; Intake Excavation, and Concreting Works; Power House Excavation, and Concreting Works.
3 rd Year	Dam Embankment Works; Dam Foundation Treatment; Intake Concreting Works; Penstock Concreting Works; Power House Substructure and Superstructure Works; Manufacturing of Gates; Installation of Gates, and Valves for Intake and Spillway; Penstock Pipe Installation; EOT Installation.
4 th Year	Installation of Hydro-Generators; Installation of Turbines; Installation of Control Equipment; Installation of Sub-Station, and Transmission Lines.

The construction plant and equipment to be used in the Project may be:

1. Stone Crushing Plant,
2. Batching and Mixing Plant,
3. Cooling Plant for cement,
4. Aggregate Processing Plant,
5. Cement Handling Plant,
6. Tyre Retreading Plant,
7. Belt-Conveyor System, and
8. Saw Mill.

Heavy equipment related to the project may be the Dozers, Shovels, Dump Trucks, Motor Graders, Cranes, Air Compressors, Drilling Machines, Jack Hammers, Tunneling Equipment, Road Rollers, Sheep Foot Rollers, Tractor Trailers, Water Pumps, Grouting Equipment, Concrete Pumps, Fuel and Water Tankers.

The machineries required for workshop and manufacturing of gates may be the lathes, shapers, drilling machines, milling machines, plate bending machines, welding machines, planing machines, grinding machines, air compressors etc.

So, a variety of materials are required in a Water Resources Project, i.e. from independent demand items to mixed demand items. In this context, Material Requirements Planning system comes into play for efficient planning and controlling

of materials in the project because sometimes it so happen that the scheduled completion date of these projects are delayed and there is cost overrun due to defective planning and controlling of materials.

The capacity of each of the construction plant and equipment, and heavy equipment depends on the type of the Water Resources Project. Their capacity is determined before the construction works actually starts. Depending on the capacities of the Plants, the material requirements for processing can be adjudged.

The principal inventories that constitute the materials management department of a Water Resources Project can be broadly listed under different demand item categories as follows:

- (i) Independent Demand Items
Cement, Steel, Explosives, Tires, Fuel and Lubricating Oil, Sand, Quarry Stones, Aggregates for Concrete etc.,
- (ii) Dependent Demand Items
Steel plates, girders, gears, pulleys, motors, chains, ropes, bearings, welding rods, oxygen and acetylene gas, non-destructive testing materials, rivets, nuts and bolts, brackets, arms, etc. for manufacturing of gates along with the hoisting mechanisms; Cement, sand, and aggregates in Batching and Mixing Plant; Timber, saw blades, gum, and spirit in Saw Mill, and Used tires, raw rubber, and coal in Tyre Retreading Plant,
- (iii) Mixed Demand Items
Under this category, the spare parts and accessories for different heavy equipment, machinery, and plant can be placed.

In order to apply the MRP system to any Water Resources Project, the following criteria must be applied,

- To fix the planning horizon:

The minimum planning horizon should be equal to or more than the cumulative or stacked lead time. If it is less than that, then MRP will be unable to issue 'Planned Order Release' for the items in the lower levels of the product tree, and consequently, the issue of work orders and purchase

orders will be too late. This loss of horizon occurs in the lower levels because the product structure is usually multileveled and the lead time is successively offset. The visibility to the future reduces as the MRP system proceeds from one level to another.

Owing to shorter horizons, some lot sizing techniques cannot be applied to the lower level components because of the unavailability of the net requirements quantity data.

So, for a typical Water Resources Project, the minimum planning horizon can be fixed as four weeks or one month.

- To fix the size of the time bucket in the system:

The fixation of the time buckets is balanced between the 'accurate timing' of the planning process, and the 'huge cost' involved in managing large system database. A time bucket size of one week is reasonably good enough for efficient issue of 'Planned Order Release' and computation of lot sizes.

For any project, the materials manager can fix one week as the 'time bucket'.

- To classify the inventories:

ABC analysis is a basic analytical management tool which enables the top management to classify inventories in order to reap the highest benefits. This technique tries to analyze the importance of distribution of items by money value in order to find its priority. This classification system depends not only on the cost value per unit, but also on the number of units consumed over a yearly planning horizon. So if the cost of an item is increased or the quantity of annual consumption is increased, then the annual consumption of the item in money value will increase, and as such the class of that item may be upgraded from lower to upper class, i.e. the lower class items are volatile.

In this ABC classification system, 20% of the total items ("A" class) constitute nearly 80% of the total inventory cost, 30% of the total items ("B" class) constitute 15% of the total inventory cost, and rest 50% items ("C" class) account for 5% of the total inventory cost. The prime purpose of this technique is to review the upper class items more frequently, and order small quantities of them than the lower class items in order to keep the inventory investment lower.

Materials involved in the Water Resources Project can be classified according to this ABC analysis technique, and if desired, only the upper class items may be given treatment in the MRP system. But MRP has the ability to process any item regardless of its classification.

In order to get the highest benefits from the MRP system, all materials must be given equal treatment, because as MRP is a priority planning system, all materials must be kept under its net to establish relative shop priorities in a production process. To exemplify, sometimes the validity of a "C" class item controlled by some other inventory policy is questionable if excluded from the MRP system.

So, all the materials of a Water Resources Project should be given equal treatment in the MRP system.

- To decide the frequency of replanning:

The frequency of replanning by an MRP system affects the system performance, and this frequency if exceeds one week becomes impractical. For frequently changing environment, the frequency of replanning can be increased, and it can be once a day or when the user wants. This replanning process can be either cyclic or continuous. In continuous replanning, the validity of the inventory status deteriorates.

If the replanning is once a day or it is a net change MRP system, then the system must be informed minute by minute about the inventory status data. In such dynamic situations, the lot sizing technique used may be unstable, and the lot sizes may move upward or downward as the forecast signals change in response to irregular demand. Due to this frequent replanning, the system shows its reaction, and may become 'Nervous', and may initiate recomputation of MRP outputs.

- To peg the requirements:

MRP computation process proceeds from top to bottom of the 'Product Tree' level by level. To identify the parent item that generated the dependent item (component), it is traced upward through the 'Product Tree', and this process is known as "Pegging". 'Single Level Pegging' locates the immediate parents, and 'Full Pegging' locates the end items that generated the component requirement. By following the 'pegs' from one item record to

another, the ultimate source to the component demand can be traced to its specified 'time bucket' in the MPS.

If a component is delayed, then it is possible to find the impact on the delivery of the end item to the customer through 'Pegging'. Usually 'Full Pegging' is required in limited situations, and so its use is remote.

■ To issue the 'Firm Planned Order':

This is the ability of the MRP system to solidify the quantity and / or the timing of a 'Planned Order Release' in a particular time bucket, in order to adjust the coverage of the net requirements, because as the scheduling process goes, these net requirements change, and over a period of time, they tend to change the 'Planned Order Release' quantity before orders actually mature for release.

This 'Firm Planned Order' does not allow the normal MRP system to compute net requirements from the gross requirements, and also not allow the lead time offset to take place in order to expedite a past due order, compress lead time, and lot size change.

This capability of the system should be used for selected situations only depending on the need.

3.3 SYSTEM CONTROLS

The materials manager of a Water Resources Project is responsible for the planning and controlling the inventory items in an MRP environment, by continuously interacting with the system. He should take inventory order action based on the outputs supplied by the system. He is the best person who can control the MRP system, and thereby, plan the flow of inventories in the Project. He should be able to:

1. Issue 'Planned Order Release' in the right quantity at the right time,
2. Place 'Purchase Orders',
3. Change or cancel quantity of orders,
4. Change the timing or reschedule the open shop orders to make it match with the date of actual need,
5. Request to reschedule the open purchase orders,
6. Handle items affected by engineering changes,

7. Monitor inventory for obsolescence and inactivity,
8. Locate and correct inventory status record errors,
9. Request to change the MPS if required,
10. Take punitive measures if found discrepancies between item requirements and coverage, and
11. Act to count the inventory physically.

3.4 COMPARISON WITH CURRENT INVENTORY MODELS USED IN W.R. PROJECTS

Material Requirements Planning system has numerous advantages over other inventory planning and controlling systems. In the present days, the Water Resources Projects in India are using either the 'Deterministic Models' or the 'Statistical Models' for planning and controlling inventories in the Project. We have thrown some lights here on the MRP system with a view to entering these Water Resources Projects with an inventory planning and controlling tool named the 'MRP'.

3.4.1 Limitations of Fixed Order Size System

'Fixed Order Size' systems under the roof of the deterministic inventory models do not take any risks or uncertainties in their design and use. The limitations of these 'Fixed Order Size' systems are:

- (a) Item demand is known, uniform, and continuous,
- (b) Production rate is known, uniform, and continuous,
- (c) Lead time is known and constant,
- (d) Order or set-up cost is known and constant,
- (e) Holding cost is known, constant, and linear,
- (f) No resource (money and space) limitations,
- (g) Infinite stock-out,
- (h) Inventory analysis cost is neglected,
- (i) Requires a huge inventory investment,
- (j) Unreliable with varying demand pattern situations,
- (k) Requires a huge investment in safety stock,
- (l) Requires forecasts for all items,
- (m) This system is based on past demand data, and hence not up to date, and
- (n) Materials covered under this system likely to be obsolete very fast.

3.4.2 Comparison With EOQ Model

The use of 'Economic Order Quantity' inventory policy can cause serious problems particularly when the demand is dependent and varying in nature, because demand should not be forecasted when calculated, and demand for the dependent demand items should be calculated from the BOM, whereas, demand for the independent demand items must be forecasted. Use of MRP will substantially reduce inventory investment in dependent demand items while improving operational efficiency by eliminating the risk of shortages associated with the 'EOQ' system, because it is better to order components from product / item requirements and to compel the component inventory to zero level between requirements. A comparison table is prepared for comparing 'EOQ' and 'MRP', and shown in Table 3.1.

Table 3.1:EOQ and MRP Comparison

'EOQ' System	'MRP' System
<ol style="list-style-type: none"> 1. Every inventory item is included, 2. Used for Independent demand, 3. Used for continuous item demand, 4. Demand pattern is random, 5. Continuous lead time demand, 6. Orders based on reorder point, 7. Forecasts depend on past demand pattern, 8. Forecasting for all items, 9. Here only quantity component is calculated, 10. Safety stock is allocated for all items 	<p>Product / Component Oriented, Used for Dependent demand, Used for discrete/lumpy demand, Calculated discrete demand pattern, Zero lead time demand, Orders based on 'time phasing' rule, System computes future production quantity and its timing, Items only in MPS are forecasted, Both quantity and time components are computed, Safety stock for end items only,</p>

3.5 SYSTEM INTEGRITY

For effective and successful performance of the MRP system, the data pertaining to inventory status record and bill of materials must be accurate, complete, and up to date. MRP system presupposes that all inventory items under its arm are

going into and out of stock. So, if the inputted data are faulty, then MRP will produce faulty outputs.

Also the validity of all data generated by an MRP system is relative to the contents of the MPS. So, if the MPS does not reflect actually what will be produced, then the order priorities derived from it by the system will not be realistic.

Otherwise, there may be 'priority validity' (i.e. order due dates may match with actual date of need) in the system but due to faulty data in the MPS, the 'priority integrity' is questionable in a priority planning system like MRP.

Therefore, it is very important to have cooperation, and trust between all the personnel involved in both the production control and inventory control of the organization in order to get the real benefits of the MRP system.

3.6 TRAINING OF PROJECT PROFESSIONALS

For a system to be successful, the users must be involved and they must be educated or in other words, the users must understand the system for its successful and effective implementation. Therefore, the users of an MRP system must have thorough training before they actually start working with the system.

Some selected professionals of the concerned Water Resources Project must take at least one month training on the MRP system to actually understand and successfully run the system such that the inventory planning and controlling will be easy, thorough, and efficient.

DESIGN OF MRP SOFTWARE

4.1 DESIGN BASICS

This software design is based on 'Object Model'. Generally speaking, Object-Oriented models view the world as a collection of objects that contain both their data and their functions, and these objects communicate with each other using their member functions. The present software is designed for MS-DOS operating environment, where the screen display is of textual form. The software can be run by the user to create and fill up records of various types, and save the data in 'File' for later retrieval either to read, or write. The various types of records the software is able to create and display are:

1. Master Production Schedule,
2. Bill of Materials,
3. Inventory Status Records,
4. Purchase Order Receipts,
5. Planned Order Releases,
6. Net-Change Reports,
7. Regenerative Reports, and
8. Exception Notices.

The MRP system is developed and designed by using OOP language, C++ because OOP is a new way of organizing software that is based on real-world objects, and C++ supports it.

The system will take inputs through the records such as,

- Master Production Schedule,
- Bill of Materials, and
- Inventory Status Records,

whereas, it will produce the screen output through,

- Planned Order Receipts,
- Planned Order Releases, and
- Exception Reports.

A user friendly Graphical User Interface is provided in the software for handling of the data and the software itself, and this GUI is designed in C++ graphics tools through which, the user can enter, alter, delete, save, and retrieve the inventory data, and as such efficiently handle the software system.

4.2 METHODOLOGY AND ARCHITECTURE

In this dissertation, the design methodology adapted is “top-down hierarchy of functions controlling the system that uses object-oriented modules for its functionality”⁽²⁶⁾. Here the system is designed by using a top-down approach but the modules are implemented using a set of interacting objects.

In the present case, the ‘MVC Architecture’⁽¹⁾ (Fig. 4.1) is implemented for the design of the software, where,

- (a) The model (M) layer implements the functionality part of the application,
- (b) The view (V) layer implements the graphical user interface (textual representation), and
- (c) The controller (C) layer implements the user interactions with the application.

In applying the MVC architecture, the responsibilities of various objects in the system are separated. The details of the application are hidden from the user interface and also the user interface is broken down into two parts, with the presentation handled by V and the interaction by C.

4.3 DESIGN PROCESS

The software is designed keeping in mind the following points,

- The software lets the user open a screen, fill it (partially or fully), and save the record in a disk file,
- The data are stored in a file and the user can retrieve them, and modify them, using a code number that is specific to each material used,
- The storage scheme of the software maintains only one blank record at a time but allows many instances of the filled in data for each record,

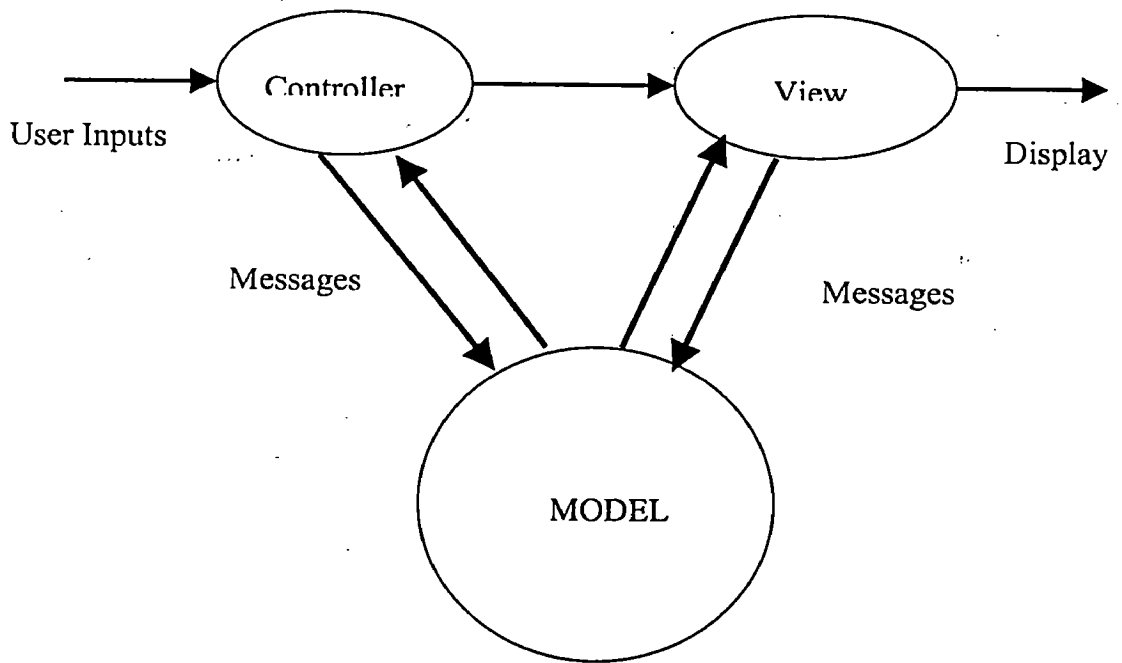


Fig. 4.1: MVC Architecture ⁽¹⁾

- The code numbers are stored in one file and the data in one separate file. In this way, there will be a single copy of each blank record at a time but multiple sets of data for each blank record to fill up the record,
- The code numbers are stored in 'Material.dat' file and the data are stored in 'Initial.dat' file,
- The material code numbers are automatically generated from the system, and if it is the first code number, then the listed code number will be 1(one), and here also there is a validation checking for entering the name of the material, which should not be more than a specified length,
- The software stores the system date as its own date automatically, and
- During data modification process, one record is replaced and the new one is overwritten at the same place.

4.4 OPERATING ENVIRONMENT

The software is designed for MS-DOS operating environment, with a text editor having either VGA or EGA graphics display adapter. The user must have Borland C++ 2.1(or higher) compiler, minimum 8 MB of RAM, and 30 MB of hard disk space in his system.

4.5 LIMITATIONS OF THE SOFTWARE

The various limitations of the software are,

1. It can generate and store 65534 code numbers for equivalent number of materials,
2. MRP computations can be made for two levels of items of the 'Product Tree',
3. Here, the time periods or buckets have been taken as weeks, and the planning horizon is up to a maximum of five weeks,
4. The maximum lead time assumed is three weeks,
5. The name of every material entered cannot exceed 20 characters,
6. The lot size of each material has been calculated by the standard EOQ formula,
7. The software has been designed for both 'lot for lot' and 'fixed lot size' ordering techniques.

CASE STUDIES

5.1 GENERAL

For the purpose of case studies which reveal how MRP can be applied to any typical Water Resources Project, we have chosen one independent demand item having zero components, and another independent demand end item with one dependent demand component. Due to unavailability of actual data, theoretical data have been taken, and many parameters have been assumed.

For the first case, the material selected is HSD oil. CI engines consume HSD oil; and the light vehicles, heavy vehicles, and the equipment used in the Water Resources Projects, which are invariably CI engine driven, intake this HSD oil as fuel.

In the second case, we have chosen 'Horizontal Girder' as an end item fabricated from rolled steel beams. Further, these 'Horizontal Girders' are used as components in the manufacturing process of different types of gates. Gates are hydromechanical control equipment and their purpose is to control the flow of water in the Water Resources Projects. In my working place, these gates are designed by the Department of Water Resources, Government of Orissa, Bhubaneswar, and manufactured by the Orissa Construction Corporation Limited, Bhubaneswar, a subsidiary of Department of Water Resources, Government of Orissa, Bhubaneswar.

5.2 CASE 1

5.2.1 Data Acquisition and Computations

The demand rate of HSD oil (considered for only one year of operation of the project) are usually calculated based on the construction schedule, and using the following formula:

Hourly demand / BHP = 0.15 * LF litres, where,

LF = Load Factor.

In actual working conditions, an equipment may not run over and above 60% load factor.

Specimen Demand Calculation for a 100 HP Bulldozer:

Brake Horse Power (BHP) = 100hp,

Load Factor = 50%,

Monthly working hours = 200 hrs,

Monthly HSD oil consumption per Bulldozer = $0.15 * 0.50 * 100 * 200$
= 1500 litres,

Total machine month = 20 (assumed)

So, total demand = $20 * 1500 = 30,000$ litres.

Monthly demand rates of HSD oil for some of the selected equipment used for a typical Water Resources Project are shown in the Table 5.1. Period of time to calculate the demand rate was adopted in weeks, and since the schedule of project construction is usually expressed in months, the monthly demand is converted to weekly demand by multiplying a factor $12/52 = 0.23$.

The assumed constant lead time = 1 week.

The maximum capacity of a fuel tanker = 12,000 litres, and so we fixed the lot size to be 12,000 litres (fixed order quantity lot sizing technique).

Number of orders in a week = $1,59,056 / 12,000 = 13.25$ or say 14, and we clubbed all these fourteen orders into one.

So, total order quantity per period, i.e. per week = 1,68,000 litres.

We fixed the bucket time to be 1 week, and the total planning horizon to be 52 weeks.

Safety stock assumed = 1,68,000 litres.

For planning and controlling the flow of HSD oil, which is an independent demand item, we use the 'Time Phased Order Point Technique'.

In this technique, demand is forecasted, and supply is usually controlled by means of order points.

Order point = safety stock + (weekly demand forecast * lead time)

= $1,68,000 + (1,59,056 * 1) = 3,27,056$ litres.

Storage tank capacity for HSD oil = $14 * 4 * 12,000 = 6,72,000$ litres, where,

the storage tanks are designed for four weeks demand of HSD oil, and per week fourteen tankers are to be arrived.

Quantity on hand i.e. quantity of HSD oil just before the start of the construction of the project = 6,72,000 litres.

Now we run the MRP software in a weekly basis for finding the quantity and timing of the 'Planned Order Releases' for the HSD oil. Results are shown in Table 5.2, 5.3, 5.4, and 5.5.

Table 5.1: Demand Rates for HSD oil

Sl. No.	Description	BHP	LF	Monthly Working hrs	Monthly HSD oil Consump. in litres	MM	Total monthly Demand in litres	Weekly Demand In litres
1.	Bulldozer, BEML	100	0.5	200	1500	20	30,000	6,900
2.	Hydraulic Excavator, 2m ³	235	0.5	200	3525	9	31,725	7297
3.	Power Shovel 1.17 m ³	160	0.6	200	2880	9	25,920	5961
4.	Terex Rear Dump Truck	880	0.4	250	13,200	30	3,96,000	91080
5.	TATA Hoe, 1.15 m ³	160	0.6	200	2880	9	25,920	5962
6.	Terex Scraper, 10.7 m ³	320	0.6	200	5760	9	51,840	11923
7.	BEML Wheel Loader	145	0.5	200	2175	9	19,575	4502
8.	Truck Crane, 30 T	250	0.4	200	3000	9	27,000	6210
9.	Truck Crane, 20 T	200	0.4	150	1800	4	7,200	1656
10.	Road Roller, 10 T	125	0.4	150	1125	12	13,500	3105
Total							6,28,680	1,44,596
Add 10% contingencies							62,868	14,459
Grand Total							6,91,548	1,59,056
Annual Demand in litres (for 52 weeks)							82,98,576	

Table 5.2: Master Production Schedule

Lead Time: 1 Sstock: 1,68,000 Code#: 1	MASTER PRODUCTION SCHEDULE				
	Date: 01/02/2003				
	WEEKS				
	1	2	3	4	5
Projected Requirement	1,59,056	1,59,056	1,59,056	1,59,056	1,59,056

Table 5.3: Bill of Materials

BILL OF MATERIALS			
Date :01/02/2003			
Description	Code #	Level	Quantity Reqd. Per End Item
HSD Oil	1	1	Nil

Table 5.4: Inventory Status Record

Lead Time: 1 Sstock: 1,68,000 Code#: 1	INVENTORY STATUS RECORD					
	Date:01/02/2003					
	WEEKS					
	0	1	2	3	4	5
Scheduled Receipt						
On Hand Qty	6,72,000					

Table 5.5: Planned Order Release

Lead Time: 1 Sstock: 168,000 Code#: 1	PLANNED ORDER RELEASE					
	Date: 01/02/2003					
	WEEKS					
	Balance	1	2	3	4	5
Projected Requirement		1,59,056	1,59,056	1,59,056	1,59,056	1,59,056
Scheduled Receipts						
On Hand Qty.	6,72,000	5,12,944	3,53,888	1,94,832	2,03,776	2,12,720
Planned Order Receipt				168000	168000	168000
Planned Order Release			168000	168000	168000	

5.2.2 Results And Discussions

The forecast of HSD oil is projected over the entire planning horizon of 52 weeks, and represents the projected requirements. The quantity on hand of 6,72,000 litres, just before the start of the project goes below the safety stock level of 1,68,000 litres in week 4, and a replenishment order of 1,68,000 litres is planned to arrive at that time. Offsetting for lead time, the 'Planned Order Release' is scheduled for week 3, where the quantity on hand drops down the order point of 3,27,056 litres. Actually, the replenishment order is triggered by the on hand quantity dropping down to the safety stock level, and it has nothing to do with the order point. Then continuously up to week 18, the quantity on hand remains below the safety stock level of 1,68,000 litres, and 'Planned Order Receipt' quantities equal to the lot size are generated, and subsequently offsetted against the lead time to produce 'Planned Order Releases'. During the whole 52 weeks, 'Planned Order Receipts' are not generated in weeks 1, 2, 3, 19, and 38. The MRP system generates an entire schedule of planned replenishment orders, using the time phased order point technique instead of one order at a time like other inventory policies.

The calculated weekly data of the inventory levels for the HSD oil spread over the entire planning horizon of 52 weeks are shown in the tabular form in Table 5.6, and in the graphical form in Fig. 5.1.

5.3 CASE 2

5.3.1 Data Acquisition and Computations

Horizontal girders are fabricated from rolled steel beams, and these rolled steel beams are purchased from steel plants. We have assumed the following data for this case study:

- Annual demand of the horizontal girders = 416 nos,
- Demand rate of the horizontal girders is fixed = 8 nos per week,
- Maximum planning horizon = 52 weeks,
- All the horizontal girders have same dimensions (ISMB 300),
- Average length of one horizontal girder = 2m,
- Unit purchase cost of horizontal girder

$$= (\text{Weight per metre} * \text{Cost of steel per kg} * \text{length})$$

$$= (44.2 * 27 * 2) = \text{Rs. } 2386.80 \text{ or say Rs. } 2387.00,$$
- Length of rolled steel beams purchased

$$= 10\text{m (i.e. equal to one long trailer truck length),}$$
- Unit purchase cost of rolled steel beam = Rs. 11,934.00,
- Annual demand of rolled steel beams = 83.2 + 10% contingencies

$$= 91.52 \text{ or say } 92 \text{ nos,}$$
- Order/setup cost of horizontal girder = Rs. 700.00,
- Order cost of rolled steel beam per lot = Rs. 900.00,
- Lead time in weeks for the horizontal girder = 1,
- Lead time in weeks for the rolled steel beam = 3,
- Holding cost fraction for both end item and the component = 0.25,
- Zero stock-outs for both end item and the component,
- Safety stock for horizontal girder in units = 8 nos,
- Safety stock for rolled steel beam = 6 nos,
- The purchased components are delivered to the storeroom one week ahead of

its requirement.

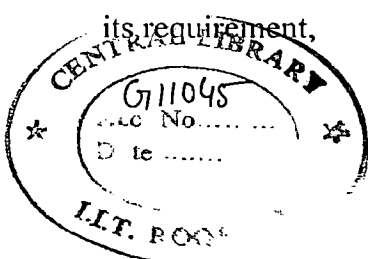


Table 5.6: Inventory Levels for MRP Model of HSD Oil

Weeks	Inventory Levels In Litres	Weeks	Inventory Levels In Litres
1	512944	27	241488*
2	353888	28	250432*
3	194832	29	259376*
4	203776*	30	268320*
5	212720*	31	277264*
6	221664*	32	286208*
7	230608*	33	295152*
8	239552*	34	304096*
9	248496*	35	313040*
10	257440*	36	321984*
11	266384*	37	330928*
12	275328*	38	171872
13	284272*	39	180816*
14	293216*	40	189760*
15	302160*	41	198704*
16	311104*	42	207648*
17	320048*	43	216592*
18	328992*	44	225536*
19	169936	45	234480*
20	178880*	46	243424*
21	187824*	47	252368*
22	196768*	48	261312*
23	205712*	49	270256*
24	214656*	50	279200*
25	223600*	51	288144*
26	232544*	52	297088*

N.B.: * mark indicates generation of 'Planned Order Receipts'.

MRP MODEL FOR HSD OIL

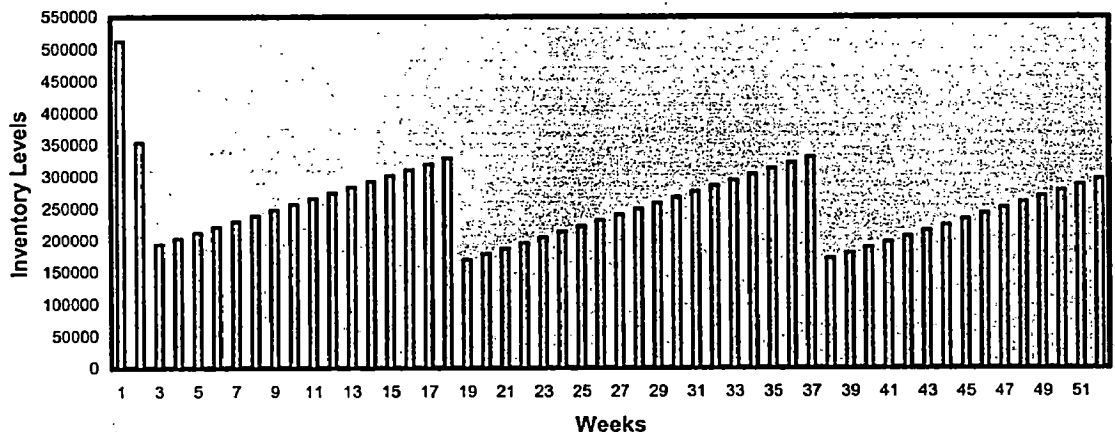


Fig. 5.1: Inventory Levels for MRP Model for HSD Oil

- Horizontal girders are not produced until the lot size of the rolled steel beams is available, and

Economic Order Quantity (EOQ) for the end item

$$= (2*416*700 / 2387*0.25)^{1/2}$$

$$= 31.24 \text{ or say } 32 \text{ nos,}$$

Reorder point for the end item = $(416*1 / 52) = 8$,

Economic Order Quantity (EOQ) for the component

$$= (2*92*900 / 11934*0.25)^{1/2}$$

$$= 7.45 \text{ or say } 8 \text{ nos,}$$

Reorder point for the component = $(92*3 / 52) = 5.31$ or say 6 nos,

5.3.2 EOQ Model

An EOQ model for this case study has been developed, and compared with the MRP model. The inventory levels of both the end item and the component are calculated and are given in Table 5.7.

It is assumed that a week before the start of production, i.e. during week 0, the following quantities of both the end item and the components are available:

Quantity of end item = 32 nos,

Quantity of components = 8 nos.

5.3.3 MRP Model

The demand rate of the horizontal girders is fixed, and was assumed to be eight per week spread over the entire planning horizon of 52 weeks. In the 'Product Structure Tree', the position of the horizontal girder is fixed at 'Level 1', and the position of the rolled steel beam is fixed at 'Level 2'. Now we will determine the material requirements in a level by level fashion.

It is assumed that the MRP system is using lot for lot ordering technique for the horizontal girders, and fixed lot size ordering technique for the rolled steel beams. The projected gross requirements of the horizontal girders are fed into the system via 'Master Production System', the detail inventory position is also fed into the system via the 'Inventory status Record', and then the application software determined the 'Planned Order Releases' of the horizontal girders. It is assumed that no quantity of the end item is expected to arrive during the whole 52 weeks.

5.3.4 Results and Discussions

The MRP system generates an entire schedule of planned replenishment orders for the whole planning horizon of 52 weeks.

The detail material plan of the horizontal girders are shown in Table 5.8, 5.9, 5.10, 5.12, and 5.13. Its inventory levels are also calculated and shown in Table 5.19.

The detail material plan of the rolled steel beams are shown in Table 5.11, 5.14, 5.15, 5.16, 5.17, and 5.18.

The inventory levels of these rolled steel beams are calculated and are shown in Table 5.19.

The inventory levels of the end item, i.e. the horizontal girder are zero after week 3, whereas it varies in a saw tooth pattern in the EOQ model.

The inventory levels of the component, i.e. the rolled steel beam vary in a regular fashion excepting when 'Scheduled Receipts' are awaited, whereas it regularly increase in the EOQ model. In comparison to the EOQ model, the component inventory levels are much smaller in the MRP system.

Annual end item inventory level in EOQ system = 1040

Annual component inventory level in EOQ system = 606.8

Total annual cost of the rolled steel beams in the EOQ system

$$\begin{aligned} &= \text{purchase cost} + \text{order cost} + \text{holding cost} \\ &= (11934 * 92) + (13 * 900) + (11934 * 0.25 * 606.8 / 52) \\ &= \text{Rs. } 11,44,443.20 \end{aligned}$$

Annual end item inventory level in MRP system = 48

Annual component inventory level in MRP system = 262.4

Total annual cost of the rolled steel beams in the MRP system

$$\begin{aligned} &= \text{purchase cost} + \text{order cost} + \text{holding cost} \\ &= (11934 * 92) + (9 * 900) + (11934 * 0.25 * 262.4 / 52) \\ &= \text{Rs. } 11,21,083.20 \end{aligned}$$

MRP cost savings in components = (EOQ annual cost – MRP annual cost)

$$= \text{Rs. } 11,44,443.20 - \text{Rs. } 11,21,083.20$$

$$= \text{Rs. } 23,360.00$$

Table 5.7: Inventory Level in EOQ Model for Case 2

Week	End Item Demand	End Item Inv. Level	Compo. Demand	Compo. Inv. Level	Week	End Item Demand	End Item Inv. Level	Compo. Demand	Compo. Inv. Level
1	8	24	0	8	27	8	8	6.4	11.2
2	8	16	0	8	28	8	32*	0	11.2
3	8	8	6.4	1.6	29	8	24	0	11.2
4	8	32*	0	1.6	30	8	16	0	17.2
5	8	24	0	1.6	31	8	8	6.4	10.8
6	8	16	0	9.6	32	8	32*	0	10.8
7	8	8	6.4	3.2	33	8	24	0	10.8
8	8	32*	0	3.2	34	8	16	0	18.8
9	8	24	0	3.2	35	8	8	6.4	12.4
10	8	16	0	11.2	36	8	32*	0	12.4
11	8	8	6.4	4.8	37	8	24	0	12.4
12	8	32*	0	4.8	38	8	16	0	20.4
13	8	24	0	4.8	39	8	8	6.4	14
14	8	16	0	12.8	40	8	32*	0	14
15	8	8	6.4	6.4	41	8	24	0	14
16	8	32*	0	6.4	42	8	16	0	22
17	8	24	0	6.4	43	8	8	6.4	15.6
18	8	16	0	14.4	44	8	32*	0	15.6
19	8	8	6.4	8	45	8	24	0	15.6
20	8	32*	0	8	46	8	16	0	23.6
21	8	24	0	8	47	8	8	6.4	17.2
22	8	16	0	16	48	8	32*	0	17.2
23	8	8	6.4	9.6	49	8	24	0	17.2
24	8	32*	0	9.6	50	8	16	0	25.2
25	8	24	0	9.6	51	8	8	6.4	18.8
26	8	16	0	17.6	52	8	32*	0	18.8

N.B.: * marks indicate 'Planned Order Receipt' Generation

EOQ MODEL 1

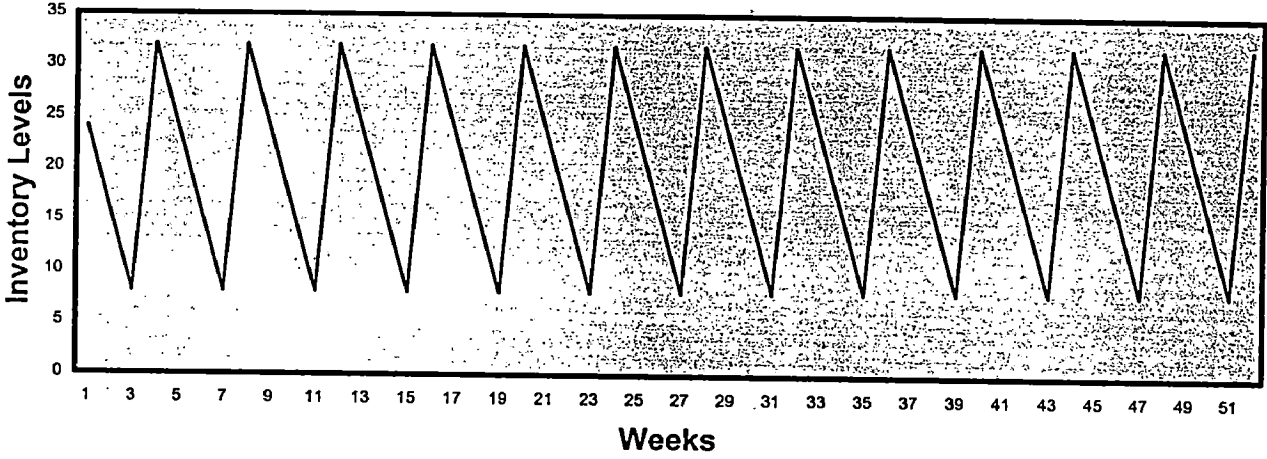


Fig. 5.2: EOQ Model 1 For the Horizontal Girder

EOQ MODEL 2

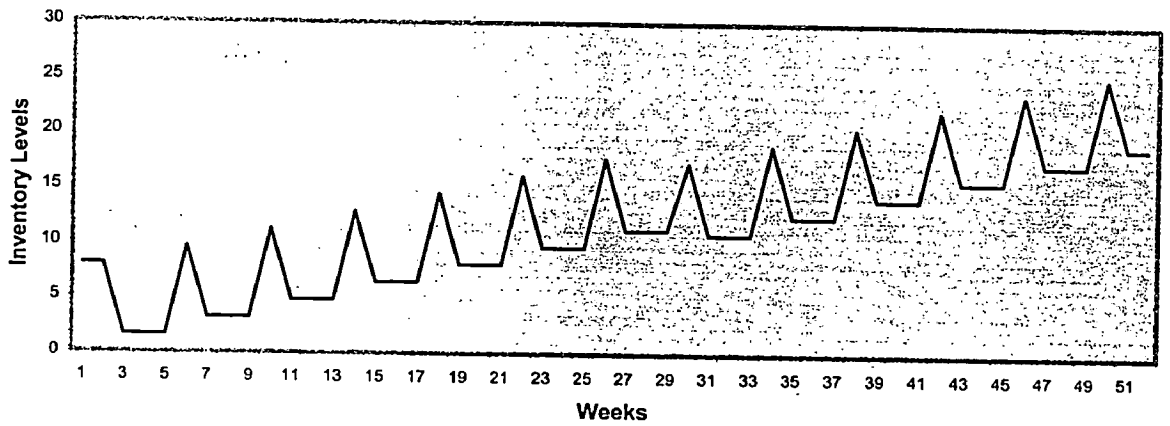


Fig. 5.3: EOQ Model 2 For Rolled Steel Beams

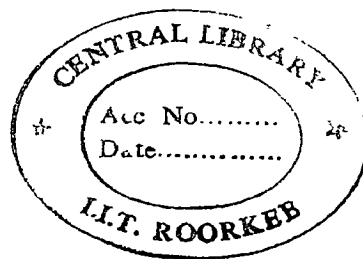


Table 5.8: Master Production Schedule for Level 1 of Case 2

Lead Time: 1 Sstock: 8 Code# : 2	MASTER PRODUCTION SCHEDULE				
	Date: 01/02/2003				
	WEEKS				
	1	2	3	4	5
Projected Requirement	8	8	8	8	8

Table 5.9: Bill of Materials for all Items Considered in Case 2

BILL OF MATERIALS			
Date: 01/02/2003			
Description	Code#	Level	Quantity. Req'd. Per End Item
Horizontal Girder	2	1	Nil
Rolled Steel Beam	3	2	0.2

Table 5.10: Inventory Status Record for Level 1 Item

Lead Time: 1 Sstock: 8 Code# : 2	INVENTORY STATUS RECORD					
	Date: 01/02/2003					
	WEEKS					
	0	1	2	3	4	5
Scheduled Receipt						
On Hand Qty	32	24	16	8	0	0

Table 5.11: Inventory Status Record for Level 2 Item

Lead Time: 3 Sstock: 6 Code# : 3	INVENTORY STATUS RECORD					
	Date: 01/02/2003					
	WEEKS					
	0	1	2	3	4	5
Scheduled Receipt						8
On Hand Qty	8	8	8	6.4	12.8	11.2

Table 5.12: Planned Order Release for Level 1 Item

Lead Time: 1 Sstock: 8 Code# : 2	PLANNED ORDER RELEASE					
	Date: 01/02/2003					
	WEEKS					
	Balance	1	2	3	4	5
Projected Requirement		8	8	8	8	8
Scheduled Receipts						
On hand Qty.	32	24	16	8	0	0
Net Requirement						8
Planned Order Receipt						8
Planned Order Release					8	8

Table 5.13: Planned Order Release for Level 1 Item

Lead Time: 1 Sstock: 8 Code # : 2	PLANNED ORDER RELEASE					
	Date: 01/02/2003					
	WEEKS					
	Balance	6	7	8	9	10
Projected Requirement		8	8	8	8	8
Scheduled Receipts						
On hand Qty.	0	0	0	0	0	0
Net Requirement		8	8	8	8	8
Planned Order Receipt		8	8	8	8	8
Planned Order Release		8	8	8	8	8

Table 5.14: Planned Order Release for Level 2 Item

Lead Time: 3 Sstock: 6 Code# : 3	PLANNED ORDER RELEASE					
	Date: 01/02/2003					
	WEEKS					
	Balance	1	2	3	4	5
Projected Requirement					1.6	1.6
Scheduled Receipts						
On hand Qty.	8	8	8	8	6.4	4.8
Net Requirement						
Planned Order Receipt						
Planned Order Release						

Table 5.15: Planned Order Release for Level 2 Item

Lead Time: 3 Sstock: 6 Code# : 3	PLANNED ORDER RELEASE					
	Date: 01/02/2003					
	WEEKS					
	Balance	6	7	8	9	10
Projected Requirement		1.6	1.6	1.6	1.6	1.6
Scheduled Receipts						
On hand Qty.	4.8	3.2	1.6	0	6.4	4.8
Net Requirement					1.6	
Planned Order Receipt					8	
Planned Order Release		8				

Table 5.16: Master Production Schedule for Level 2

Lead Time: 3 Sstock: 6 Code # : 3	MASTER PRODUCTION SCHEDULE				
	Date: 01/02/2003				
	WEEKS				
	1	2	3	4	5
Projected Requirement				1.6	1.6

Table 5.17: Master Production Schedule for Level 2

Lead Time: 3 Sstock: 6 Code# : 3	MASTER PRODUCTION SCHEDULE				
	Date: 01/02/2003				
	WEEKS				
	6	7	8	9	10
Projected Requirement	1.6	1.6	1.6	1.6	1.6

Table 5.18: Master Production Schedule for Level 2

Lead Time: 3 Sstock: 6 Code # : 3	MASTER PRODUCTION SCHEDULE				
	Date: 01/02/2003				
	WEEKS				
	11	12	13	14	15
Projected Requirement	1.6	1.6	1.6	1.6	1.6

Table 5.19: Inventory Level in MRP Model for Case 2

Week	End Item Demand	End Item Inv. Level	Compo. Demand	Compo. Inv. Level	Week	End Item Demand	End Item Inv. Level	Compo. Demand	Compo. Inv. Level
1	8	24	0	8	27	8	0	1.6	3.2
2	8	16	0	8	28	8	0	1.6	1.6
3	8	8	0	8	29	8	0	1.6	6.4*
4	8	0	1.6	6.4	30	8	0	1.6	12.8 ^{SR}
5	8	0	1.6	4.8	31	8	0	1.6	11.2
6	8	0	1.6	3.2	32	8	0	1.6	9.6
7	8	0	1.6	1.6	33	8	0	1.6	8
8	8	0	1.6	6.4*	34	8	0	1.6	6.4
9	8	0	1.6	4.8	35	8	0	1.6	4.8
10	8	0	1.6	3.2	36	8	0	1.6	3.2
11	8	0	1.6	1.6	37	8	0	1.6	1.6
12	8	0	1.6	6.4*	38	8	0	1.6	6.4*
13	8	0	1.6	4.8	39	8	0	1.6	4.8
14	8	0	1.6	3.2	40	8	0	1.6	3.2
15	8	0	1.6	9.6 ^{SR}	41	8	0	1.6	1.6
16	8	0	1.6	8	42	8	0	1.6	6.4*
17	8	0	1.6	6.4	43	8	0	1.6	4.8
18	8	0	1.6	4.8	44	8	0	1.6	3.2
19	8	0	1.6	3.2	45	8	0	1.6	1.6
20	8	0	1.6	1.6	46	8	0	1.6	6.4*
21	8	0	1.6	6.4*	47	8	0	1.6	4.8
22	8	0	1.6	4.8	48	8	0	1.6	3.2
23	8	0	1.6	3.2	49	8	0	1.6	1.6
24	8	0	1.6	1.6	50	8	0	1.6	6.4*
25	8	0	1.6	6.4*	51	8	0	1.6	4.8
26	8	0	1.6	4.8	52	8	0	1.6	3.2

N.B.: ^{SR} indicates 'Scheduled Receipt', and * mark indicates 'Planned Order Receipt' at Component Level

MRP MODEL 1

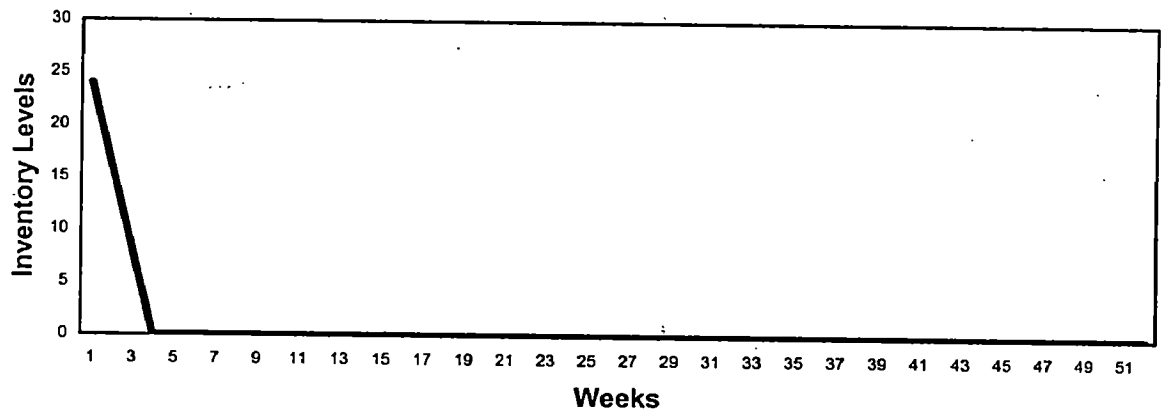


Fig. 5.4: Inventory levels for MRP Model of the Horizontal Girder

MRP MODEL 2

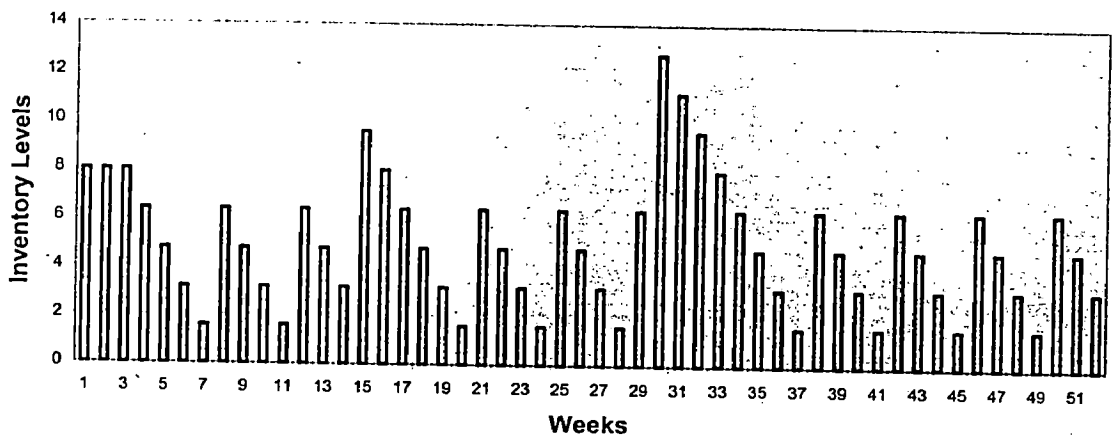


Fig. 5.5: Inventory Levels for Rolled Steel Beams

CONCLUSIONS

6.1 GENERAL

Inventory systems develop as the inventory control needs of an organization evolve. The signal for greater inventory control and installation of new systems can come from many sources. If the organization concerned loses orders, does not adequately indicate inventory status, causes too many stock-outs, accumulates excessive and surplus inventory, does not serve the purposes of the users, fails in production process, fails in timely completion of the work, or simply loses performance, then time has stepped in to rethink for developing a new inventory system or for overhauling the existing inventory system.

Water Resources Projects are invariably failing in completion of the project in due dates, and one of the major causes of it is failure in the adopted inventory planning and controlling systems. In this computer age, for smooth and sound management of huge quantum of materials in these projects, they must adopt the 'Material Requirements Planning' system. MRP system can be applied in Water Resources Projects for almost all materials irrespective of their type, quantity, and nature. To illustrate the use of MRP in Water Resources Projects, two case studies have been covered in this dissertation, and the conclusion of these case studies are presented in the following paragraphs.

6.2 CASE STUDY 1

The behavior of the HSD oil as an inventory in W.R. Projects has been studied through the application of MRP system to it. MRP generated periodical 'Planned Order Releases' for the entire planning horizon considered at the planning level, but in other inventory policies, the planning can be done period by period and only one at a time. So using MRP, the materials managers can better plan and control the flow of inventories. Like HSD oil, MRP can also be applied to all other independent demand

items, such as, Petrol, Lubricating Oil, Cement, Steel, and Tires using the 'Time Phased Order Point Technique'.

6.3 CASE STUDY 2

In the second case study, which covers the fabrication aspects of the Water Resources Projects, a typical case of horizontal girders for vertical lift gates, and their raw materials, i.e. rolled steel beams have been studied. It is found that using MRP, the inventory levels of the horizontal girders have been reduced drastically in comparison to the EOQ model, and in the component level, the inventory level is also very low in comparison to the EOQ model, where its inventory level is gradually increasing, and the plateaus in the EOQ model graph for the components indicate that inventory lies dormant in the storeroom. Also use of MRP system over the EOQ system is cost beneficial. Similarly, MRP system can be used for all other fabricated materials like all the components of a gate, and the steel ribs for the tunnels.

From the case studies conducted, it is found that 'Material Requirements Planning' system is best suited for planning and controlling all types of materials in a Water Resources Project.

6.4 SCOPE FOR FUTURE RESEARCH

Actual data and parameters may be obtained from any Water Resources Project, and similar studies may be conducted for all the materials like, Petrol, Lubricating Oil, Cement, Tires, production of steel ribs for tunnels, all the components of 'Vertical Lift Gates', and 'Radial Gates'.

A MRP system may be designed having an object database management system (ODBMS), and this database may be connected to the MRP application designed in any object-oriented programming language.

A full fledged MRP system may be constructed with at least five levels in the 'Product Tree', planning horizon up to 10 weeks, low level coding facilities, facilities to apply all the 'lot sizing techniques' and 'pegging' facilities etc.

Bibliography

REFERENCES

1. Bertrand Meyer (1997), "Object-Oriented Software Construction", Prentice Hall PTR, New York.
2. Candace Arai Yano and Robert C. Carlson (1987), "Interaction Between Frequency of Rescheduling and the Role of Safety Stock in Material Requirements Planning Systems", International Journal of Production Research, Vol. 25, No.2(pp 221-232).
3. Chrwan-Jyh Ho (1989), "Evaluating the Impact of Operating Environments on MRP System Nervousness", International Journal of Production Research, Vol. 27, No.7 (pp 1115-1135).
- ✕ 4. D. H. Noorie, et al,(1990), "Object-Oriented Management Planning Systems for Advanced Manufacturing", International Journal of Computer Integrated Manufacturing, Vol. 3, No. 6, (pp 373-378).
- ✕ 5. D. J. Bragg, et al,(1999), "The Effects of Partial Order Release and Component Reservation on Inventory and Customer Service Performance in an MRP Environment", International Journal of Production Research, Vol. 37, No. 3, (pp 523-538).
- ✕ 6. D. McAreavey, et al,(1988), "Designing the Closed Loop Elements of a Material Requirements Planning System in a Low Volume, Make-to-Order Company (with Case Study)", International Journal of Production Research, Vol. 26, No.7 (pp 1141-1159).
- ✕ 7. Ellis Horowitz et al. (1999), "Computer Algorithms/C++", Galgotia Publications Pvt. Ltd., New Delhi.
8. George Chryssolouris (1992), "Manufacturing Systems, Theory & Practice", Springer-Verlag, New York.
9. Grady Booch (2001), "Object-Oriented Analysis & Design with Applications", Pearson Education Asia, New Delhi.
10. Harlan C. Meal, et al,(1987), "Material Requirements Planning in Hierarchical Production Planning Systems", International Journal of Production Research, Vol. 25, No.7 (pp 947-956).

11. Herbert Schildt (1999), "C++: The Complete Reference, Third Edition", Tata McGraw-Hill Publishing Company Limited, New Delhi.
12. J. Hoey, et al (1986), "Designing a Material-Requirements-Planning System to Meet the Needs of Low-volume, Make-to-Order Companies (with Case Study)", International Journal of Production Research, Vol. 24, No.2 (pp 375-386).
13. J. Miltenburg (2001), "Production Planning Problem where Products have Alternate Routings and Bill Of Material", International Journal of Production Research, Vol. 39, No. 8, (pp 1755-1775).
14. John A. Havers and Frank W. Stubbs, Jr (1971), "Handbook of Heavy Construction", McGraw-Hill Book Company, New York.
15. John G. Wacker (1985), "A Theory of Material Requirements Planning (MRP): An Empirical Methodology to Reduce Uncertainty in MRP Systems", International Journal of Production Research, Vol. 23, No. 4 (pp 807-824).
- ✓ 16. Joseph Orlicky (1975), "Material Requirements Planning", McGraw-Hill Book Company, New York. *year 1960?*
17. Joseph S. Martinich (1999), "Production & Operations Management", John Wiley & Sons, Inc., New York.
18. Khalid Sheikh (2001), "Manufacturing Resource Planning (MRP II)", Tata McGraw-Hill Publishing Company Limited, New Delhi.
19. Michael J. Folk, et al (1998), "File Structures, An Object-Oriented Approach with C++", Pearson Education Asia, Inc., New Delhi.
20. Mitra et al (1986), "A Re-examination of Lot Sizing Procedures for Requirement Planning System: Some Modified Rule", International Journal of Production Research, Vol. 21, No. 4 (pp 471-478).
21. Oliver W. Wight (1974), "Production & Inventory Management in the Computer Age", Cahners Books International, Inc., Massachusetts.
22. R. L. Peurifoy and W. B. Ledbetter (1985), "Construction Planning, Equipment, and Methods", McGraw-Hill Book Company, New York.
23. Richard B. Chase and Nicholas J. Aquilano (1977), "Productions & Operations Management", Richard D. Irwin, Inc., Illinois.
24. Richard J. Tersine (1988), "Principles of Inventory & Materials Management", North-Holland, New York.

25. Robert Lafore (2000), "Object-Oriented Programming in Turbo C++", Galgotia Publications Pvt. Ltd, New Delhi.
26. Shari Lawrence Pfleeger (2001), "Software Engineering, Theory and Practice", Pearson Education Asia, Inc., New Delhi.
27. Timon Chih-Ting Du and Philip M. Wolfe (2000), "Building an Active Material Requirements Planning System", International Journal of Production Research, Vol. 38, No.2 (pp 241-252).

SOFTWARE LISTING

```

/*
Application of Material Requirements Planning (MRP)
model in Water Resources Projects
DETAIL SOFTWARE IN A FLOPPY DISK
File: mrp.cpp
*/
#include <iostream.h>
#include <fstream.h>
#include <process.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>
#include <iomanip.h>
#include <graphics.h>
#include <math.h>

typedef char option[20];
const int ROW = 20, COL = 20;
option a[]={
    "NEW ITEM CODE",
    "INPUT DATA",
    "CLOSE CODE",
    "DISPLAY CODES",
    "MODIFY DATA",
    "RECORDS",
    "EXIT"
};
option b[]=
{
    "INV.STA.RECORD",
    "MAS.P.SCHEDULE",
    "BILL OF MAT.",
    "PLA.OR.RECEIPT",
    "PLA.OR.RELEASE",
    "DAILY REPORT",
    "WEEKLY REPORT",
    "EXCEP. NOTICE",
    "EXIT"
};
class entry_menu
{
private:
    int i,done;
public:
    void box(int x1,int y1,int x2,int y2);
    char menu();
    void main_menu();
    void control_menu();
    char re_menu();
    void screen();
};
class shape
{
public:
    void box(int, int, int, int, char);
};
items

```

```

class initial
{
public:
    void add_to_file(unsigned int,char i_name[20],
        unsigned int,unsigned int,unsigned int,unsigned int,
        unsigned int,unsigned int,unsigned int,unsigned int);
    void add_data(unsigned int,unsigned int,unsigned
        int,unsigned int,unsigned int,unsigned int,unsigned
        int,unsigned int,unsigned int,unsigned int,unsigned
        int,unsigned int,unsigned int);
    void modify(void);
    char *return_name(unsigned int);
    void input(unsigned int);
    void input_data(void);
    void display_code_list(void);
    void display(unsigned int);
    void delete_code(unsigned int);
    unsigned int last_code(void);
    unsigned int search_code(unsigned int);
    int recordno(unsigned int);
    unsigned int compute_nr1(unsigned int cno);
    unsigned int compute_nr2(unsigned int cno);
    unsigned int compute_nr3(unsigned int cno);
    unsigned int compute_nr4(unsigned int cno);
    unsigned int compute_nr5(unsigned int cno);
    unsigned int compute_oh1(unsigned int cno);
    unsigned int compute_oh2(unsigned int cno);
    unsigned int compute_oh3(unsigned int cno);
    unsigned int compute_oh4(unsigned int cno);
    unsigned int compute_oh5(unsigned int cno);
    unsigned int compute_po1(unsigned int cno);
    unsigned int compute_po2(unsigned int cno);
    unsigned int compute_po3(unsigned int cno);
    unsigned int compute_po4(unsigned int cno);
    unsigned int compute_po5(unsigned int cno);
    unsigned int compute_plo1(unsigned int cno);
    unsigned int compute_plo2(unsigned int cno);
    unsigned int compute_plo3(unsigned int cno);
    unsigned int compute_plo4(unsigned int cno);
    unsigned int lot_sizing(unsigned int);
    unsigned int compute_rp(unsigned int);
    void display_isr(void);
    void display_mps(void);
    void display_bom(void);
    void display_por1(void);
    void display_por2(void);
    void daily_report(void);
    void weekly_report(void);
    void excep_notice(void);

private:
    void modify_code(unsigned int,char i_name[20],
        unsigned int,unsigned int,unsigned int,unsigned int,
        unsigned int,unsigned int,unsigned int,unsigned int,
        unsigned int,unsigned int,unsigned int,unsigned int,
        unsigned int,unsigned int,unsigned int);
    unsigned int codeno;char name[20];
    unsigned int ademand,hcost,ocost;//add_to_file
    unsigned int proj_req1,proj_req2,proj_req3,proj_req4,

```

```

proj_req5,sche_req1,sche_req2,sche_req3,sche_req4,
sche_req5,on_hand0,ltime,level,quantity,num_com,sstock;
unsigned int on_hand1,on_hand2,on_hand3,on_hand4,
on_hand5,net_req1,net_req2,net_req3,net_req4,
net_req5,puor1,puor2,puor3,puor4,puor5,plno1,plno2,
plno3,plno4,plno5,week0,week1,week2,
week3,week4,week5,lotsize,rpoint;
};
class material
{
public:
    void new_code(void);
    void close_code(void);
    void display_code(void);
    void delete_code(unsigned int);
    void add_data(char f_type[10]);
private:
    void box_for_display(int);
    unsigned int no_of_days(unsigned int, unsigned int,
    unsigned int, unsigned int,unsigned int,
    unsigned int);
    void add_to_file(unsigned int,unsigned int,
    unsigned int,unsigned int,char f_type[10]);
    char file_n[10];
    unsigned int codeno;//add_to_file
    unsigned int dd, mm, yy;
};
void entry_menu::normalvideo(int x,int y,char *str)
{
    gotoxy(x,y);
    cprintf("%s",str);
}
void entry_menu::reversevideo(int x,int y,char *str)
{
    textcolor(5+143);
    textbackground(WHITE);
    gotoxy(x,y);
    cprintf("%s",str);
    textcolor(GREEN);
    textbackground(BLACK);
}
void entry_menu::box(int x1,int y1,int x2,int y2)
{
    for(int col=x1;col<x2;col++)
    {
        gotoxy(col,y1);
        cprintf("%c",196);
        gotoxy(col,y2);
        cprintf("%c",196);
    }
    for(int row=y1;row<y2;row++)
    {
        gotoxy(x1,row);
        cprintf("%c",179);
        gotoxy(x2,row);
        cprintf("%c",179);
    }
    gotoxy(x1,y1);
    cprintf("%c",218);
    gotoxy(x1,y2);
}

```

```

        cprintf("%c",192);
        gotoxy(x2,y1);
        cprintf("%c",191);
        gotoxy(x2,y2);
        cprintf("%c",217);
    }
void entry_menu::control_menu()
{
    char choice;
    material ma;initial in;
    do
    {
        choice = menu();
        clrscr();
        switch (choice)
        {
            case '1':
                _setcursortype(_NORMALCURSOR);
                box(3, 1, 75, 24);
                box(5, 2, 73, 23);
                ma.new_code();
                break;
            case '2':
                _setcursortype(_NORMALCURSOR);
                box(3, 1, 75, 24);
                box(5, 2, 73, 23);
                in.input_data();
                break;
            case '3':
                _setcursortype(_NORMALCURSOR);
                box(3, 1, 75, 24);
                box(5, 2, 73, 23);
                ma.close_code();
                break;
            case '4':
                _setcursortype(_NORMALCURSOR);
                box(3, 1, 75, 24);
                box(5, 2, 73, 23);
                in.display_code_list();
                break;
            case '5':
                _setcursortype(_NORMALCURSOR);
                box(3, 1, 75, 24);
                box(5, 2, 73, 23);
                in.modify();
                break;
            case '6':
                _setcursortype(_NORMALCURSOR);
                box(3, 1, 75, 24);
                box(5, 2, 73, 23);
                main_menu();
                break;
            case '7':
                exit(0);
        }
    }
    while(choice!=6);
}
void entry_menu::main_menu()
{
    char choice;initial ib;

```

```

do
{
choice = re_menu();
clrscr();
switch (choice)
{
case '1':
box(3, 1, 75, 24);
box(5, 2, 73, 23);
ib.display_isr();
break;
case '2':
box(3, 1, 75, 24);
box(5, 2, 73, 23);
ib.display_mps();
break;
case '3':
box(3, 1, 75, 24);
box(5, 2, 73, 23);
ib.display_bom();
break;
case '4':
box(3, 1, 75, 24);
box(5, 2, 73, 23);
ib.display_por1();
break;
case '5':
box(3, 1, 75, 24);
box(5, 2, 73, 23);
ib.display_por2();
break;
case '6':
box(3, 1, 75, 24);
box(5, 2, 73, 23);
ib.daily_report();
break;
case '7':
box(3, 1, 75, 24);
box(5, 2, 73, 23);
ib.weekly_report();
break;
case '8':
box(3, 1, 75, 24);
box(5, 2, 73, 23);
ib.excep_notice();
break;
case '9':
return;
}
}while (choice != 8);
}
void entry_menu::screen(void)
{
driver = DETECT;
initgraph(&driver, &mode, "");
setbkcolor(10);
setcolor(5);
settextstyle(0,0,2);
outtextxy(50,50," P A P E R L E S S ");
settextstyle(0,0,5);
outtextxy(50,100,"MATERIAL");
}

```

```

    outtextxy(50,200,"REQUIREMENTS");
    outtextxy(50,300,"PLANNING ");
    settextstyle(0,0,3);
    outtextxy(50,370,"SYSTEM");
    delay(5000);closegraph();
    initgraph(&driver,&mode,"");
    setbkcolor(10);
    setcolor(1);
    settextstyle(0,0,2);
    outtextxy(50,100,"A");
    outtextxy(50,150,"DISSERTATION");
    outtextxy(50,200,"GUIDED BY:");
    outtextxy(50,250,"PROF. GOPAL CHAUHAN, WRDTC,");
    outtextxy(50,275,"&");
    outtextxy(50,300,"DR. PRADEEP KUMAR, M&IED,");
    outtextxy(50,340,"IIT ROORKEE,");
    outtextxy(50,365,"ROORKEE -247 667, INDIA");
    delay(5000);closegraph();
    initgraph(&driver,&mode,"");
    setbkcolor(10);
    setcolor(4);
    settextstyle(0,0,2);
    outtextxy(30,30,"SUBMITTED IN PARTIAL FULFILLMENT OF" );
    outtextxy(30,80,"THE REQUIREMENTS FOR THE AWARD OF ");
    outtextxy(30,130,"THE DEGREE OF ");
    outtextxy(30,180,"MASTER OF TECHNOLOGY");
    settextstyle(0,0,1);
    outtextxy(30,230,"IN");
    settextstyle(0,0,2);
    outtextxy(30,280,"WRD(MECHANICAL)");
    delay(5000);
    closegraph();
}
void material::new_code(void)
{
    gotoxy(3,3);
    for (i = 1; i<= 76; i++)
        cprintf(" ");
    textbackground(BLACK);
    textcolor(BLACK+BLINK);
    textbackground(WHITE);
    gotoxy(30,3);
    cprintf("OPEN NEW ITEM CODE");
    textcolor(LIGHTGRAY);
    textbackground(BLACK);
    int d1, m1, y1;
    struct date d;
    getdate(&d);
    d1 = d.da_day;m1 = d.da_mon;y1 = d.da_year;
    unsigned int cno;
    cno = ini.last_code();
    cno++;
    if (cno ==1)
    {
        ini.add_to_file(cno,"abc",1,1,1,1,1,1,1,1);
        ini.delete_code(cno);
        gotoxy(2,2);
        cout<<"Press Enter";
        getch();
        add_to_file(cno, 1, 1, 2003,"INITIAL");
        delete_code(cno);
    }
}

```



```

}
char i_name[20]; unsigned int ad, t_ad,
hc, t_hc, oc, t_oc, lt, t_lt, lv, t_lv, q, t_q, nc, t_nc,
ss, t_ss; char swap[10];
gotoxy(2, 3);
cout<<"Date: "<<dl<<'/'<<ml<<'/'<<y1;
gotoxy(2, 5);
cout<<"Item Code#:"<<cno;
gotoxy(55, 6);
cout<<"Input Parameters";
gotoxy(55, 7);
cout<<"=====";
gotoxy(2, 9);
cout<<"Item Name                :";
gotoxy(2, 10);
cout<<"Annual Demand(in Units)   :";
gotoxy(2, 11);
cout<<"Holding Cost(Rs. per unit per Year) :";
gotoxy(2, 12);
cout<<"Ordering Cost(Rs. per Order)   :";
gotoxy(2, 13);
cout<<"Safety Stock(in Units)       :";
gotoxy(2, 14);
cout<<"Lead Time(in Weeks)          :";
gotoxy(2, 15);
cout<<"Level in Product Tree (Top Level 1) :";
gotoxy(2, 16);
cout<<"Quantity Required for Upper Level Item:";
gotoxy(2, 17);
cout<<"No. of Components(if end item)   :";

do
{
    valid = 1;
    clear(55, 9);
    gotoxy(55, 9);
    gets(i_name);
   strupr(i_name);
    if (i_name[0] == '0')
        return;
    if (strlen(i_name) == 0 || strlen(i_name) > 20)
    {
        valid = 0;
        gotoxy(2, 22);
        cprintf("Name must not be greater than 20");
        getch();
    }
}
while (!valid);
do
{
    valid = 1;
    clear(55, 10);
    gotoxy(55, 10);
    gets(swap);
    t_ad=atoi(swap);
    ad=t_ad;
    if (swap[0] == '0')
    {
        valid = 0;
    }
}

```

```

}
while (!valid);
do
{
    valid = 1;
    clear(55,11);
    gotoxy(55,11);
    gets(swap);
    t_hc=atoi(swap);
    hc=t_hc;
    if (swap[0] == '0')
    {
        valid = 0;
    }
}
while (!valid);
do
{
    valid = 1;
    clear(55,12);
    gotoxy(55,12);
    gets(swap);
    t_oc=atoi(swap);
    oc=t_oc;
    if (swap[0] == '0')
    {
        valid = 0;
    }
}
while (!valid);
do
{
    valid = 1;
    clear(55,13);
    gotoxy(55,13);
    gets(swap);
    t_ss=atoi(swap);
    ss=t_ss;
    if (swap[0] == '0')
    {
        valid = 0;
    }
}
while (!valid);
do
{
    valid = 1;
    clear(55,14);
    gotoxy(55,14);
    gets(swap);
    t_lt=atoi(swap);
    lt=t_lt;
    if (swap[0] == '0')
    {
        valid = 0;
    }
}
while (!valid);
do
{
    valid = 1;

```

```

        clear(55,15);
        gotoxy(55,15);
        gets(swap);
        t_lv=atoi(swap);
        lv=t_lv;
        if (swap[0] == '0')
        {
            valid = 0;
        }
    }
    while (!valid);
    do
    {
        valid = 1;
        clear(55,16);
        gotoxy(55,16);
        gets(swap);
        t_q=atoi(swap);
        q=t_q;
        if (swap[0] == '0')
        {
            valid = 0;
        }
    }
    while (!valid);
    do
    {
        valid = 1;
        clear(55,17);
        gotoxy(55,17);
        gets(swap);
        t_nc=atoi(swap);
        nc=t_nc;
        if (swap[0] == '0')
        {
            valid = 0;
        }
    }
    while (!valid);

    do
    {
        clear(2, 24);
        valid = 1;
        gotoxy(2, 24);
        cout << "Want to save the record <Y/N> ?";
        ch=getche();
        if(ch=='0')
            return;
        ch = toupper(ch);
    }
    while (ch != 'N' && ch != 'Y');
    if (ch == 'N')
        return;
    unsigned int lot,rp;char f_type[10];
    strcpy(f_type, "INITIAL");
    ini.add_to_file(cno, i_name,ad,hc,oc,ss,lt,lv,q,nc);
    add_to_file(cno, dl, ml, y1, f_type);
}
void initial::input_data(void)
{

```

```

gotoxy(3,3);
for (j = 1; j<= 76;j++)
cprintf(" ");
textbackground(BLACK);
textcolor(BLACK+BLINK);
textbackground(WHITE);
gotoxy(35, 3);
cprintf("INPUT DATA");
textcolor(LIGHTGRAY);
textbackground(BLACK);
int d1, m1, y1;
struct date d;
getdate(&d);
d1 = d.da_day;m1 = d.da_mon;y1 = d.da_year;
unsigned int cno;
cno =last_code();
if (cno ==1)
{
    add_data(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1);
    delete_code(cno);
    gotoxy(2,1);
    cout << "Press Enter";
    getch();
    mi.add_data("INITIAL");
    mi.delete_code(cno);
}
char i_name[20];char swap[10];
unsigned int pr1,pr2,pr3,pr4,pr5,sr1,sr2,
sr3,sr4,sr5,oh0;
gotoxy(2,6);
cout<<"Date: "<<d1<< '/'<<m1<< '/'<<y1;
display(cno);
gotoxy(2,5);
cout<<"Code#:"<<cno;
gotoxy(2,7);
cout<<"Item Name:"<<i_name;
gotoxy(1,9);
for(j=1;j<=79;j++)
cout << "=";
gotoxy(2,10);
cout<<"PERIOD:";
gotoxy(28,10);
cout<<"Week0      Week1      Week2      Week3      Week4      Week5";
gotoxy(1,11);
for(j=1;j<=79;j++)
cout<<"=";
gotoxy(2,12);
cout<<"Put Week Values:";
gotoxy(1,13);
for(j=1;j<=79;j++)
cout << "=";
gotoxy(2,14);
cout<<"Projected Requirement:";
gotoxy(2,15);
cout<<"Scheduled Receipt      :";
gotoxy(2,16);
cout<<"Quantity On Hand      :";
gotoxy(1,18);
for(j=1;j<=79;j++)
cout << "=";
unsigned int t_w0,t_w1,t_w2,t_w3,t_w4,t_w5,

```

```

wk0,wk1,wk2,wk3,wk4,wk5;
unsigned int t_pr1,t_pr2,t_pr3,t_pr4,t_pr5,
t_sr1,t_sr2,t_sr3,t_sr4,t_sr5,t_oh0;
unsigned int nr1,nr2,nr3,nr4,nr5,pol,po2,
po3,po4,po5,plol,plo2,plo3,plo4,plo5;

do
{
mi.clear(28,12);
valid = 1;
gotoxy(28,12);
gets(swap);
t_w0=atoi(swap);
wk0=t_w0;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);

do
{
mi.clear(39,12);
valid = 1;
gotoxy(39,12);
gets(swap);
t_w1=atoi(swap);
wk1=t_w1;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);

do
{
mi.clear(47,12);
valid = 1;
gotoxy(47,12);
gets(swap);
t_w2=atoi(swap);
wk2=t_w2;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);

do
{
mi.clear(56,12);
valid = 1;
gotoxy(56,12);
gets(swap);
t_w3=atoi(swap);
wk3=t_w3;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);

```

```

do
{
mi.clear(65,12);
valid = 1;
gotoxy(65,12);
gets(swap);
t_w4=atoi(swap);
wk4=t_w4;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);

```

```

do
{
mi.clear(75,12);
valid = 1;
gotoxy(75,12);
gets(swap);
t_w5=atoi(swap);
wk5=t_w5;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);

```

```

do
{
mi.clear(39,14);
valid = 1;
gotoxy(39,14);
gets(swap);
t_pr1=atoi(swap);
pr1=t_pr1;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);

```

```

do
{
mi.clear(47,14);
valid = 1;
gotoxy(47,14);
gets(swap);
t_pr2=atoi(swap);
pr2=t_pr2;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);

```

```

do
{
mi.clear(56,14);
valid = 1;
gotoxy(56,14);

```

```

gets(swap);
t_pr3=atoi(swap);
pr3=t_pr3;
if (swap[0] == '0')
{
    valid = 0;
}
}while (!valid);

do
{
    mi.clear(65,14);
    valid = 1;
    gotoxy(65,14);
    gets(swap);
    t_pr4=atoi(swap);
    pr4=t_pr4;
    if (swap[0] == '0')
    {
        valid = 0;
    }
}while (!valid);

do
{
    mi.clear(75,14);
    valid = 1;
    gotoxy(75,14);
    gets(swap);
    t_pr5=atoi(swap);
    pr5=t_pr5;
    if (swap[0] == '0')
    {
        valid = 0;
    }
}while (!valid);

do
{
    mi.clear(39,15);
    valid = 1;
    gotoxy(39,15);
    gets(swap);
    t_sr1=atoi(swap);
    sr1=t_sr1;
    if (swap[0] == '0')
    {
        valid = 0;
    }
}while (!valid);

do
{
    mi.clear(47,15);
    valid = 1;
    gotoxy(47,15);
    gets(swap);
    t_sr2=atoi(swap);
    sr2=t_sr2;
    if (swap[0] == '0')
    {
        valid = 0;
    }
}

```

```

    }
}while (!valid);

do
{
mi.clear(56,15);
valid = 1;
gotoxy(56,15);
gets(swap);
t_sr3=atoi(swap);
sr3=t_sr3;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);

do
{
mi.clear(65,15);
valid = 1;
gotoxy(65,15);
gets(swap);
t_sr4= atoi(swap);
sr4=t_sr4;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);
do
{
mi.clear(75,15);
valid = 1;
gotoxy(75,15);
gets(swap);
t_sr5= atoi(swap);
sr5=t_sr5;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);
do
{
mi.clear(28,16);
valid = 1;
gotoxy(28,16);
gets(swap);
t_oh0= atoi(swap);
oh0=t_oh0;
if (swap[0] == '0')
{
valid = 0;
}
}while (!valid);
do
{
mi.clear(2,24);
valid = 1;

```



```

        gotoxy(2,24);
        cout << "Do you want to save the record <Y/N> ? ";
        ch = getche();
        if (ch == '0')
            return;
        ch = toupper(ch);
    }while (ch != 'N' && ch != 'Y');
    if (ch == 'N')
        return;
    char f_type[10];
    strcpy(f_type, "INITIAL");
    add_data(pr1,pr2,pr3,pr4,pr5,sr1,sr2,
    sr3,sr4,sr5,oh0,wk0,wk1,wk2,wk3,wk4,wk5);
    mi.add_data(f_type);
}

unsigned int initial::lot_sizing(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    unsigned int lot;
    while (file.read((char *)this, sizeof(initial)))
    {
        if (codeno == cno)
        {
            lot=sqrt(2*ademand*ocost/hcost);
            lotsize=lot;
            break;
        }
    }
    file.close();
    return lotsize;
}

void material::close_code(void)
{
    clrscr();
    char i_cno[10];char ch;
    unsigned int t,cno;
    gotoxy(70, 1);
    cout << "<0>=EXIT";
    gotoxy(2,6);
    cout << "Enter the Code#:";
    gets(i_cno);
    t=atoi(i_cno);
    cno=t;
    if (cno == 0)
        return;
    clrscr();
    initial ini;
    if (!ini.search_code(cno))
    {
        gotoxy(2,8);
        cout << "Code not found! ";
        getch();
        return;
    }
    gotoxy(70,1);
    cout<<"<0>=EXIT";
    gotoxy(3, 3);
    textbackground(WHITE);
}

```

```

for (int i = 1; i <= 76; i++)
    cprintf(" ");
textbackground(BLACK);
textcolor(BLACK+BLINK);
textbackground(WHITE);
gotoxy(34, 3);
cprintf("CLOSE CODE");
textcolor(LIGHTGRAY);
textbackground(BLACK);
int d1, m1, y1;
struct date d;
getdate(&d);
d1 = d.da_day; m1 = d.da_mon; y1 = d.da_year;
gotoxy(2, 6);
cout << "Date: " << d1 << "/" << m1 << "/" << y1;
ini.display(cno);
do
{
    clear(2, 22);
    gotoxy(2, 22);
    cout << "Close this Code <Y|N> ?";
    ch = getch();
    if (ch == '0')
        return;
    ch = toupper(ch);
}
while (ch != 'N' && ch != 'Y');
if (ch == 'N')
    return;
ini.delete_code(cno);
delete_code(cno);
gotoxy(2, 23);
cout << "Item Code Deleted";
gotoxy(2, 24);
cout << "Press any key to continue...";
getch();
}
void initial::display(unsigned int cno)
{
    ifstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    while (file.read((char *)this, sizeof(initial)))
    {
        if (cno == codeno)
        {
            gotoxy(2, 5);
            cout << "Code#:" << codeno;
            gotoxy(2, 7);
            cout << "Item Name:";
            puts(name);
            break;
        }
    }
    file.close();
}
void initial::modify_code(unsigned int cno, char i_name[20],
unsigned int ad, unsigned int hc, unsigned int oc, unsigned int
ss, unsigned int lt, unsigned int lv, unsigned int q, unsigned
int nc, unsigned int pr1, unsigned int pr2, unsigned int pr3,
unsigned int pr4, unsigned int pr5, unsigned int srl, unsigned

```

```

int sr2,unsigned int sr3,unsigned int sr4,unsigned int sr5,
unsigned int oh0)
{
    int recno;recno=recordno(cno);
    fstream file;
    file.open("INITIAL.dat", ios::out|ios::ate);
    strcpy(name, i_name);
    ademand=ad;hcost=hc;ocost=oc;sstock=ss;
    ltime=lt;level=lv;quantity=q;num_com=nc;
    proj_req1=pr1;proj_req2=pr2;proj_req3=pr3;
    proj_req4=pr4;proj_req5=pr5;sche_req1=sr1;
    sche_req2=sr2;sche_req3=sr3;sche_req4=sr4;
    sche_req5=sr5;on_hand0=oh0;
    int location;
    location = (recno-1) * sizeof(initial);
    file.seekp(location);
    // Overwrites the modified record into INITIAL.dat data file
    file.write((char *)this, sizeof(initial));
    file.close();
    return;
}
unsigned int initial::last_code(void)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    int count = 0;
    // Finds the last code#
    while (file.read((char *)this, sizeof(initial)))
        count = codeno;
    file.close();
    return count;
}
unsigned int initial::compute_nrl(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week1==2))
        {
            net_req1=proj_req1-sche_req1-on_hand0;
            if(net_req1<=0)
            {
                net_req1=0;
            }
            break;
        }
        if((codeno == cno) && (level==2) && (week1==2))
        {
            net_req1=proj_req1-sche_req1-on_hand0;
            if(net_req1<=0)
            {
                net_req1=0;
            }
            break;
        }
    }
    file.close();
}

```

```

        return net_req1;
    }
    unsigned int initial::compute_nr2(unsigned int cno)
    {
        fstream file;
        file.open("INITIAL.dat", ios::in);
        file.seekg(0, ios::beg);
        on_hand1=compute_oh1(cno);
        while (file.read((char *)this, sizeof(initial)))
        {
            if((codeno == cno) && (level==1) && (week2==3))
            {
                net_req2=proj_req2-sche_req2-on_hand1;
                if(net_req2<=0)
                {
                    net_req2=0;
                }
                break;
            }
            if((codeno == cno) && (level==2) && (week2==3))
            {
                net_req2=proj_req2-sche_req2-on_hand1;
                if(net_req2<=0)
                {
                    net_req2=0;
                }
                break;
            }
        }
        file.close();
        return net_req2;
    }
    unsigned int initial::compute_nr3(unsigned int cno)
    {
        fstream file;
        file.open("INITIAL.dat", ios::in);
        file.seekg(0, ios::beg);
        on_hand2=compute_oh2(cno);
        while (file.read((char *)this, sizeof(initial)))
        {
            if((codeno == cno) && (level==1) && (week3==4))
            {
                net_req3=proj_req3-sche_req3-on_hand2;
                if(net_req3<=0)
                {
                    net_req3=0;
                }
                break;
            }
            if((codeno == cno) && (level==2) && (week3==4))
            {
                net_req3=proj_req3-sche_req3-on_hand2;
                if(net_req3<=0)
                {
                    net_req3=0;
                }
                break;
            }
        }
    }
}

```

```

        file.close();
        return net_req3;
    }
    unsigned int initial::compute_nr4(unsigned int cno)
    {
        fstream file;
        file.open("INITIAL.dat", ios::in);
        file.seekg(0, ios::beg);
        on_hand3=compute_oh3(cno);
        while (file.read((char *)this, sizeof(initial)))
        {
            if((codeno == cno) && (level==1) && (week4==5))
            {
                net_req4=proj_req4-sche_req4-on_hand3;
                if(net_req4<=0)
                {
                    net_req4=0;
                }
                break;
            }
            if((codeno == cno) && (level==2) && (week4==5))
            {
                net_req4=proj_req4-sche_req4-on_hand3;
                if(net_req4<=0)
                {
                    net_req4=0;
                }
                break;
            }
        }
        file.close();
        return net_req4;
    }
    unsigned int initial::compute_nr5(unsigned int cno)
    {
        fstream file;
        file.open("INITIAL.dat", ios::in);
        file.seekg(0, ios::beg);
        on_hand4=compute_oh4(cno);
        while (file.read((char *)this, sizeof(initial)))
        {
            if((codeno == cno) && (level==1) && (week5==6))
            {
                net_req5=proj_req5-sche_req5-on_hand4;
                if(net_req5<=0)
                {
                    net_req5=0;
                }
                break;
            }
            if((codeno == cno) && (level==2) && (week5==6))
            {
                net_req5=proj_req5-sche_req5-on_hand4;
                if(net_req5<=0)
                {
                    net_req5=0;
                }
                break;
            }
        }
    }

```

```

    }
    file.close();
    return net_req5;
}
unsigned int initial::compute_po1(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    lotsize=lot_sizing(cno);
    net_req1=compute_nr1(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week1==2))
        {
            if(net_req1>=lotsize)
            {
                puor1=net_req1;
            }

            else if((net_req1<lotsize) &&(net_req1>0))
            {
                puor1=lotsize;
            }

            else if(net_req1==0)
            {
                puor1=0;
            }
            break;
        }
        if((codeno == cno) && (level==2) && (week1==2))
        {
            if(net_req1>=lotsize)
            {
                puor1=net_req1;
            }

            else if((net_req1<lotsize) &&(net_req1>0))
            {
                puor1=lotsize;
            }

            else if(net_req1==0)
            {
                puor1=0;
            }
            break;
        }
    }
    file.close();
    return puor1;
}
unsigned int initial::compute_po2(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    lotsize=lot_sizing(cno);
    net_req2=compute_nr2(cno);

```

```

while (file.read((char *)this, sizeof(initial)))
{
    if((codeno == cno) && (level==1) && (week2==3))
    {
        if(net_req2>=lotsize)
        {
            puor2=net_req2;
        }

        else if((net_req2<lotsize) &&(net_req2>0))
        {
            puor2=lotsize;
        }

        else if(net_req2==0)
        {
            puor2=0;
        }
        break;
    }
    if((codeno == cno) && (level==2) && (week2==3))
    {
        if(net_req2>=lotsize)
        {
            puor2=net_req2;
        }

        else if((net_req2<lotsize) &&(net_req2>0))
        {
            puor2=lotsize;
        }

        else if(net_req2==0)
        {
            puor2=0;
        }
        break;
    }
}
file.close();
return puor2;
}

unsigned int initial::compute_po3(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    lotsize=lot_sizing(cno);
    net_req3=compute_nr3(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week3==4))
        {
            if(net_req3>=lotsize)
            {
                puor3=net_req3;
            }

            else if((net_req3<lotsize) &&(net_req3>0))
            {

```

```

        puor3=lotsize;
    }

    else if(net_req3==0)
    {
        puor3=0;
    }
    break;
}
if((codeno == cno) && (level==2) && (week3==4))
{
    if(net_req3>=lotsize)
    {
        puor3=net_req3;
    }

    else if((net_req3<lotsize) &&(net_req3>0))
    {
        puor3=lotsize;
    }

    else if(net_req3==0)
    {
        puor3=0;
    }
    break;
}
}
file.close();
return puor3;
}
unsigned int initial::compute_po4(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    lotsize=lot_sizing(cno);
    net_req4=compute_hr4(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week4==5))
        {
            if(net_req4>=lotsize)
            {
                puor4=net_req4;
            }

            else if((net_req4<lotsize) &&(net_req4>0))
            {
                puor4=lotsize;
            }

            else if(net_req4==0)
            {
                puor4=0;
            }
            break;
        }
        if((codeno == cno) && (level==2) && (week4==5))
        {

```



```

        if(net_req4>=lotsize)
        {
            puor4=net_req4;
        }

    else if((net_req4<lotsize) &&(net_req4>0))
    {
        puor4=lotsize;
    }

    else if(net_req4==0)
    {
        puor4=0;
    }
    break;
}

}
file.close();
return puor4;
}
unsigned int initial::compute_po5(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    lotsize=lot_sizing(cno);
    net_req5=compute_nr5(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week5==6))
        {
            if(net_req5>=lotsize)
            {
                puor5=net_req5;
            }

            else if((net_req5<lotsize) &&(net_req5>0))
            {
                puor5=lotsize;
            }

            else if(net_req5==0)
            {
                puor5=0;
            }
            break;
        }
        if((codeno == cno) && (level==2) && (week5==6))
        {
            if(net_req5>=lotsize)
            {
                puor5=net_req5;
            }

            else if((net_req5<lotsize) &&(net_req5>0))
            {
                puor5=lotsize;
            }

            else if(net_req5==0)

```

```

        {
            puor5=0;
        }
        break;
    }

    }
    file.close();
    return puor5;
}
unsigned int initial::compute_oh1(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    puor1=compute_po1(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week1==2))
        {
            on_hand1=sche_req1+puor1+on_hand0-proj_req1;
            break;
        }
        if((codeno == cno) && (level==2) && (week1==2))
        {
            on_hand1=sche_req1+puor1+on_hand0-proj_req1;
            break;
        }
    }
    file.close();
    return on_hand1;
}
unsigned int initial::compute_oh2(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    puor2=compute_po2(cno);
    on_hand1=compute_oh1(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week2==3))
        {
            on_hand2=sche_req2+puor2+on_hand1-proj_req2;
            break;
        }
        if((codeno == cno) && (level==2) && (week1==2))
        {
            on_hand2=sche_req2+puor2+on_hand1-proj_req2;
            break;
        }
    }
    file.close();
    return on_hand2;
}
unsigned int initial::compute_oh3(unsigned int cno)
{

```

```

fstream file;
file.open("INITIAL.dat", ios::in);
file.seekg(0, ios::beg);
puor3=compute_po3(cno);
on_hand2=compute_oh2(cno);
while (file.read((char *)this, sizeof(initial)))
{
    if((codeno == cno) && (level==1) && (week3==4))
    {
        on_hand3=sche_req3+puor3+on_hand2-proj_req3;
        break;
    }
    if((codeno == cno) && (level==2) && (week3==4))
    {
        on_hand3=sche_req3+puor3+on_hand2-proj_req3;
        break;
    }
}
file.close();
return on_hand3;
}
unsigned int initial::compute_oh4(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    puor4=compute_po4(cno);
    on_hand3=compute_oh3(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week4==5))
        {
            on_hand4=sche_req4+puor4+on_hand3-proj_req4;
            break;
        }
        if((codeno == cno) && (level==2) && (week4==5))
        {
            on_hand4=sche_req4+puor4+on_hand3-proj_req4;
            break;
        }
    }

    file.close();
    return on_hand4;
}

unsigned int initial::compute_oh5(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    puor5=compute_po4(cno);
    on_hand4=compute_oh4(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week5==6))
        {
            on_hand5=sche_req5+puor5+on_hand4-proj_req5;
            break;
        }
    }
}

```

```

    }
    if((codeno == cno) && (level==2) && (week5==6))
    {
        on_hand5=sche_req5+puor5+on_hand4-proj_rec
        break;
    }
}
file.close();
return on_hand5;
}
unsigned int initial::compute_plol(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    puor2=compute_po2(cno);
    puor3=compute_po3(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week1==2) &&
(ltime==1))
        {
            plnol=puor2;
            break;
        }
        if((codeno == cno) && (level==1) && (week1==2) &&
(ltime==2))
        {
            plnol=puor3;
            break;
        }
        if((codeno == cno) && (level==2) && (week1==2) &&
(ltime==1))
        {
            plnol=puor2;
            break;
        }
        if((codeno == cno) && (level==2) && (week1==2) &&
(ltime==2))
        {
            plnol=puor3;
            break;
        }
    }
    file.close();
    return plnol;
}
unsigned int initial::compute_plo2(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    puor3=compute_po3(cno);
    puor4=compute_po4(cno);
    while (file.read((char *)this, sizeof(initial)))
    {

```

```

        if((codeno == cno) && (level==1) && (week2==3) &&
(ltime==1))
        {
            plno2=puor3;
            break;
        }
        if((codeno == cno) && (level==1) && (week2==3) &&
(ltime==2))
        {
            plno4=puor4;
            break;
        }
        if((codeno == cno) && (level==2) && (week2==3) &&
(ltime==1))
        {
            plno2=puor3;
            break;
        }
        if((codeno == cno) && (level==2) && (week2==3) &&
(ltime==2))
        {
            plno2=puor4;
            break;
        }
    }
    file.close();
    return plno2;
}
unsigned int initial::compute_plo3(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    puor4=compute_po4(cno);
    puor5=compute_po5(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week3==4) &&
(ltime==1))
        {
            plno3=puor4;
            break;
        }
        if((codeno == cno) && (level==1) && (week3==4) &&
(ltime==2))
        {
            plno3=puor5;
            break;
        }
        if((codeno == cno) && (level==2) && (week3==4) &&
(ltime==1))
        {
            plno3=puor4;
            break;
        }
        if((codeno == cno) && (level==2) && (week3==4) &&
(ltime==2)).
    }
}

```

```

        {
            plno3=puor5;
            break;
        }
    }
    file.close();
    return plno3;
}
unsigned int initial::compute_plo4(unsigned int cno)
{
    fstream file;
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    puor5=compute_po5(cno);
    while (file.read((char *)this, sizeof(initial)))
    {
        if((codeno == cno) && (level==1) && (week4==5) &&
(ltime==1))
        {
            plno4=puor5;
            break;
        }
        if((codeno == cno) && (level==2) && (week4==5) &&
(ltime==1))
        {
            plno5=puor5;
            break;
        }
    }
    file.close();
    return plno5;
}
void initial::display_isr(void)
{
    fstream file;
    gotoxy(30,1);
    cout<<"INVENTORY STATUS RECORD";
    int dl, ml, yl;
    struct date d;
    getdate(&d);
    dl=d.da_day;ml=d.da_mon;yl=d.da_year;
    gotoxy(65, 2);
    cout<<"Date:"<<dl<<"/"<<ml<<"/"<<yl;
    gotoxy(1, 2);
    cout<<"Lead Time:"<<ltime;
    gotoxy(1,3);
    cout<<"SStock   :"<<sstock;
    gotoxy(18,3);
    for (j = 18; j <= 79; j++)
    cout<<"=";
    gotoxy(42,4);
    cout<<"Weeks";
    gotoxy(18,5);
    for (j = 18; j <= 79; j++)
    cout<<"=";
    gotoxy(1, 4);
    cout<<"Code#:"<<last_code();
    gotoxy(18,6);
}

```

```

5");
printf("Balance          1          2          3          4
gotoxy(1, 7);
for (j = 1; j <=79; j++)
cout << "=";
gotoxy(1,9);
cout<<"Sche. Receipt:";
gotoxy(1,10);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,11);
cout<<"On Hand Qty  :";
gotoxy(1,12);
for (j = 1; j <= 79; j++)
cout << "=";
unsigned int oh0,sr1,sr2,sr3,sr4,sr5;
file.open("INITIAL.dat", ios::in);
file.seekg(0, ios::beg);
while (file.read((char *)this, sizeof(initial)));
{
    if(codeno==cno)
    {
        on_hand0=oh0;
        sche_req1=sr1;sche_req2=sr2;
        sche_req3=sr3;sche_req4=sr4;
        sche_req5=sr5;
    }
    gotoxy(col,11);
    cout<<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<on_hand0;
    col=32;
    gotoxy(col,9);
    cout<<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<sche_req1;
    col=43;
    gotoxy(col,9);
    cout<<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<sche_req2;
    col=54;
    gotoxy(col,9);
    cout<<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<sche_req3;
    col=65;
    gotoxy(col,9);
    cout<<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<sche_req4;
    col=75;
    gotoxy(col,9);
    cout<<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<sche_req5;
    if (col > 79)
    {
        flag = 1;
        col = 20;
        gotoxy(2, 24);
        cout << "Press any key to continue.... ";
    }
}

```

```

        getch();
        clrscr();
    }
}
    if (col > 79)
    {
        flag = 1;
        col = 20;
        gotoxy(2, 24);
        cout << "Press any key to continue.... ";
        getch();
        clrscr();
    }
file.close();
if (!flag)
{
    gotoxy(2, 24);
    cout << "Press any key to continue...";
    getch();
}
}
void initial::display_mps(void)
{
    fstream file;
    gotoxy(30,1);
    cout<<"MASTER PRODUCTION SCHEDULE";
    int d1, m1, y1;
    struct date d;
    getdate(&d);
    d1 = d.da_day;m1 = d.da_mon;y1 = d.da_year;
    gotoxy(65, 2);
    cout<<"Date:"<<d1<<"/"<<m1<<"/"<<y1;
    gotoxy(1, 2);
    cout<<"Lead Time:"<<lttime;
    gotoxy(1,3);
    cout<<"SStock    ":"<<sstock;
    gotoxy(18,3);
    for (j = 18; j <= 79; j++)
    cout << "=";
    gotoxy(42,4);
    cout<<"Weeks";
    gotoxy(18,5);
    for (j = 18; j <= 79; j++)
    cout << "=";
    gotoxy(1, 4);
    cout << "Code#:"<<last_code();
    gotoxy(18,6);
    cprintf("Balance          1          2          3          4
5");
    gotoxy(1, 7);
    for (j = 1; j <=79; j++)
    cout << "=";
    gotoxy(1,9);
    cout<<"Projected Req:";
    gotoxy(1,10);
    for (j = 1; j <=79; j++)
    cout << "=";
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    int col;unsigned int pr1,pr2,pr3,pr4,pr5;

```



```

while (file.read((char *)this, sizeof(initial)));
{
    // Checks the code no
    if (codeno == cno)
    {
        proj_req1=pr1;proj_req2=pr2;
        proj_req3=pr3;proj_req4=pr4;
        proj_req5=pr5;

    }

    gotoxy(col,9);
    cout <<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<proj_req1;
    col=43;
    gotoxy(col,9);
    cout <<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<proj_req2;
    col=53;
    gotoxy(col,9);
    cout <<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<proj_req3;
    col=63;
    gotoxy(col,9);
    cout <<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<proj_req4;
    col=74;
    gotoxy(col,9);
    cout <<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<proj_req5;

    if (col > 79)
    {
        flag = 1;
        col =20;
        gotoxy(2, 24);
        cout << "Press any key to continue.... ";
        getch();
        clrscr();
    }
}
file.close();
if (!flag)
{
    gotoxy(2, 24);
    cout << "Press any key to continue...";
    getch();
}
}
void initial::display_bom(void)
{
    fstream file;
    gotoxy(30,1);
    cout<<"BILL OF MATERIALS";
    int d1, m1, y1;
    struct date d;

```

```

getdate(&d);
d1 = d.da_day;m1 = d.da_mon;y1 = d.da_year;
gotoxy(65, 2);
cout<<"Date:"<<d1<<"/"<<m1<<"/"<<y1;
gotoxy(1,4);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(2, 6);
cout<<"Code Nos";
gotoxy(20,6);
cout<<"Item Name";
gotoxy(35,6);
cout<<"Level";
gotoxy(42,6);
cout<<"Quantity";
gotoxy(57,6);
cout<<"Lead Time";
file.open("INITIAL.dat", ios::in);
file.seekg(0, ios::beg);
int row=7;
while (file.read((char *)this, sizeof(initial)))
{
    gotoxy(2,row);
    cout<<codeno;
    gotoxy(20,row);
    puts(name);
    gotoxy(37,row);
    cout<<level;
    gotoxy(44,row);
    cout<<num_com;
    gotoxy(59,row);
    cout<<ltime;
    if (row > 23)
    {
        flag = 1;
        row = 7;
        gotoxy(2, 24);
        cout << "Press any key to continue.... ";
        getch();
        clrscr();
    }
}
gotoxy(1, row);
for (j = 1; j <= 79; j++)
cout << "=";
row++;
file.close();
if (!flag)
{
    gotoxy(2, 24);
    cout << "Press any key to continue...";
    getch();
}
}
void initial::display_por2(void)
{
    fstream file;
    gotoxy(30,1);
    cout<<"PLANNED ORDER RELEASE";
    int d1, m1, y1;

```

```

struct date d;
getdate(&d);
d1 = d.da_day;m1 = d.da_mon;y1 = d.da_year;
gotoxy(65, 2);
cout << "Date:" << d1 << "/" << m1 << "/" << y1;
gotoxy(1, 2);
cout<<"Lead Time:"<<ltime;
gotoxy(1,3);
cout<<"SStock      :"<<sstock;
gotoxy(18,3);
for (j = 18; j <= 79; j++)
cout << "=";
gotoxy(42,4);
cout<<"Weeks";
gotoxy(18,5);
for (j = 18; j <= 79; j++)
cout << "=";
material mi;
gotoxy(1, 4);
cout << "Code#:"<<last_code();
gotoxy(18,6);
cprintf("Balance          1          2          3          4
5");
gotoxy(1, 7);
for (j = 1; j <=79; j++)
cout << "=";
gotoxy(1,9);
cout<<"Projected Req :";
gotoxy(1,10);
for (j = 1; j <=79; j++)
cout << "=";
gotoxy(1,11);
cout<<"Sche. Receipt :";
gotoxy(1,12);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,13);
cout<<"On Hand Qty      :";
gotoxy(1,14);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,15);
cout<<"Net Req Qty      :";
gotoxy(1,16);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,17);
cout<<"Pla.O.Receipt :";
gotoxy(1,18);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,19);
cout<<"Plan.O.Release:";
gotoxy(1,20);
for (j = 1; j <= 79; j++)
cout << "=";
file.open("INITIAL.dat", ios::in);
file.seekg(0, ios::beg);
int col,unsigned int pr1,pr2,pr3,pr4,pr5,sr1,sr2,
sr3,sr4,sr5,oh0;unsigned int cno;
while (file.read((char *)this, sizeof(initial)));

```

```

{
    if(codeno==cno)
    {
        proj_req1=pr1;proj_req2=pr2;proj_req3=pr3;
        proj_req4=pr4;proj_req5=pr5;sche_req1=sr1;
        sche_req2=sr2;sche_req3=sr3;sche_req4=sr4;
        sche_req5=sr5;on_hand0=oh0;

    }

    flag = 0;delay(1);
    if(week0==1)
    {
        col=20;
        gotoxy(col,13);
        cout <<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<on_hand0;
    }
    if(week1==2)
    {
        col=33;
        gotoxy(col,9);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<proj_req1;
        col=33;
        gotoxy(col,11);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<sche_req1;
        col=33;
        gotoxy(col,13);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_oh1(cno);
        col=33;
        gotoxy(col,15);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_nrl(cno);
        col=33;
        gotoxy(col,17);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_pol(cno);
        col=33;
        gotoxy(col,19);
        cout <<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_plo1(cno);
    }
    if(week2==3)
    {
        col=44;
        gotoxy(col,9);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<proj_req2;
        col=44;
        gotoxy(col,11);
        cout<<setw(5)<<setprecision(0)

```

```

<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req2;
col=44;
gotoxy(col,13);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_oh2(cno);
col=44;
gotoxy(col,15);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_nr2(cno);
col=44;
gotoxy(col,17);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_po2(cno);
col=44;
gotoxy(col,19);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_plo2(cno);
}
if(week3==4)
{
col=55;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<proj_req3;
col=55;
gotoxy(col,11);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req3;
col=55;
gotoxy(col,13);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_oh3(cno);
col=55;
gotoxy(col,15);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_nr3(cno);
col=55;
gotoxy(col,17);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_po3(cno);
col=55;
gotoxy(col,19);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_plo3(cno);
}
if(week4==5)
{
col=65;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)

```

```

<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<proj_req4;
col=65;
gotoxy (col, 11);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<sche_req4;
col=65;
gotoxy (col, 13);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<compute_oh4 (cno);
col=65;
gotoxy (col, 15);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<compute_nr4 (cno);
col=65;
gotoxy (col, 17);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<compute_po4 (cno);
col=65;
gotoxy (col, 19);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<compute_plo4 (cno);
}
if (week5==6)
{
col=74;
gotoxy (col, 9);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<proj_req5;
col=74;
gotoxy (col, 11);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<sche_req5;
col=74;
gotoxy (col, 13);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<compute_oh5 (cno);
col=74;
gotoxy (col, 15);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<compute_nr5 (cno);
col=74;
gotoxy (col, 17);
cout<<setw(5) <<setprecision(0)
<<setiosflags (ios::left)
<<setiosflags (ios::fixed) <<compute_po5 (cno);
}
if (col > 79)
{
    flag = 1;
    col =10;
    gotoxy(2, 24);
}

```

```

        cout << "Press any key to continue.... ";
        getch();
        clrscr();
    }
}

file.close();
if (!flag)
{
    gotoxy(2, 24);
    cout << "Press any key to continue...";
    getch();
}
}

void initial::daily_report(void)
{
    fstream file;
    gotoxy(30,1);
    cout<<"NET CHANGE REPORT";
    int d1, m1, y1;
    struct date d;
    getdate(&d);
    d1 = d.da_day;m1 = d.da_mon;y1 = d.da_year;
    gotoxy(65, 2);
    cout << "Date:" << d1 << "/" << m1 << "/" << y1;
    gotoxy(1, 2);
    cout<<"Lead Time:"<<ltime;
    gotoxy(1,3);
    cout<<"SStock   :"<<sstock;
    gotoxy(18,3);
    for (j = 18; j <= 79; j++)
    cout << "=";
    gotoxy(42,4);
    cout<<"Weeks";
    gotoxy(18,5);
    for (j = 18; j <= 79; j++)
    cout << "=";
    material mi;
    gotoxy(1, 4);
    cout << "Code#:"<<last_code();
    gotoxy(18,6);
    cprintf("Balance          1          2          3          4
5");
    gotoxy(1, 7);
    for (j = 1; j <=79; j++)
    cout << "=";
    gotoxy(1,9);
    cout<<"Projected Req :";
    gotoxy(1,10);
    for (j = 1; j <=79; j++)
    cout << "=";
    gotoxy(1,11);
    cout<<"Sche. Receipt :";
    gotoxy(1,12);
    for (j = 1; j <= 79; j++)
    cout << "=";
    gotoxy(1,13);
    cout<<"On Hand Qty   :";
    gotoxy(1,14);
    for (j = 1; j <= 79; j++)

```

```

cout << "=";
gotoxy(1,15);
cout<<"Net Req Qty   :";
gotoxy(1,16);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,17);
cout<<"Pla.O.Receipt :";
gotoxy(1,18);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,19);
cout<<"Plan.O.Release:";
gotoxy(1,20);
for (j = 1; j <= 79; j++)
cout << "=";
file.open("INITIAL.dat", ios::in);
file.seekg(0, ios::beg);
int col;unsigned int cno;
unsigned int pr1,pr2,pr3,pr4,pr5,sr1,sr2,
sr3,sr4,sr5,oh0;
while (file.read((char *)this, sizeof(initial)));
{
    if(codeno==cno)
    {
        proj_req1=pr1;proj_req2=pr2;proj_req3=pr3;
        proj_req4=pr4;proj_req5=pr5;sche_req1=sr1;
        sche_req2=sr2;sche_req3=sr3;sche_req4=sr4;
        sche_req5=sr5;on_hand0=oh0;

    }
    {
        col=20;
        gotoxy(col,13);
        cout <<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<on_hand0;
    }
    if (week1==2)
    {
        col=34;
        gotoxy(col,9);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<proj_req1;
        col=34;
        gotoxy(col,11);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<sche_req1;
        col=34;
        gotoxy(col,13);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_oh1(cno);
        col=34;
        gotoxy(col,15);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_nr1(cno);
        col=34;
    }
}

```



```

gotoxy(col,17);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_p01(cno);
col=34;
gotoxy(col,19);
cout <<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_p101(cno);
}
if(week2==3)
{
col=44;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<proj_req2;
col=44;
gotoxy(col,11);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req2;
col=44;
gotoxy(col,13);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_oh2(cno);
col=44;
gotoxy(col,15);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_nr2(cno);
col=44;
gotoxy(col,17);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_po2(cno);
col=44;
gotoxy(col,19);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_p102(cno);
}
if(week3==4)
{
col=54;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<proj_req3;
col=54;
gotoxy(col,11);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req3;
col=54;
gotoxy(col,13);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_oh3(cno);
col=54;

```

```

gotoxy(col,15);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_nr3(cno);
col=54;
gotoxy(col,17);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_po3(cno);
col=54;
gotoxy(col,19);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_plo3(cno);
}
if(week4==5)
{
col=65;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<proj_req4;
col=65;
gotoxy(col,11);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req4;
col=65;
gotoxy(col,13);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_oh4(cno);
col=65;
gotoxy(col,15);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_nr4(cno);
col=65;
gotoxy(col,17);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_po4(cno);
col=65;
gotoxy(col,19);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_plo4(cno);
}
if(week5==6)
{
col=74;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<proj_req5;
col=74;
gotoxy(col,11);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req5;
col=74;

```

```

        gotoxy(col,13);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_oh5(cno);
        col=74;
        gotoxy(col,15);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_nr5(cno);
        col=74;
        gotoxy(col,17);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_po5(cno);
    }
    if (col > 79)
    {
        gotoxy(2, 24);
        cout << "Press any key to continue.... ";
        getch();
        clrscr();
    }
}

file.close();
if (!flag)
{
    gotoxy(2, 24);
    cout << "Press any key to continue...";
    getch();
}
}

void initial::weekly_report(void)
{
    fstream file;
    gotoxy(30,1);
    cout<<"REGENERATIVE REPORT";
    int d1, m1, y1;
    struct date d;
    getdate(&d);
    d1 = d.da_day;m1 = d.da_mon;y1 = d.da_year;
    gotoxy(65, 2);
    cout << "Date:" << d1 << "/" << m1 << "/" << y1;
    gotoxy(1, 2);
    cout<<"Lead Time:"<<ltime;
    gotoxy(1,3);
    cout<<"SStock   ":"<<sstock;
    gotoxy(18,3);
    for (j = 18; j <= 79; j++)
    cout << "=";
    gotoxy(42,4);
    cout<<"Weeks";
    gotoxy(18,5);
    for (j = 18; j <= 79; j++)
    cout << "=";
    material mi;
    gotoxy(1, 4);
    cout << "Code#:"<<last_code();
    gotoxy(18,6);
}

```

```

5");
    cprintf("Balance          1          2          3          4
gotoxy(1, 7);
for (j = 1; j <=79; j++)
cout << "=";
gotoxy(1,9);
cout<<"Projected Req :";
gotoxy(1,10);
for (j = 1; j <=79; j++)
cout << "=";
gotoxy(1,11);
cout<<"Sche. Receipt :";
gotoxy(1,12);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,13);
cout<<"On Hand Qty   :";
gotoxy(1,14);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,15);
cout<<"Net Req Qty   :";
gotoxy(1,16);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,17);
cout<<"Pla.O.Receipt :";
gotoxy(1,18);
for (j = 1; j <= 79; j++)
cout << "=";
gotoxy(1,19);
cout<<"Plan.O.Release:";
gotoxy(1,20);
for (j = 1; j <= 79; j++)
cout << "=";
file.open("INITIAL.dat", ios::in);
file.seekg(0,ios::beg);
int col;unsigned int cno;
unsigned int pr1,pr2,pr3,pr4,pr5,sr1,sr2,
sr3,sr4,sr5,oh0;
while (file.read((char *)this, sizeof(initial)));
{
    if(codeno==cno)
    {
        proj_req1=pr1;proj_req2=pr2;proj_req3=pr3;
        proj_req4=pr4;proj_req5=pr5;sche_req1=sr1;
        sche_req2=sr2;sche_req3=sr3;sche_req4=sr4;
        sche_req5=sr5;on_hand0=oh0;

    }

    if(week0==1)
    {
        col=20;
        gotoxy(col,13);
        cout <<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<on_hand0;
    }
    if(week1==2)
    {

```

```

col=33;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<proj_req1;
col=33;
gotoxy(col,11);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req1;
col=33;
gotoxy(col,13);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_oh1(cno);
col=33;
gotoxy(col,15);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_nr1(cno);
col=33;
gotoxy(col,17);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_po1(cno);
col=33;
gotoxy(col,19);
cout <<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_plo1(cno);
}
if(week2==3)
{
col=44;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<proj_req2;
col=44;
gotoxy(col,11);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req2;
col=44;
gotoxy(col,13);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_oh2(cno);
col=44;
gotoxy(col,15);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_nr2(cno);
col=44;
gotoxy(col,17);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_po2(cno);
col=44;
gotoxy(col,19);
cout<<setw(5)<<setprecision(0)

```

```

<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_plo2(cno);
}
if (week3==4)
{
col=55;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<proj_req3;
col=55;
gotoxy(col,11);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req3;
col=55;
gotoxy(col,13);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_oh3(cno);
col=55;
gotoxy(col,15);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_nr3(cno);
col=55;
gotoxy(col,17);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_po3(cno);
col=55;
gotoxy(col,19);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_plo3(cno);
}
if (week4==5)
{
col=65;
gotoxy(col,9);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<proj_req4;
col=65;
gotoxy(col,11);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<sche_req4;
col=65;
gotoxy(col,13);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_oh4(cno);
col=65;
gotoxy(col,15);
cout<<setw(5)<<setprecision(0)
<<setiosflags(ios::left)
<<setiosflags(ios::fixed)<<compute_nr4(cno);
col=65;
gotoxy(col,17);
cout<<setw(5)<<setprecision(0)

```

```

    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<compute_po4(cno);
    col=65;
    gotoxy(col,19);
    cout<<setw(5)<<setprecision(0)
    <<setiosflags(ios::left)
    <<setiosflags(ios::fixed)<<compute_plo4(cno);
    }
    if (week5==6)
    {
        col=74;
        gotoxy(col,9);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<proj_req5;
        col=74;
        gotoxy(col,11);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<sche_req5;
        col=74;
        gotoxy(col,13);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_oh5(cno);
        col=74;
        gotoxy(col,15);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_nr5(cno);
        col=74;
        gotoxy(col,17);
        cout<<setw(5)<<setprecision(0)
        <<setiosflags(ios::left)
        <<setiosflags(ios::fixed)<<compute_po5(cno);
    }

    if (col > 79)
    {
        gotoxy(2, 24);
        cout << "Press any key to continue.... ";
        getch();
        clrscr();
    }
}

file.close();
if (!flag)
{
    gotoxy(2, 24);
    cout << "Press any key to continue...";
    getch();
}
}

void initial::excep_notice(void)
{
    fstream file;
    gotoxy(30,1);
    cout<<"EXCEPTION NOTICES";
    int dl, ml, yl;

```

```

    struct date d;
    getdate(&d);
    d1 = d.da_day; m1 = d.da_mon; y1 = d.da_year;
    gotoxy(65, 2);
    cout << "Date:" << d1 << "/" << m1 << "/" << y1;
    gotoxy(1, 5);
    cout<<"Lead Time:"<<lttime;
    gotoxy(1,6);
    cout<<"SStock    ":"<<sstock;
    material mi;
    gotoxy(1, 7);
    cout << "Code#:"<<last_code();
    gotoxy(1,8);
    for (j = 1; j <= 79; j++)
    cout << "=";
    file.open("INITIAL.dat", ios::in);
    file.seekg(0, ios::beg);
    unsigned int pr1;
    unsigned int cno;
    while (file.read((char *)this, sizeof(initial)));
    {
        if(codeno==cno)
        {
            proj_req1=pr1;
        }
        gotoxy(2,10);
        cout <<"Print Notice:"<<proj_req1;
    }
    file.close();
    if (!flag)
    {
        gotoxy(2, 24);
        cout << "Press any key to continue...";
        getch();
    }
}
// Main
void main(void)
{
    entry_menu m_menu;
    initgraph(&gdriver, &gmode, "");
    m_menu.screen();closegraph();
    m_menu.control_menu();
}

```

75