

UNIVERSITY OF JORDAN

FACULTY OF GRADUATE STUDIES

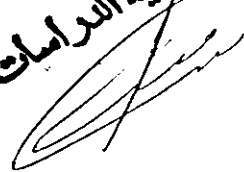
CV 2  
14/5/95

**APPLICATION OF OBJECT-ORIENTED DATABASE FOR  
MATERIAL REQUIREMENT PLANNING**

**NAEL NAIM SHABARO**

**SUPERVISED BY**

**Dr. BASSAM TALHOUNI**

مركز كلية الدراسات العليا  


**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF MASTER OF SCIENCE IN INDUSTRIAL ENGINEERING,  
FACULTY OF GRADUATE STUDIES, UNIVERSITY OF JORDAN**

**AUGUST 1995**

**THIS THESIS WAS DEFENDED SUCCESSFULLY IN THE  
5TH OF AUGUST 1995.**

**COMMITTEE MEMBERS**

**SIGNATURE**

**1- DR. BASSAM TALHUNI**

*Bassam S. Talhuni*

**2- Dr. Abed Raheem Jallad**

*Abed Raheem Jallad*

**3- DR. MUNEEB QUTAISHAT**

*Muneeb Qutaishat*

*To My Family With Love*

## ACKNOWLEDGMENT

In writing this dissertation, one needs help, and I owe my thanks to the many who helped me, in several different ways. I am indebted to my supervisor Dr. Bassam Talhouni for his supervision and for the time he dedicated for me. I wish to acknowledge a particular debt of gratitude to Dr. Khaled Tuqan for his support and encouragement. I would also like to thank Dr. Mahmoud Abu-Ali and Dr. Muneeb Qutaishat for their contribution to my work.

A special thank goes to my friend Yousof Kutkut, who supported me by all his means. My thanks go, also to my friends: Ibraheem Abdoh, Majdi Beit-Shaweesh, Bashar Arafah, Mohammad Owais, and Yousof Abdallat who have been a source of encouragement and help all throughout.

Lastly, but certainly mostly, to my parents, brothers and sisters who have always been a true backup throughout my entire life.

To all of you and to those whom I forgot to mention, *thank you*.

*Nael Shabaro*

# CONTENTS

Committee decision	ii
Dedication	iii
Acknowledgment	iv
Contents	v
List Of Illustrations	viii
List Of Appendices	x
Abstract	xi

## CHAPTER ONE

### INTRODUCTION & LITERATURE REVIEW

[1.1] Order Processing Life Cycle .....	2
[1.1.1] Master Production Scheduling .....	2
[1.1.2] Material Requirement Planning .....	4
[1.1.3] Capacity Requirement Planning .....	4
[1.1.4] Loading And Sequencing .....	5
[1.2] Focus On The Application .....	7
[1.3] Literature Review .....	9
[1.4] Characteristic Of The Needed Solution .....	16
[1.5] Different Approaches To Solve The Problem .....	18
[1.5.1] Hierarchical data model .....	18
[1.5.2] Network data model .....	19
[1.5.3] EER data model .....	19
[1.5.4] Relational data model .....	20
[1.5.5] RM/T data model .....	22
[1.5.6] Functional data model .....	23
[1.5.7] Object-oriented data model .....	24
[1.6] Choosing A Model .....	25
[1.7] The Object-Oriented Approach .....	26
[1.8] Object Modeling Technique.....	29
[1.9] Assumptions .....	30
[1.10] Case study .....	31

## CHAPTER TWO

<b>ANALYSIS</b> .....	32
[2.1] Analysis Methodology .....	32
[2.2] Problem Statement .....	34

[2.2.1] System Input .....	31
[2.2.1.1] The MPS .....	35
[2.2.1.2] Externally Originating Orders .....	36
[2.2.1.3] Forecast Of Independent Demand .....	37
[2.2.1.4] The Inventory Record File .....	37
[2.2.1.5] BOM File .....	38
[2.2.2] Processing Logic .....	40
[2.2.3] Coverage Of Net Requirements.....	43
[2.2.4] Safety Stock .....	44
[2.2.5] Offset Lead-Time .....	45
[2.2.6] Lot-Sizing .....	47
[2.2.7] System Output .....	48
[2.3] Data Modeling .....	49
[2.4] Object Model .....	50
[2.4.1] Objects .....	50
[2.4.2] Classes .....	51
[2.4.3] Object Diagram .....	51
[2.4.3.1] Class diagram .....	52
[2.4.3.4] Instance Diagrams .....	52
[2.4.4] Attributes .....	54
[2.4.5] Operations And Methods .....	54
[2.4.6] Data Dictionary .....	58
[2.4.7] Links And Associations .....	58
[2.4.7.1] Modeling Association As A Class .....	63
[2.4.8] Object Oriented Abstraction .....	64
[2.4.8.1] Identification .....	64
[2.4.8.2] Aggregation .....	65
[2.4.8.3] Generalization .....	72
[2.4.8.4] Selection .....	72
[2.5] Dynamic Modeling .....	78
[2.5.1] Events .....	79
[2.5.2] Scenarios And Event Traces .....	80
[2.5.3] States .....	84
[2.5.4] State Diagrams .....	87
[2.5.5] Conditions .....	90
[2.5.6] Operations .....	90
[2.6] Functional Model .....	93
[2.6.1] Input/Output .....	94
[2.6.2] Data Flow Diagrams .....	95
[2.6.3] Specifying Operations .....	96
[2.6.4] SPECIFYING CONSTRAINTS .....	99
[2.6.4.1] Preconditions .....	99
[2.6.4.2] Postconditions .....	99

**CHAPTER THREE****SYSTEM DESIGN**

[3.1] Breaking The System Into Subsystems.....	101
[3.2] Organization Of Subsystems .....	103
[3.2.1] Vertical Layers .....	103
[3.2.2] Horizontal Partitions.....	104
[3.3] System Topology .....	105
[3.4] Estimating Hardware Requirements .....	106
[3.5] Managing Data Stores .....	107
[3.6] System Architecture .....	107
[3.7] Hardware Trade-off priorities .....	108

**CHAPTER FOUR****RESULTS, DISCUSSION, & RECOMMENDATION**

[4.1] Results .....	109
[4.2] Discussion .....	110
[4.2.1] Data Types .....	112
[4.2.2] Data Integrity .....	113
[4.2.3] Schema Evolution .....	115
[4.3] Recommendation .....	116
<b>REFERENCES</b> .....	117
<b>APPENDIX A</b> .....	120
<b>APPENDIX B</b> .....	122
<b>APPENDIX C</b> .....	127
<b>ARABIC ABSTRACT</b> .....	129

## LIST OF ILLUSTRATIONS

Figure 1.1 Order processing life cycle .....	3
Figure 1.2 Effect of rescheduling frequency on the total cost .....	6
Figure 1.3 Conventional Approaches .....	10
Figure 1.4 The object oriented environment .....	24
Figure 1.5 Proposed Solution .....	28
Figure 1.6 System development life cycle .....	29
Figure 2.1 MRP inputs and outputs .....	35
Figure 2.2 Master production scheduling .....	36
Figure 2.3 External originating orders .....	36
Figure 2.4 Forecast of independent items .....	37
Figure 2.5 Inventory record file .....	38
Figure 2.6 Bill of material .....	39
Figure 2.7 Bill of material case example .....	39
Figure 2.8 Planning horizon .....	41
Figure 2.9 Gross requirement planning .....	42
Figure 2.10 Gross requirement planning case example .....	42
Figure 2.11 Coverage of net requirement .....	44
Figure 2.12 Coverage of net requirement case example .....	44
Figure 2.13 Safety stock case example .....	45
Figure 2.14 Offset lead-time .....	46
Figure 2.15 Offset lead-time case example .....	46
Figure 2.16 Lot-size for certain product .....	47
Figure 2.17 Lot-size case example .....	48
Figure 2.18 The MRP objects .....	50
Figure 2.19 The MRP classes .....	51
Figure 2.20 A class diagram for MRP classes .....	52
Figure 2.21 An MRP instance diagram .....	53
Figure 2.22 Classes with their attributes .....	55
Figure 2.23 Classes with their operations .....	56
Figure 2.24 Classes with their operations case example .....	57
Figure 2.25 Data dictionary .....	59
Figure 2.26 Association of MRP classes .....	61
Figure 2.27 Order class with operations .....	63
Figure 2.28 Identification of MRP classes .....	65
Figure 2.29 Aggregation of parts into assemblies .....	66
Figure 2.30 Aggregation of parts into subassemblies case example .....	66
Figure 2.31 Aggregation of subassemblies into products .....	67
Figure 2.32 Aggregation of subassemblies into products case example .....	67
Figure 2.33 Aggregation of products into product family .....	68
Figure 2.34 Aggregation of products into product family case example .....	68
Figure 2.35 Aggregation of product families into product group .....	69
Figure 2.36 Aggregation of product families into product group case example .....	69
Figure 2.37 Aggregation of machines into work cell .....	70



Figure 2.38 Aggregation of machines into work cell case example .....	70
Figure 2.39 Aggregation of work cells into machine line .....	71
Figure 2.40 Aggregation of work cells into machine line case example .....	71
Figure 2.41 Generalization of MRP objects .....	73
Figure 2.42 Generalization of MRP objects case example .....	74
Figure 2.43 Selection based on critical lead-time .....	75
Figure 2.44 Selection based on critical lead-time case example .....	75
Figure 2.45 Selection based on critical part value .....	76
Figure 2.46 Selection based on critical part value case example .....	76
Figure 2.47 Selection based on both critical lead-time and value .....	77
Figure 2.48 Selection based on both critical lead-time and value case example .....	77
Figure 2.49 Normal order process scenario .....	80
Figure 2.50 Change of order process scenario .....	81
Figure 2.51 Event trace for an order .....	82
Figure 2.52 Event trace for a change in order .....	83
Figure 2.53 Various characterization of order release state .....	85
Figure 2.54 Life cycle for order release .....	86
Figure 2.55 Life cycle for change in order .....	86
Figure 2.56 State diagram for the order class .....	88
Figure 2.57 State diagram for the time period class .....	89
Figure 2.58 State diagram for order class with activities .....	91
Figure 2.59 State diagram for change in order with activity .....	92
Figure 2.60 Data flow diagram for MRP system .....	96
Figure 3.1 Organization of typical MRP subsystems .....	104
Figure 3.2 Data flow diagram for MRP subsystems .....	105
Figure C.1 Complete MRP schema .....	128

**ABSTRACT****APPLICATION OF OBJECT-ORIENTED DATABASE  
FOR MATERIAL REQUIREMENT PLANNING****NAEL NAIM SHABARO****SUPERVISED BY****DR. BASSAM TALHOUNI**

This dissertation applies the object-oriented database technology to inventory data management. The purpose of this research is to provide a framework for the analysis and design phases of the object-oriented data management approach. This approach, when implemented on a commercial object-oriented database language, can promise optimized performance for both hardware and software requirements.

A generic material requirement planning (MRP) system and the object modeling technique (OMT) were chosen to be the domain and the application methodology, respectively. The MRP system was thoroughly investigated. The static, dynamic, and behavior aspects of the system were modeled using the object, dynamic, and functional models. The result was a clear real-world representation of the system enriched with evolving semantics to determine what implementation must do.

## **CHAPTER ONE**

### **INTRODUCTION & LITERATURE REVIEW**

The order processing cycle has been a challenging problem for a long time. The great number of product variety, the complexity of processes that products have to go through, the large quantities of parts and machines used, and the dramatic number of changing parameters, gave the order processing problem its complexity.

Knowing this fact about the order processing problem, makes it a challenging problem that is difficult to handle with efficiency. On the other hand, practitioners have always tried to find a suitable solution that is capable of utilizing their resources efficiently and reducing the overall cost which can lead their industry to profits and good reputation when implementing this solution.

In this chapter, an overview of the different processes an order has to go through, before it is shipped to the customer will be given. Among these processes, the MRP process will be chosen to be the focus of our work. Requirements for these processes will also, be illustrated. Previous work of others who have been trying to satisfy these requirements will be described and evaluated and the needs for a new solution will be disclosed. The characteristics of the needed solution will also, be discussed. Different approaches to meet these requirements will be evaluated. Among them, the object-oriented approach

will be chosen and the reasons behind this choice will be stated. After that, different steps that will be carried throughout this project will be overviewed.

## **[1.1] ORDER PROCESSING LIFE CYCLE**

In any manufacturing system, there are three sources of demand:

- Independent demand coming directly from the customers.
- Forecast demand coming from the sales department.
- Externally originating demand, which is interplant demand.

These three sources of demand represent the total orders for the products. In order to satisfy these orders, the orders handling process must go through the following stages :

- Master Production Scheduling.
- Material Requirements Planning.
- Capacity Requirements Planning.
- Loading and sequencing of feasible plans.

These steps are illustrated in Figure 1.1.

### **[1.1.1] MASTER PRODUCT SCHEDULING**

The Master Product Scheduling (MPS) is found at the front end of any industry. An effective MPS provides the basis for making customer delivery promises, utilizing plant capacity effectively, attaining the firm's strategic objectives as reflected in the production plan, and resolving the trade-offs between manufacturing and marketing.

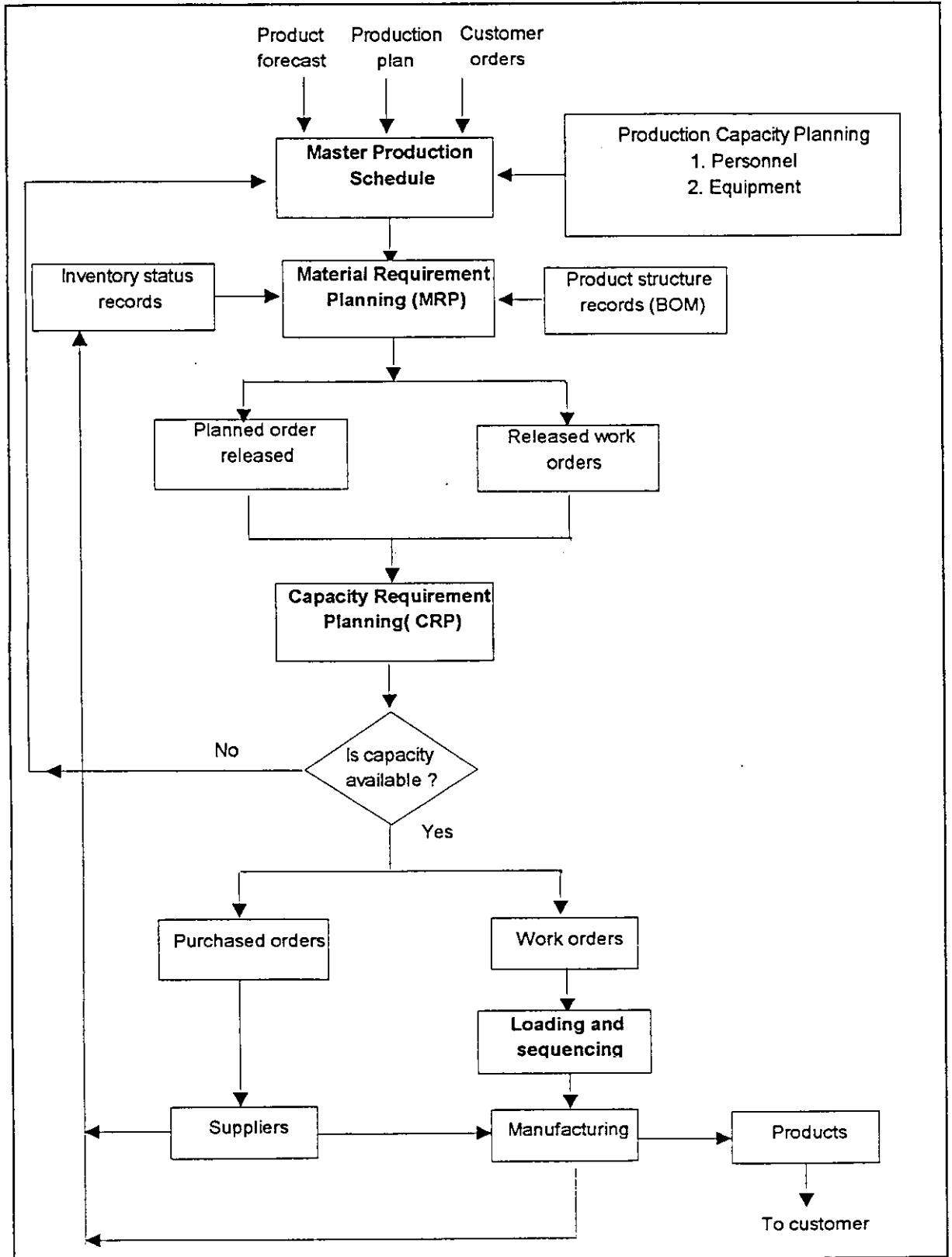


Figure 1.1 Closed loop order process cycle

## **[1.1.2] MATERIAL REQUIREMENTS PLANNING (MRP)**

This is the second step that comes after the master scheduling process. The master scheduling plan assigns due-dates for customer orders, but does not take into account the availability of materials on the required time to satisfy the order. It is the role of the MRP process to determine the exact quantities of materials and when to order them. In the MRP process, different products are required and through the process of requirements explosion, all the necessary parts required to produce these products are calculated and ordered to be available when they are needed to satisfy the due-date for the different products assigned by the MPS.

It is the core of this research to investigate the MRP problem and try to design a decision-support system for this purpose.

## **[1.1.3] CAPACITY REQUIREMENTS PLANNING (CRP)**

Like MRP, capacity availability has to be investigated before a schedule plan is said to be feasible. Capacity has different items such as machines, labor, floor space, energy,...etc. If these items are not available at the time they are required by the schedule plan, then this plan cannot be executed. It is beyond the scope of this research to investigate this problem, so we will not discuss this issue any further.

## [1.1.4] LOADING & SEQUENCING

This process is concerned with the assigning of different products on different machines in a certain sequence of time. The process of scheduling is a very complicated task and takes a long time to generate the scheduling plan using conventional approaches to information systems. This complexity results from the huge variety of products, machines, planning factors, and production parameters found in any industry. These parameters struggle to attain optimal values in order to get the maximum from the same available resources.

In a recent research done by Ajlouni, M. [1], the scheduling problem was investigated on a hypothetical situation. The result of the research showed that the overall performance, in terms of rescheduling cost of the industrial system, improved dramatically with the increase of the frequency of rescheduling. This result is illustrated in Figure 1.2.

It is worth noting that the results indicate that if we were to reduce the scheduling period to a very short time; i.e. hours, a tremendous improvement happens to the system in terms of overall cost.

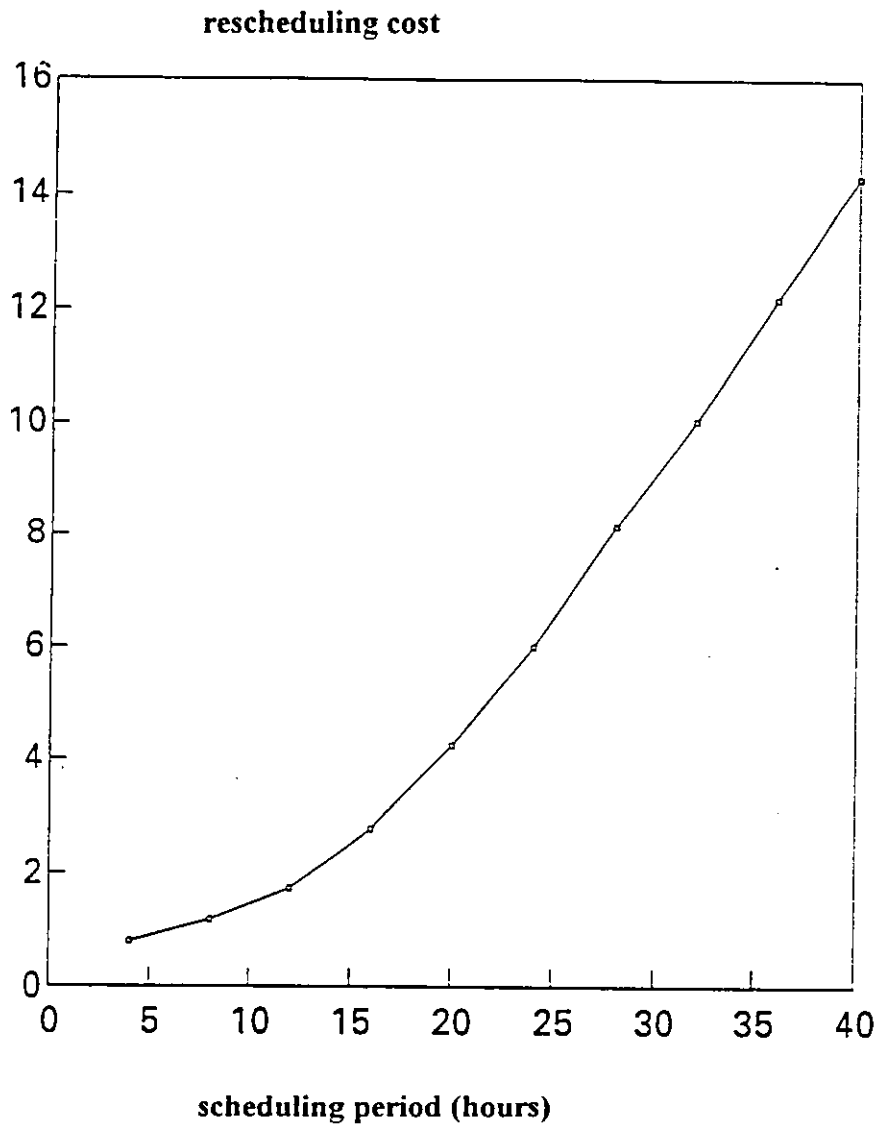


Figure 1.2 Rescheduling cost versus the scheduling period (Reprinted with a permission from [1]).



## [1.2] FOCUS ON THE APPLICATION

In the last section, the order process cycle was shown. From this cycle, the MRP process is chosen to be the area of our research. Because of the importance of this process, it is the crucial step in the order cycle and an efficient solution should be presented for this problem. An efficient solution, means an MRP system capable of processing the required decisions with satisfying frequency.

The appropriate frequency for processing the MRP time phased records depends on the firm, its products, and its operations. The most common practice is weekly processing regeneration. But some firms regenerate every two weeks or even monthly, while others regenerate all the MRP time phased records twice per week or even more often.

The prime motivation for less frequent processing is the computational cost. This can be especially high with regeneration of all MRP records, since a new record is created for every active part in the product structure file at each processing. But computational time required varies considerably from company to company depending on :

- The software environment present.
- The hardware equipment available.
- The complexity of product structure.
- The complexity of operations involved and other factors.

For companies using regeneration, typically 8 to 24 hours of Central Processing Unit (CPU) time is required [2]. For example, at one time, at the Hill Rom Company in

Batesville, Indiana, regeneration for approximately 13,000 active part numbers was done over each weekend, requiring approximately 16 hours on an IBM 370-145 computer [2]. (The IBM 370-145 is an obsolete technology which is replaced by the IBM ES-9000, which is 3-4 times more powerful than the older one [3] ).

455276

The problem with processing less frequently is that the portrayal of component status and needs expressed in the records becomes increasingly out of date and inaccurate. This decrease in accuracy has both anticipated and unanticipated causes. As the anticipated scheduled receipts are received and requirements satisfied, the inventory balances change. As unanticipated scrap, requirement changes, stock corrections, or other such transactions occur, they cause inaccuracies if not reflected in all the time-phased records influenced by the transactions. Changes in one record are linked to other lower level components. Thus, some change transactions may cascade throughout the product structure. If these transactions are not reflected in the time-phased records early enough, the results can be poor planning.

A logical response to the pressure for more frequent processing is to reduce the required amount of calculation by processing only the records affected by the change. This "net change" approach only creates a new part number record when a transaction makes the present component plan inaccurate. Although this can be done as the transaction occurs, typically all transactions are pooled daily and then processed overnight.

The argument for a net change approach is that it can reduce computer time enough to make daily or even more frequent processing possible. Companies where this can be done have the added advantage of smoothing the computational requirements over the

week. On the other hand, daily processing of part of the records could lead to even greater overall computation cost than weekly regeneration. Since only some of the record are reviewed at each processing, there is a need for very accurate computer records and for an accurate transaction processing procedure. Even if this happens, the results will never reach the same optimality as processing the whole plan, because you are trying to append to an existing feasible plan. However, some net change users do an occasional regeneration to clean up all records. The state of the art in hardware and software technology now supports on-line systems with a daily updating cycle. The area of research for this project is to support this idea.

### **[1.3] LITERATURE REVIEW**

During the past period, different efforts have been made to tackle the problem of MRP. Only very few production planning proposals tackle with consistency and concurrency problems in data management. First, file processing was implemented. File processing was completely unsuitable for this purpose, because its processing efficiency was coupled with a complete lack of data independence. After that Database Management Systems (DBMS) were used as logical devices that provide storage and retrieval functions for data and planning procedures. Different approaches like the hierarchical, network or relational were used. Unfortunately, these approaches were unable to perform satisfactorily, especially in cases where huge amounts of data represented in complex data structures and sophisticated planning procedures were implemented. The reason for that misperformance was the complications resulted from trying to simulate order

feasibility on the full level of details of the planning factors found in the shop floor. The situation is illustrated in Figure 1.3

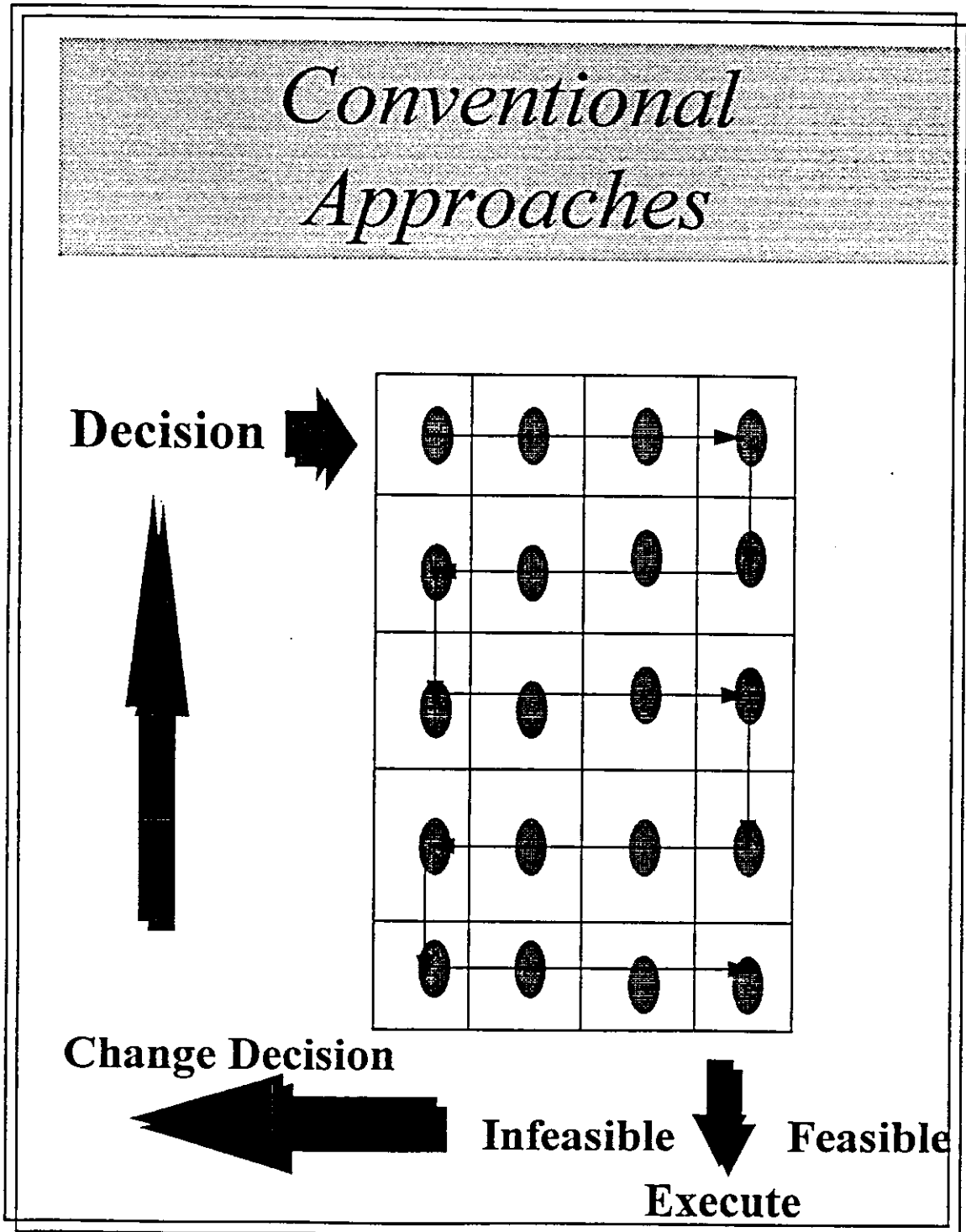


Figure 1.3 Conventional approaches to order processing problem.

Trying to find better solutions, recent researches have tried to utilize database management systems in an active manner. Active database utilization was built around the concept of transforming the planning / control procedure into a relational data structure to be implemented by relational database management systems. This transformation aims at integrating database applications to production planning and control, so that the design and the implementation be both general and theoretically consistent. In doing so, a noticeable change occurred, but problems of impedance mismatch as well as problems of shared objects remained unsolved.

Following, we try to review some of the past experiences that, researchers as well as companies have gained through their search for an appropriate solution:

Baid, and Nagarur [4] integrated the decision support systems for the flexible manufacturing systems using intelligent simulation. In their work an integrated decision support system for a manufacturing system, considering the three-level interdependent planning hierarchy; which is part-subassembly-product was presented. Decision making processes were identified for all the management planning levels. Master production schedule, bill of materials, and inventory records generated the material requirements plan, and in turn the schedule of arrival of the parts in the shop. The decision support system was designed to support various kinds of decisions at all levels using simulation as the main technique. A generic simulation model was developed for the flexible manufacturing system. Simulation was designed to be of an intelligent type for proper integration.

Muller, Jackman, and Fitzwater [5] simulated the work order release mechanism for a flexible manufacturing system. A simulation model and real-time interface module for an operational flexible manufacturing system (FMS) were developed to evaluate work order release sequences on real-time basis. Unlike most simulation studies, the evaluation is based on the transient behavior of the system and not steady-state performance. The time window in which a work order is predicted to be completed is considered in order to determine if a particular work order sequence meets due-date requirements set forth by the MRP system. In addition, results of the simulation can be used to schedule the building of fixtures and the loading of raw material. The model was successfully interfaced with a real-time control database so that the initial conditions could be determined. The results gave better work order release decisions.

Mori, Masatoshi, and Sennosuke [6], described an improvement to the MRP which is integrated into Computer Integrated Manufacturing (CIM). Latest sales information and production plan information are reflected by CIM in the production control system, in order to realize timely production. The production control system considered was able to respond to prompt changes in the market of the manufacturing industry.

Luk, and Albert [7], described a MRP package called DBSI. This package was implemented via the Fujikama 486-33C model running under SCO UNIX System V.3.2. The integrated package covered all aspects of manufacturing, inventory, sales, accounting, shipping, receiving, purchasing, cost of materials, pricing, and billing of materials. With a UNIX network file system and remote file system support, the system design was able to log on and run applications from a terminal to more than one server.

Li, Rong-Kwei, Yu-Tang, and sadashiv [8], described a heuristic rescheduling algorithm for computer-based production scheduling systems. In their work, a new rescheduling algorithm to address the need for rescheduling resulting from the changes that take place on the factory shop-floor was proposed. The algorithm was based on the construction of a scheduling binary tree and the net change concept was adopted from the MRP system. It revises schedules by rescheduling only those operations that need to be revised. This approach was embedded in an existing simulation-based scheduling system to improve its effectiveness.

Nakashima, Youei, Noriaki, and Satoru [9], integrated an expert system with object-oriented database management system. In their work Artificial Intelligence (AI) techniques were used to develop and integrate a scheduling domain shell with an object-oriented database management system for applications to the field of planning in production control. This system realized a model based on the actual production structure in the database and dynamic scheduling in order to have closed control of information in production. In addition meta-information descriptions and constraints processing mechanisms related to constraints in object-oriented DBMS were developed. By doing this, a knowledge and database can be constructed and integrated with AI techniques.

Chung, Fischer, and Gary [10], illustrated the manipulation of object-oriented databases for the structure of a bill of materials. In their work, the basic concepts of applying an object-oriented database (OODB) system, called ORION, to the management of a bill of material was addressed. This illustration attempted to demonstrate the linkages between

the different manufacturing environments and the production planning environment with specific reference to the MRP system. The structure of a composite object made up of objects is the major consideration of the object-oriented BOM (OOBOM). The concentration is on composite objects in the OOBOM schema evolution, their propagation of changing object instances, and the unique object identity of each manufacturing part. The proposed OOBOM is rudimentary, and may be imbedded by using the Itasca TM system, which is a commercial OODB whose prototype is ORION.

Sayed Kamal [11], presented an evaluation of several generic software systems such as spreadsheets and databases. Symphony and Smart were used to develop viable and useful systems for manufacturing planning and control. Both systems offer an easy entry point for managers yet have considerable power in the hands of 'experts'. Symphony was used to create a MRP system with the potential for use in small manufacturing companies. Smart was used to develop a sophisticated interactive production management system which has been successfully implemented. Both systems showed good results for small companies, but for the purpose of large companies, alternative approaches have to be found out.

Chung, Fischer, and Gary [12], described a conceptual data model that integrated elements of semantic relationships with object-orientation concepts to develop a data model for a Bill of Materials (BOM). The semantic relationships included Referencing, Owns and Composed-of, as well as their reversed relationships Referenced-By, Owned-By, and Part-Of. The object orientation concepts contained features of object-oriented programming such as data abstraction and inheritance. A BOM system is one of



the major inputs to the planning and control of the manufactured products. A product has many part subassemblies, which have further subassemblies, and so on. Raw materials are represented by 'leaves' of the BOM system. A structure of the BOM can be regarded as an abstraction hierarchy of an object-oriented data model., and from this point of view, the proposed conceptual BOM data model, named OOBOM system was demonstrated from a sample product.

Hatfield [13], applied the Artificial Intelligence (AI) tools in chemical process planning. The process equipment and raw material requirements for a batch chemical process were obtained through batch process modeling. Object-oriented classes were developed and used to model chemicals, chemical mixtures, process equipment, and basic production facilities. These object-based models were interconnected through relational data structures to form an efficient composite modeling network. A natural language interface was developed integral to the process simulator and was used for implementing process recipe instructions during a process simulation session. A composite object schema was used for modeling data and information regarding the existing production facility at different levels of abstraction. The factory data structure was made to consist of production equipment modules which in turn were designed containing functional equipment groups such as tanks, condensers, and pumps. Equipment groups were further divided into individual equipment items. Other data modeling objects were created for providing import links to equipment specific vendor databases. The complete factory model was constructed, and was used in identifying new equipment requirements through composite modeling by modifying its internal structure as needed to accommodate the equipment requirements for a new chemical process. A planner was

developed, in conjunction with a graph search technique, for generating different equipment allocation plans using an objective function based on the new equipment cost and the material requirements plans.

Winter [14], investigated the utilization of an active integrated database for the vertical integration of production planning was investigated. In this work, it was shown that abstraction and planning procedures can be transformed into derived relations. Based on abstraction hierarchies that consist of derived relations, a search procedure was presented that enable the planner to generate feasible plans with minimal backtracking. Feasibility analysis was supported by a reusable analysis pattern that can be applied for many different planning and control tasks at an arbitrary level of detail. It was shown also, that all abstraction and many disaggregation procedures could be implemented by means of relational data structures. Consistency issues and the role of set-oriented processing were also discussed.

#### **[1.4] CHARACTERISTICS OF THE NEEDED SOLUTION**

To be more specific about this dilemma, the problems of using conventional approaches to combine both, the planning problem and the decision support systems can be summarized as follows:

- Order feasibility cannot be simulated on-line, because of the unacceptable response time for order entry decisions found in most existing systems.

- Plans generated by the planning system may turn out to be infeasible. Even in cases, where the sources of infeasibility can be identified, the entire planning process has to be repeated at full level of details, because of the lack of abstraction capabilities.

In my opinion, the MRP problem should be viewed as part of production planning and control problem. Planning and control tasks can be interpreted as a complex manipulation of huge amounts of data in complex data structures. Therefore, design and implementation of planning / control systems should focus on a solution to the data management problem instead of focusing on dedicated solutions to some special subproblem.

In order to overcome these problems, the proposed solution should have the following features:

- It can handle huge amount of data represented in complex data structures.
- Response time for changes as well as execution time is minimal.
- The availability of built-in abstraction procedures.
- The ability to closely associate data with the planning procedures to avoid inconsistency and interface mismatch

Trying to look for such a system, one can think about appending the abstraction capabilities suggested by the artificial intelligence to the existing powerful relational database management system. In doing so, it is true that part of the problem can be

---

The second rule is an obvious limitation of a hierarchical data structures. Although this problem could be solved, redundancy and a lot of processing overhead is still

required. *Today, the hierarchical data model is used exclusively with hierarchical data base management system and such systems have been phased out.*

### **[1.5.2] NETWORK DATA MODEL**

It was stated in the hierarchical data model that the single-parent rule forces redundant and excessive data and structure. When this rule can be violated, we can create a network data model and further eliminate redundancy. The network model permits as much or as little structure as desired. We can create a hierarchy (a special case of network) if that is needed. As with the hierarchical data model, if a certain relationship is not explicitly included in the database definition, then it cannot be used by a DBMS in processing the data. *Examples of the network DBMS are : IDMS, PRIME DBMS, DBMS-170, DBMS 11, IDS II AND DBMS-1100 [16].*

### **[1.5.3] EXTENDED ENTITY-RELATIONSHIP DATA MODEL (EER)**

The data models used in modern database management systems, hierarchical and network models have proved to be effective schemes for organizing data for convenient and efficient processing. With few limitations (especially for hierarchical), data and a wide variety of relationships can easily be modeled using these architectures. This means that at the internal data model level, we can build databases to support data processing efficiently for a wide range of data, as well as to support query and maintenance processing. On the other hand, for enterprise, external, and conceptual data models, it is better to use description notations and conventions that

are independent of the particular DBMS. In this way, a database designer can specify comprehensive database requirements without bias toward the use of a particular DBMS.

Today, the data model that is most commonly used for enterprise, external, and logic data analysis is the entity-relationship data model. Many Computer-Assisted Software Engineering (CASE) tools and database analysis aids include this model convention, and much attention has been given to it, with many extensions proposed since the original version in 1976 such as the Extended Entity-Relationship (EER) model. The EER model concentrates on conceptual modeling and provides a convenient diagrammatic technique for representing data models.

#### **[1.5.4] RELATIONAL DATA MODEL**

The relational data model was conceived by Codd in a series of papers published in the early 1970s [15]. The model has a sound theoretical foundation, based on the mathematical theory of relations and first order logic, but presents the user with a simple view of the data in the form of a two-dimensional table. The column of the tabular relation represents attributes. Each attribute has a distinct name, and is always referenced by that name, never by its position. The relation notion of a domain is closely related to the concept of a data type in programming languages. Attribute values in a relation must be single, non-decomposable data items. Each row of a tabular relation must be distinct. Thus no two rows may agree on all their attribute values. The relational data model is different from other models not only in architecture, but also in the following ways:

### [1.5.5] RELATIONAL DATA MODEL RM/T

The relational data model offered a significant step forward in data modeling, moving, as it did, away from the machine-oriented, pointer based techniques which symbolizes the older network and hierarchical models. At the same time it offered a realistic framework for efficient implementation as evidenced by the large number of successful relational database management system now in the marketplace. However, inadequacies in the relational model quickly became apparent when it was applied to complex, highly structured application domains. Within the simple framework offered by the model it was impossible to represent and maintain control over situations involving large numbers of interrelated, complex entities. The problem lies in the fact that the relational model is incapable of representing much of the semantic information in real-world applications [15].

Recognizing the inadequacies of the relational model, Ted Codd, the founder of the relational model defined a new semantic data model called RM/T. In RM/T a database is considered to consist of a set of entities that represent the objects and concepts, and their interrelationships, present in the real world situation. That is, relationships are modeled as entities (unlike the EER model, in which relationships are a separate concept). Each entity has a set of associated properties, and entities may be manipulated by a set of operations consisting of create, update and delete entities. Also, all entity types are classified as either associative, characteristic, kernel or designative.

Although, the RM/T model provides an additional semantic at the conceptual level, as in the case with the EER model, RM/T maps easily to the pure relational model at a lower level of abstraction. This of course has many practical advantages considering the extensive availability of efficient relational database management system. It is important to say that, the RM/T model extends the conventional relational model to capture more of the semantics of real-world applications. It retains the simple, tabular representation format of the relational model but provides enhanced facilities for representing entities and relationships. In many respects RM/T can be viewed as combination of the relational model and the entity-relationship model. Further more the RM/T model incorporates some of the features of the object-oriented approach to data modeling.

#### **[1.5.6] FUNCTIONAL DATA MODEL**

The functional data model defines entities by means of the functions which may be applied to them [15]. The relationship between an entity set and an attribute is a function that maps the entities in that set into the domain of the attributes. The representation of the entity is left to the implementor and is defined at a lower level of abstraction. The functional description of the system may be schematically represented in a diagram which is similar to the entity-relationship diagram. The difference between them is that relationships among entities or between entities and their attributes, are represented by single-valued and multi-valued functions. The functional data model is characterized by inherent simplicity, but incorporates a high

level of semantic integrity. This can have a significant advantage for representing complex relationships among entities.

### [1.5.7] OBJECT-ORIENTED DATA MODEL

The object-oriented data model, like the relational model, is not a diagrammatic model but a definitional one. The purpose of the object-oriented model is to capture all meanings of data and to embed this as integrity and structural clauses in the database definitions. The main characteristics of that model is its ability to handle such concepts like class-subclass, aggregation, dynamic properties and structure, and handling objects of different types (images, voice prints, as well as text and data). A major goal of the object-oriented data model is to model the behavior, not just the structure of data.

Objects interact by understanding certain messages, which are passed between objects. A message, together with any arguments in the message, activates a procedure. The object reacts to the message by executing the appropriate procedure and returning a message in response [16]. This scenario is depicted in Figure 1.4.

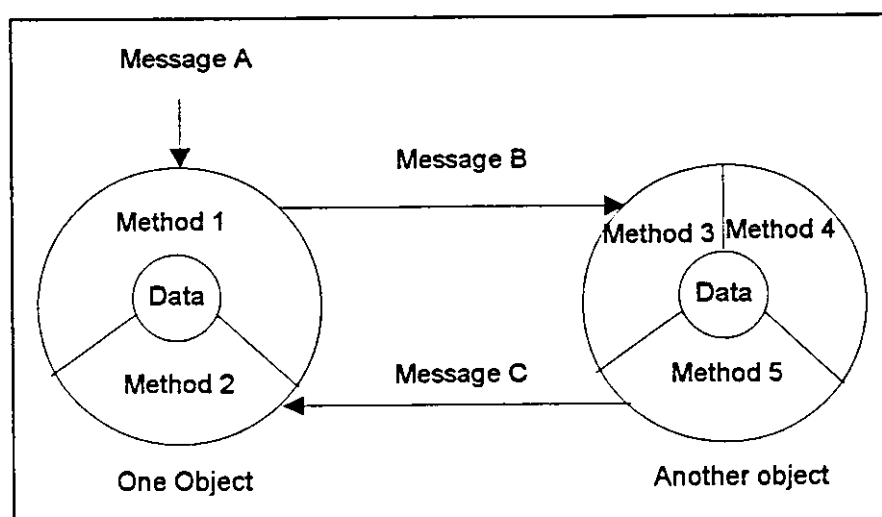


Figure 1.4 The object-oriented environment for objects, methods, and data.



## [1.6] CHOOSING A DATA MODEL

The diversity of data models presented previously makes it difficult to select an appropriate data model. No single choice is best in all situations; in fact, evidence suggests that the best choice is to use several data models, each for different purposes.

For MRP, this system is characterized by huge amounts of data with complex objects present in the system, a great product variety, and large quantities of parts and machines involved. Also, there is a need for abstraction capabilities so that orders feasibility can be simulated on-line. For these factors and for the following reasons, the object-oriented data model stands best to satisfy these requirements:

1. It tackles more challenging problem domains. Object-Oriented Analysis (OOA) brings extra emphasis to the understanding of the problem domain.
2. It improves analyst and problem domain expert interaction. OOA organizes analysis and specification using the methods of organization which pervade people's thinking.
3. It increases the internal consistency of analysis results. OOA reduces the bandwidth between analysis activities, by treating Attributes and Activities as an intrinsic whole.
4. It explicitly represents commonality. OOA uses inheritance to identify and capitalize on commonality of Attributes and Services.
5. It builds specifications resilient to change. OOA packages volatility within problem-domain constructs, providing stability over changing requirements and similar systems.

6. It reuses analysis results, accommodating both families of systems and practical tradeoffs within a system. OOA organizes results based upon problem domain constructs, for present reuse and for future reuse.
  7. It provides a consistent underlying representation for analysis (what is to be built) and design (how is to be built this time). OOA establishes a continuum of representation, for systematically expanding analysis results into a specific design.
- For this, the object-oriented data model will be chosen for analysis and design.

## **[1.7] THE OBJECT-ORIENTED APPROACH**

In this project, the order processing problem is tackled from a different perspective, which is the object-oriented approach. This approach is built around objects, which is a real-world simulation. As opposed to the relational or any other schema, the object-oriented approach tries to bring the computer utilities to the concepts of the real-world, while the other conventional approaches try to transfer the existing real-world system to the world of computer. In the latter case, much of the significant requirements are sacrificed. Benefiting from these advantages, we can look at the system as if it were composed of individual objects encapsulating all the data they require inside the boundaries of this castle, which is the object. Performing all the necessary operations inside this castle, and inheriting all the shared genetic curricula from their ancestors. In this way, one can get a system that is flexible, easy to maintain, extendible and having a 3-4 times better speed performance than any other existing system using the relational database concept.

In this approach, the order process problem will be viewed as a sequence of the following steps:

1. Set the level of abstraction to a maximum.
2. Take a decision.
3. Simulate decision feasibility on the maximum level of abstraction.
4. If decision is feasible, reduce the level of abstraction to a minimum and execute the decision.
5. If decision is infeasible, modify decision and repeat step 3.

These steps are illustrated in Figure 1.5

In this project, The levels of abstractions will be controlled through the object-oriented abstraction hierarchies. The decision will be regarding material requirements planning. In order to carry out these steps, the object modeling technique (OMT) [17] will be used for system analysis and design.

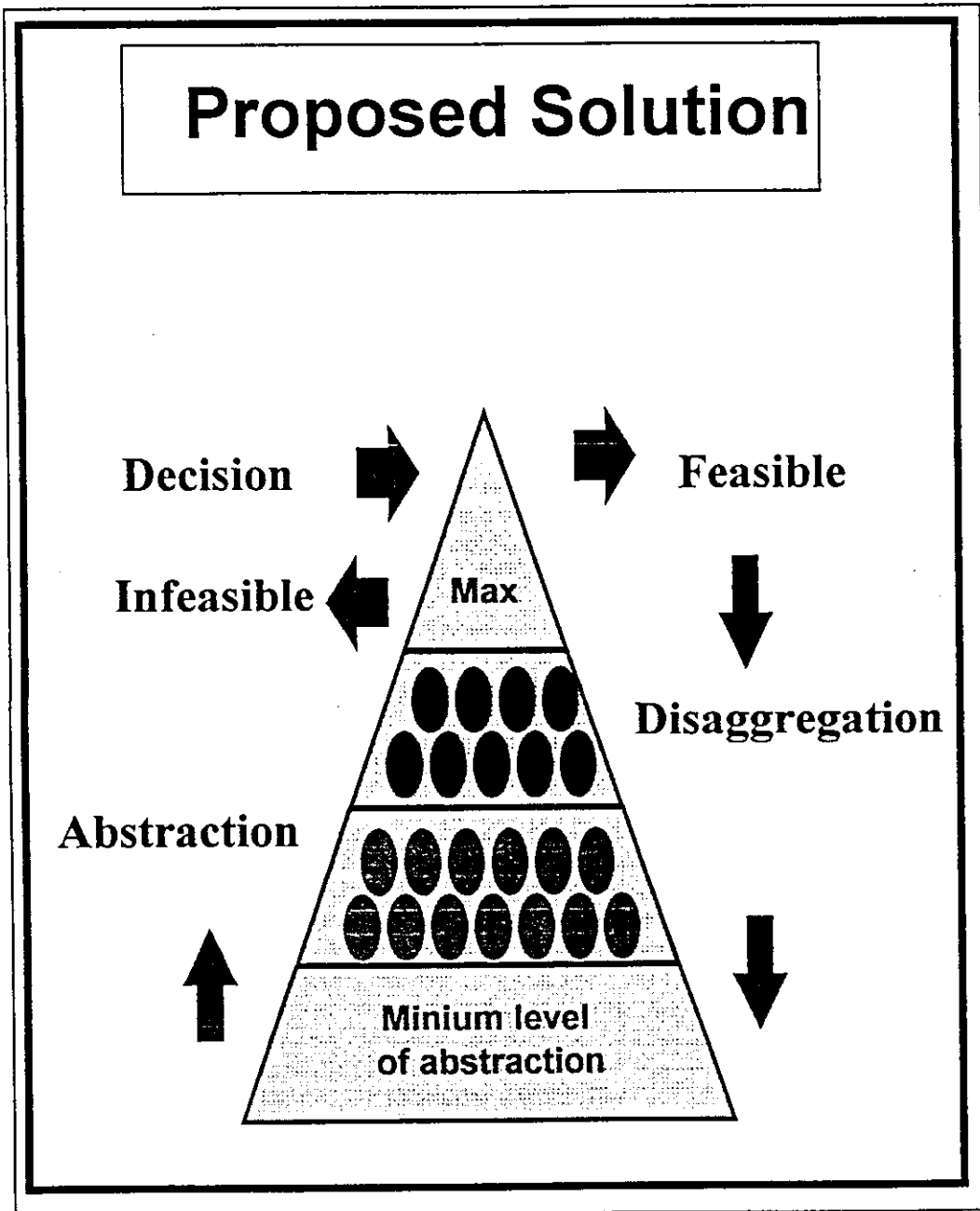


Figure 1.5 Proposed solution.

## [1.8] OBJECT MODELING TECHNIQUE (OMT)

According to the Object Modeling Technique, the system development life cycle consists of the following stages:

- System analysis.
- System design.
- Object design.
- Implementation.

These steps are illustrated in Figure 1.6.. After these steps are completed, the steps of testing and maintenance follow. It is the scope of this project, to carry out the first three stages; which are system analysis, system design, and object design. After carrying these steps out, the system will be ready for implementation using any commercial object-oriented database language such as : Gemstone V4.0, Smalltalk, or Paradox. Notations used in this technique are illustrated in appendix b.

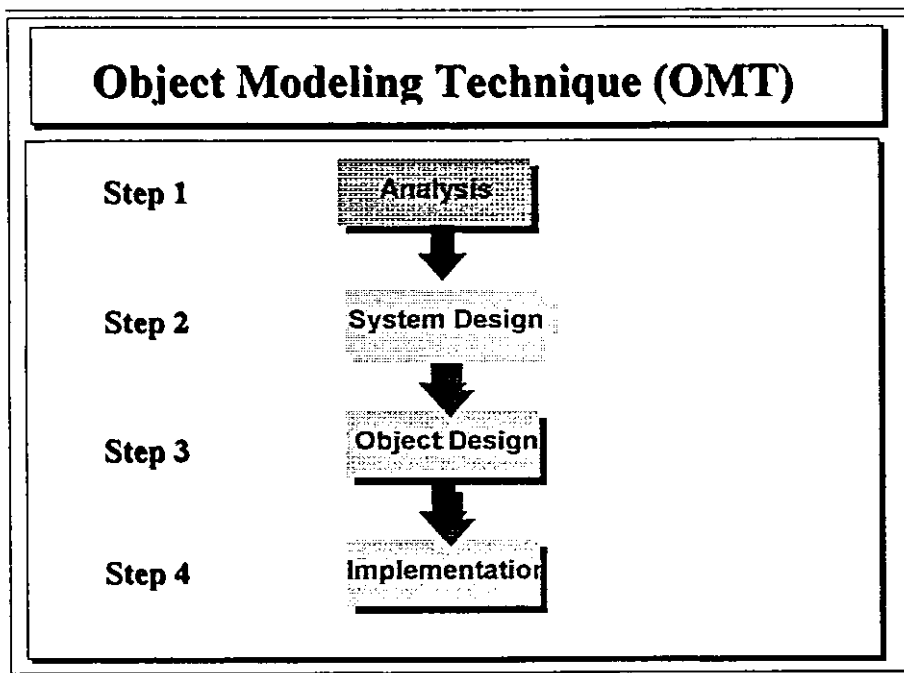


Figure 1.6 System development life cycle.


## [1.9] ASSUMPTIONS

- A Master Production Schedule (MPS) exists and can be stated in Bill of Material (BOM) terms.
- All inventory items are uniquely identified.
- A BOM exist at planning time.
- Inventory records containing data on the status of every item are available.
- Integrity of file data.
- Individual item lead times are known.
- Individual item lot-size is known.
- Individual item safety stock is known.
- Every inventory item goes in and out of stock.
- All of the components of an subassembly are needed at the time of the subassembly order release.
- Discrete disbursement and usage of component materials.
- Process independence of manufactured items.
- Materials found in any industry can be classified into : parts, subassemblies, and products.

## [1.10] CASE STUDY

In order to bring more emphasis to the concepts illustrated in this project, examples from real-world industry will accompany our discussion. All examples are taken from Whiterose Food Company. Whiterose is a private company producing mainly : chocolate, biscuits, wafers, toffee, gum, snakes and candies. These groups of products have many product brands, sizes, shapes, tastes, and colors. In producing these products, different raw materials, machines, and processes are required. Sources of demand for different products can be classified to customers, forecasts, and external orders. Raw materials go through different processes to become subassemblies and subassemblies require further processing before they are products. Same materials may be used to produce different products. Machine lines are found to produce certain product groups and in some cases cross interfaces among different groups are possible.

It is obvious from the above description that, problems found in this industry are typical manufacturing problems that require effective solutions. In the next sections, examples from this industry related to the analysis and design procedures will be provided as necessarily. Among the different product groups existing in this company, the snacks line will be the focus of our discussion. This line produces different products of extruded and fried Flips such as Cheese Extruded Flips (Cheese E.), Tomato Extruded Flips (Tomato E.), Paprika Extruded Flips (Paprika E.), Pizza Extruded Flips (Pizza E.), Chicken Extruded Flips (Chicken E.), Falafil Extruded Flips (Falafil E.), and Onion Extruded Flips (Onion E.).

To facilitate referring to these examples, the following sign will be placed on the top of each example  CASE EXAMPLE.

# **CHAPTER TWO**

## **ANALYSIS**

The goal of analysis is to develop a model of what the system will do. The model is expressed in terms of objects and relationships, dynamic control flow, and functional transformations. The process of capturing requirements and consulting with requester should continue throughout analysis.

### **[2.1] ANALYSIS METHODOLOGY [17]**

1. Write or obtain an initial description of the problem (Problem statement).
2. Build an object model
  - Identify object classes.
  - Build a data dictionary containing the descriptions of classes, attributes, and associations.
  - Add associations between classes.
  - Add attributes for objects and links.
  - Organize and simplify object classes using inheritance.
  - Test access paths using scenarios and iterate the above steps as necessary
  - Group classes into modules, based on close coupling and related functions.



=> Object Model = Object Model Diagram + Data Dictionary.

### 3. Develop a Dynamic Model

- Prepare scenarios of typical interaction sequences.
- Identify events between objects and prepare an event trace for each scenario.
- Prepare an event flow diagram for the system.
- Develop a state diagram for each class that has important dynamic behavior.
- Check for consistency and completeness of events shared among the state diagrams.

=> Dynamic Model = State Diagrams + Global Event Flow Diagram.

### 4. Construct a Functional Model

- Identify input and output values.
- Use data flow diagram as needed to show functional dependencies.
- Describe what each function does.
- Identify constraints.
- Specify optimization criteria.

=> Functional Model = data flow diagrams + constraints.

### 5. Verify, iterate, and refine the three models

- Add key operations that were discovered during preparation of the functional model to the object model. Do not show all operations during analysis as this would clutter the object model; just show the most important operations.

- Verify that the classes, associations, attributes, and the operations are consistent and complete at the chosen level of abstraction. Compare the three models with the problem statement and relevant domain knowledge, and test the models using scenarios.
- Develop more detailed scenarios (including error conditions) as variations on the basic scenarios. Use these “What-if?” scenarios to further verify the three models.
- Iterate the above steps as needed to complete the analysis.

=> Analysis Document = Problem Statement + Object Model + Dynamic Model + Functional Model.

## **[2.2] PROBLEM STATEMENT**

The Material Requirements Planning (MRP) system consists of logically related procedures, decision rules, and records designed to translate a Master Production Schedule (MPS) into time phased net requirements and the planned coverage of such requirements for each component inventory item needed to implement this schedule. An MRP system replans net requirements and coverage as a result of changes in either the MPS, or inventory status or product composition [18].

### **[2.2.1] SYSTEM INPUT**

The MRP system receives inputs from the following sources:

1. The Master Production Schedule (MPS).
2. Orders from components originating from sources external to the plant using the system.
3. Forecasts for items subject to independent demand.
4. The inventory record (item master) file.
5. Bill of material (product-structure) file.

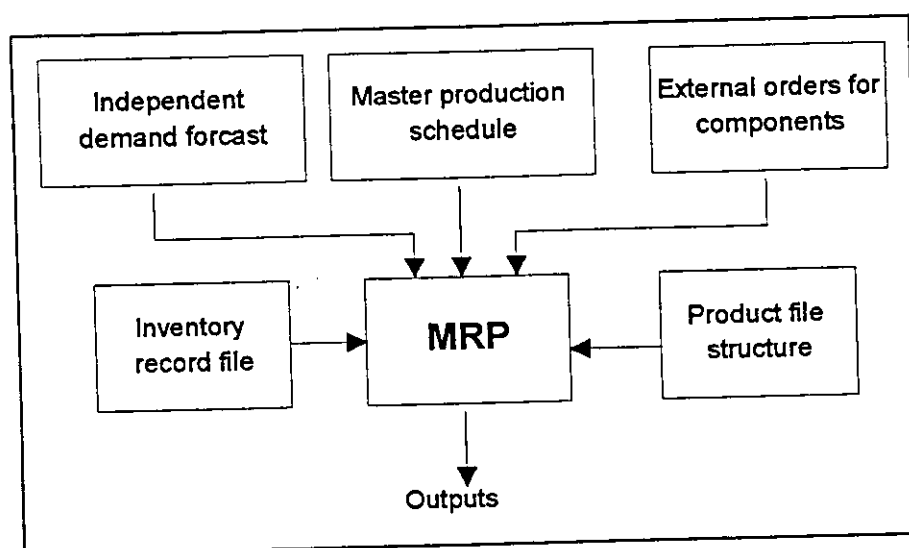


Figure 2.1 MRP inputs and outputs.

### [2.2.1.1] THE MASTER PRODUCTION SCHEDULE (MPS)

The MPS expresses the overall plan of production. It is stated in terms of end items, which may be either (shippable) products or highest-level assemblies from which these products are eventually built in various configurations, according to a final subassembly schedule.

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

<i>Period</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
Cheese E.	500	120	-	500	200	-	300	50	-
Onion E.	-	250	220	100	150				
Paprika E.	200	230	230	200	-	300	-	-	-
Pea-nut E.	100	400	330	300	150	100	245	100	100
Tomato E.	-	120	200	-	200	400	200	240	200

Figure 2.2 Master production scheduling case example.

### [2.2.1.2] EXTERNALLY ORIGINATED ORDERS

These orders are for components include service-part orders, interplant orders, original equipment manufacturer (OEM) orders by other manufacturers who uses these components in their products, and any other special-purpose orders not related to the regular production plan. Components may be ordered for purposes of experimentation, destructive testing, promotion, equipment maintenance, etc. The MRP treats orders of this category as additions to the gross requirements for the respective component items.

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

<i>Period</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
Cheese E.	5	5	5	5	5	5	5	5	5
Onion E.	2	2	2	2	2	2	2	2	2
Paprika E.	4	4	4	4	4	4	4	4	4
Pea-nut E.	3	3	3	3	3	3	3	3	3
Tomato E.	2	2	2	2	2	2	2	2	2

Figure 2.3 Externally originating order case example.

### [2.2.1.3] FORECASTS OF INDEPENDENT DEMAND

MRP system can be programmed to perform this type of demand. By means of applying some statistical forecasting techniques, quantities for future demand can be estimated and treated as item gross requirements by the MRP system.

#### WHITEROSE CASE EXAMPLE

<i>Period</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>4</i>	<i>5</i>	<i>6</i>	<i>7</i>	<i>8</i>	<i>9</i>
Cheese E.	200	220	200	500	200	-	300	50	-
Onion E.	100	200	120	100	150	200	300	200	20
Paprika E.	100	200	130	200	200	300	200	100	100
Pea-nut E.	140	450	230	300	150	100	245	100	100
Tomato E.	20	-	500	100	120	200	300	250	190

Figure 2.4 Forecast of independent demand case example.

### [2.2.1.4] THE INVENTORY RECORD FILE

The inventory record file, also called master item file, comprises the individual item records containing the status data required for the determination of net requirements. The file is kept up to date by the posting of inventory transactions which reflects the various inventory events taking place. Each transaction ( stock receipt, disbursement, scrap, etc.) changes the status of the respective inventory item. The reporting of transactions therefore constitutes an indirect input to the MRP system. Transactions update items status, which is then consulted and modified in the course computing requirements.

In addition to the status data, the inventory records also contain so-called planning factors used principally for the determination of the size and timing of planned orders. Planning factors include item lead time, safety stock (if any), scrap allowances, lot sizing algorithms, etc. Planning factor values are subject to change at the system user's discretion. A change in one or more planning factors normally changes inventory status.

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

<i>Item</i>	<i>Status</i>	<i>Lead time</i>	<i>Safety stock</i>	<i>Lot-size</i>
Cheese E.	130	2	200	250
Onion E.	200	2	100	250
Paprika E.	300	2	100	250
Pea-nut E.	120	2	50	250
Tomato E.	200	2	50	250

Figure 2.5 Inventory record file case example.

#### [2.2.1.5] THE BILL OF MATERIAL FILE

The bill of material file, also known as the product structure file, contains information on the relationships of components and subassemblies, which are essential to the correct development of gross and net requirements. The bill of material plays a passive role in the requirements computation.

In our system the end item is seen as multi-level product structure and each level contains information on the required parts and their quantities. The following diagram shows the hierarchy of bills of materials:

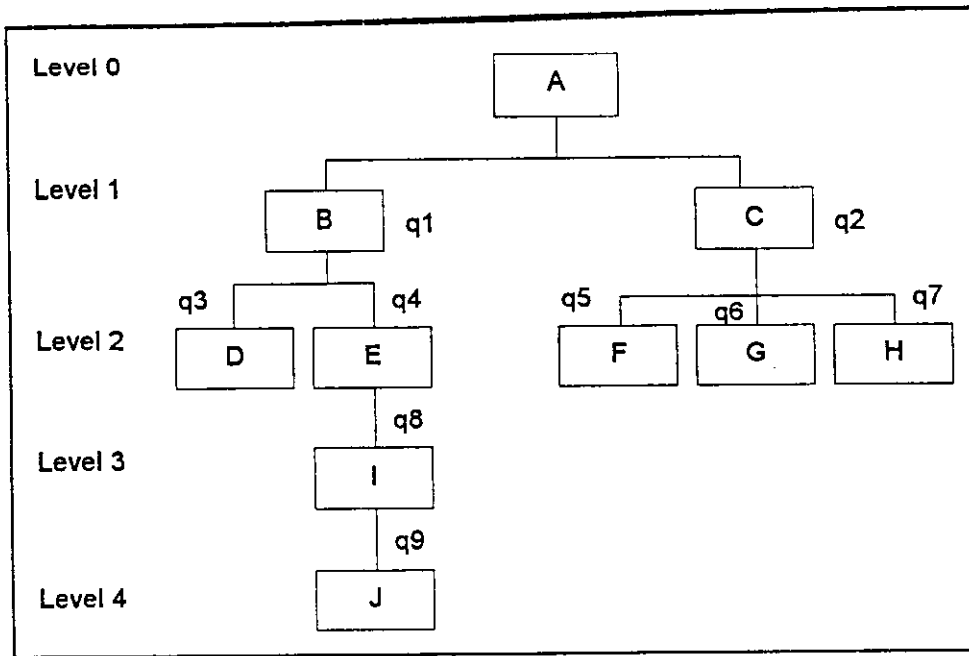


Figure 2.6 Bill of material form.

**WHITEROSE** CASE EXAMPLE

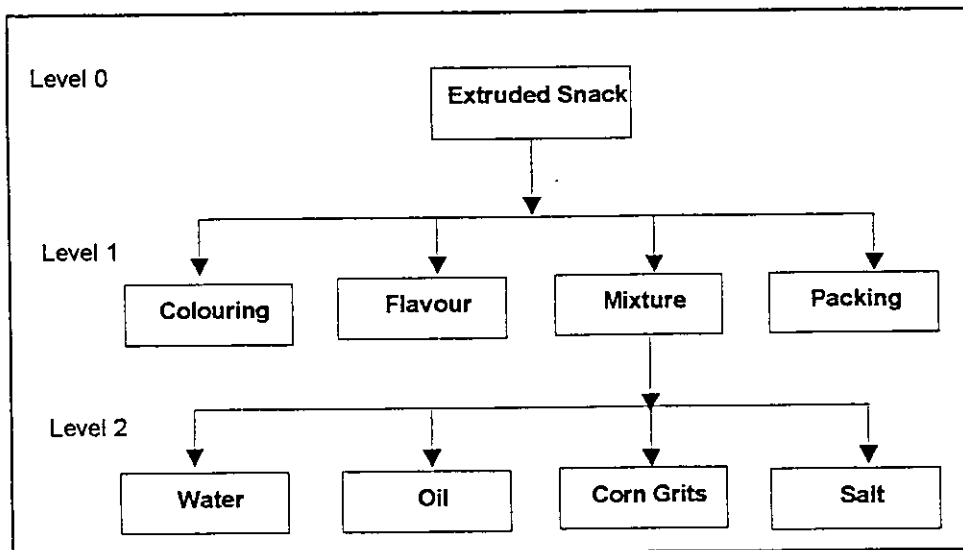


Figure 2.7 Bill of material case example.

## [2.2.2] PROCESSING LOGIC

In order to utilize the MRP in the proper way, the system procedures must analyze and answer the following items thoroughly:

- What do we have?
- What do we need?
- What do we do?

The first question can be analyzed and answered from the *Inventory Record File*. This file contains information about the status of each item. Any transaction such as stock receipt, disbursement, etc. are shown in this file.

As for the second question, it is necessary to investigate the Master Production Schedule, Externally Originated Orders and Forecasts For Independent Demand Items. These sources of demand sums up the total requirements of each item.

Now it is the time to explain the procedure required to satisfy demand. This procedure is called the *MRP Record Processing*. Following is an illustration of this procedure:

**Time Phasing:** Is the first step to be taken in order to control the system in an increments of periods. This is done through dividing the *Planning Horizon* into periods. These periods could be days, weeks, months, quarters or even years. It is the scope of this project to design the system with as short periods as possible like *days*, so as to obtain maximum response to expected changes, as will be seen later. The following table illustrates the idea of time phasing:



*Planning Horizon*

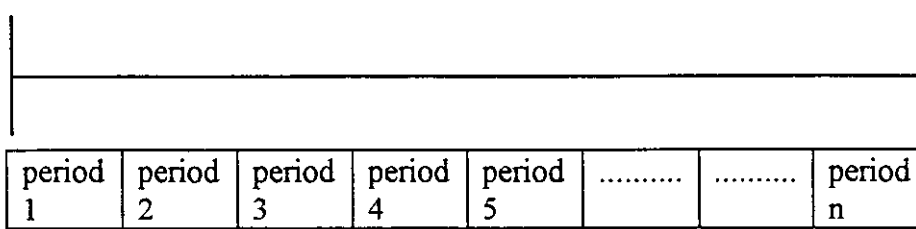


Figure 2.8 Planning Horizon.

**Quantity on hand:** Is the inventory status for the required item. This value is taken from the inventory record file and is either 0 or positive.

**Quantity on order:** Is the scheduled quantities to be placed in stock in the future. It is either under manufacturing or under delivery.

**Gross requirements:** Is the quantity of the item that will be disbursed, i.e., issued to support a parent order (or orders), rather than the total quantity that will be consumed by the end product. These two quantities may or may not be identical.

There may be multiple sources of demand, and therefore of gross requirements, for a given component item. An item demand may be subject to:

1. Dependent demand from several parent items that use it in common.
2. Independent demand from externally originated orders.
3. Forecasts of independent demand.

These gross requirements for the item are combined and summarized period by period in the following way:

<i>Periods</i>									
Item (A)	1	2	3	4	5	6	7	8	9
dependent demand + external orders + forecasts = <b>Gross Requirements</b>									

Figure 2.9 Gross requirements form.

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

<i>Periods</i>									
<b>Cheese E.</b>	1	2	3	4	5	6	7	8	9
dependent demand +	500	120	-	500	200	-	300	50	-
external orders +	5	5	5	5	5	5	5	5	5
forecasts =	200	220	200	500	200	-	300	50	-
<b>Gross Requirements</b>	<b>705</b>	<b>345</b>	<b>205</b>	<b>1005</b>	<b>705</b>	<b>5</b>	<b>605</b>	<b>105</b>	<b>5</b>

Figure 2.10 Gross requirements case example.

**Net Requirements:** The logic of the net requirements computation is simple:

Gross requirements  
*minus* Scheduled receipts  
*minus* On hand  
*equal* Net requirements

The net requirements are calculated on periodical base. At the beginning of the period, all sources of demand are added. Quantities of items scheduled to be received at the beginning of this period and the available inventory on hand at the beginning of the

period i.e., at the end of the previous period, are all subtracted from the summed demand.

From this formulas, it is understood that if the result is a negative value, i.e., if the sum of quantities on hand and on order exceeds the gross requirement, the net requirement is zero. While a positive result means that an action has to be taken in order to compensate this shortage.

**[2.2.3] COVERAGE OF NET REQUIREMENTS**

In the last section, it was explained that an action has to be taken only when the net requirements are negative. In MRP systems, this action should be the *Planned Orders*. Planned orders are new orders scheduled for release in the future. The convention followed in our system is as follows:

	<i>Periods</i>						
	1	2	3	4	5	.....	n
<i>Gross requirements</i>	G						
<i>Scheduled receipt</i>	R						
<i>On hand [H]</i>	H						
<i>Net requirements</i>	N						
<i>Planned-order release</i>	P						

Figure 2.11 Coverage of net requirements form.

$$N = G - R - H$$

If N = zero or negative, P = zero

else P should have a value equal or greater than N according to the following factors:

- Safety Stock.
- Offset Lead Time.
- Lot Sizing.

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

	<i>Periods</i>								
	1	2	3	4	5	6	7	8	9
<i>Gross requirements</i>	705	345	205	1005	705	5	605	605	605
<i>Scheduled receipt</i>	100	200	-	300	500	500	800	500	500
<i>On hand [1300]</i>	695	550	345	-360	-565	-70	125	20	-85
<i>Net requirements</i>	-	-	-	360	565	70	-	-	85
<i>Planned-order release</i>	-	-	-	360	565	70	-	-	85

Figure 2.12 Coverage of net requirements case example.

#### [2.2.4] SAFETY STOCK

Safety stock or reverse stock serves primarily to compensate for, or to absorb, fluctuation in demand. For purposes of the MRP calculation, the quantity of safety stock is either subtracted from the on-hand quantity or added to the gross requirement. Either alternative produces the same effect, namely increasing net requirements as follows:

$$N = G + S - R - H$$

or

$$N = G - R - (H - S) = G + S - R - H$$

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

Safety stock = 100

	<i>Periods</i>								
	1	2	3	4	5	6	7	8	9
<i>Gross requirements</i>	705	345	205	1005	705	5	605	605	605
<i>Scheduled receipt</i>	100	200	-	300	500	500	800	500	500
<i>On hand [1300 - 100]</i>	595	450	245	-460	-665	-170	25	-80	-185
<i>Net requirements</i>	-	-	-	460	665	170	-	80	185
<i>Planned-order release</i>	-	-	-	460	665	170	-	80	85

Figure 2.13 Safety stock of 100 kg. case example.

### [2.2.5] OFFSET LEAD TIME

Planned, or normal, lead times must be used by the MRP system for purposes of planning to determine order release dates. The lead time of a manufacturing item is made up of a number of elements, such as: Queuing time, processing time, setup time, transportation time, inspection time and others. Lead time is stated in periods and it affects MRP processing in the following way:

Lead time value ( in periods ) is subtracted from the order completion date, so the order release will be behind the order completion date by the value of the lead time.

Ex. If in period 5 a quantity Q is required and the lead time for this item is 2, so the order release should be in period 3.

	<i>Periods</i>						
	1	2	3	4	5	.....	n
<i>Gross requirements</i>	G						
<i>Scheduled receipt</i>	R						
<i>On hand [H]</i>	H						
<i>Net requirements</i>	N				Q		
<i>Planned-order release</i>	P		P				

Figure 2.14 Offset lead-time.

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

Lead-time = 2

Safety stock = 100

	<i>Periods</i>								
	1	2	3	4	5	6	7	8	9
<i>Gross requirements</i>	705	345	205	1005	705	5	605	605	605
<i>Scheduled receipt</i>	100	200	-	300	500	500	800	500	500
<i>On hand [1300 - 100]</i>	595	450	245	-460	-665	-170	25	-80	-185
<i>Net requirements</i>	-	-	-	460	665	170	-	80	185
<i>Planned-order release</i>	-	460	665	170	-	80	85	-	-

Figure 2.15 Lead-time of two periods case example.

## [2.2.6] LOT SIZING

In lot sizing, the ordering of inventory items must be quantized in standard quantities.

These quantities might be equal or greater than the net requirements, for reasons of economy or convenience.

There are many lot sizing techniques such as: fixed order quantity, economic order quantity (EOQ), lot for lot, Fixed period requirements and period order quantity. It is beyond the scope of this project to study these techniques and the lot size will be viewed as a controlled parameter of the system. As for record processing, it will be shown as follows:

**Lot size = Z**

	<i>Periods</i>						
	1	2	3	4	5	.....	n
<i>Gross requirements</i>	G						
<i>Scheduled receipt</i>	R						
<i>On hand [H]</i>	H						
<i>Net requirements</i>	N		<i>Q</i>				
<i>Planned-order release</i>	P		<i>P</i>				

Fig 2.16 Lot-size for a certain product.

If  $P \leq Z$ , then  $P = Z$

If  $P > Z$ , then  $P = \{\text{Ceiling}(P/Z) * Z\}$

( *Ceiling is the command which returns the nearest integer higher than or equal to Z* ).

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

**Lead-time = 2**

**Safety stock = 100**

**Lot-size = 100**

	<i>Periods</i>								
	1	2	3	4	5	6	7	8	9
<i>Gross requirements</i>	705	345	205	1005	705	5	605	605	605
<i>Scheduled receipt</i>	100	200	-	300	500	500	800	500	500
<i>On hand [1300 -100]</i>	595	450	245	-460	-665	-170	25	-80	-185
<i>Net requirements</i>	-	-	-	460	665	170	-	80	185
<i>Planned-order release</i>	-	500	700	100	-	1000	200	-	-

Figure 2.17 Lot size of 100 kg. case example.

### [2.2.7] SYSTEM OUTPUTS

The primary outputs of an MRP system are the following:

- Order-Release notices, calling for the placement of planned orders.
- Rescheduling notices, calling for changes in open orders due dates.
- Cancellation notices, calling for cancellation or suspension of open orders.
- Item status analysis backup data.
- Planned orders scheduled for release in the future.



## [2.3] DATA MODELING

In the previous sections, the MRP system was clearly defined and described. From this description, we can now move to the next step, which is data modeling.

A data model is an abstract representation (a description) of data concerning entities, events, activities and their association within an organization [17]. More liberally a data model represents (describes) an organization itself. The purpose of data modeling is two folds: First, to represent data and second, to make the data understandable. Thus the data model can develop a global design for the database with the ultimate objective of achieving an efficient implementation which satisfies the requirements.

Different approaches to data modeling were described in the introduction and the choice was made on the object-oriented data model, because it was found best in satisfying our needs.

The object-oriented data model is a combination of three models:

1. Object model.
2. Dynamic model.
3. Functional model.

## [2.4] OBJECT MODEL

An object model captures the static structure of a system by showing the objects in the system, relationships among the objects, and the attributes and operations that characterize each class of objects. In order to build the model, we will follow the Object Modeling Technique (OMT) notation and methodology throughout our work.

### [2.4.1] OBJECTS

An Object is simply something that makes sense in an application context. For the OMT, an object is defined as a concept abstraction, or thing with crisp boundaries and meaning for the problem at hand. All objects have identity and are distinguishable. Identity means that objects are distinguished by their inherent existence and not by descriptive properties that they may have.

Regarding the MRP system, the following objects are of concern:

Parts	Subassemblies	Products
Processes	Machines	Time periods
Customers	Forecasting	External demand

Figure 2.18 The MRP objects.

### [2.4.2] CLASSES

An object class describes a group of objects with similar properties (attributes), common behavior (operations), common relationships to other objects, and common semantics.

As for the MRP objects defined above, it is possible to derive the following classes:

Part class	Subassembly class	Product class
Process class	Machine class	Time period class
Customer class	Forecasting class	External demand class

Figure 2.19 The MRP classes.

### [2.4.3] OBJECT DIAGRAMS

Object diagrams provide a graphic format notation for modeling objects, classes and their relationships. There are two types of object diagrams used in our analysis: a class diagram and an instance diagram.

### [2.4.3.1] CLASS DIAGRAM

which is a schema, pattern, or template for describing many possible instances of data. A class diagram describes object classes. The OMT symbol for a class is box with class name in **boldface**. For the MRP system, the class diagram shown below.

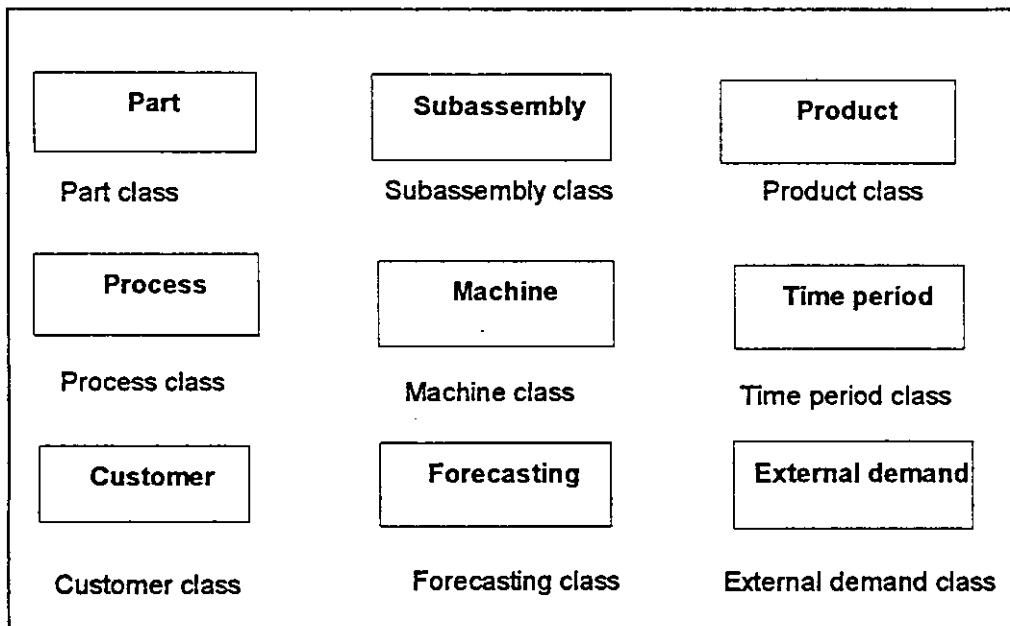


Figure 2.20 A class diagram for the MRP classes.

### [2.4.3.2] AN INSTANCE DIAGRAM

It describes how a particular set objects relate to each other. An instance diagram describes object instances. A given class diagram corresponds to an infinite set of instance diagrams. The OMT symbol for an instance diagram is a round box. The class name in parentheses is at the top of the object box in boldface. The object name is listed in a regular typeface.

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

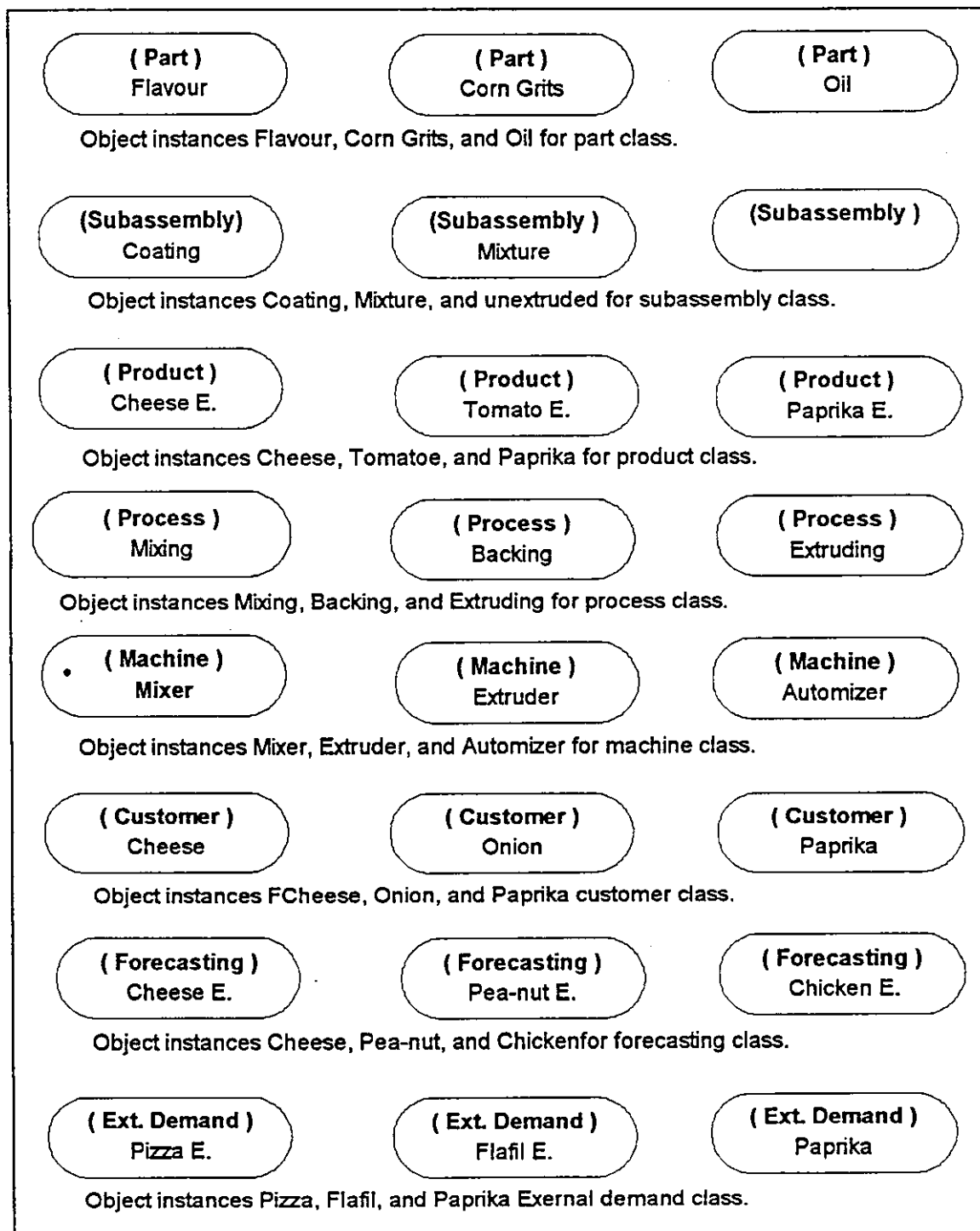


Figure 2.21 An MRP instance diagram case example.

#### **[2.4.4] ATTRIBUTES**

An attribute is a data value held by the objects in a class. Each attribute has a value for each object instance. Attributes are listed in the second part of the class box. Each attribute name may be followed by optional details such as type and default value. The type is preceded by a colon, while the default value is preceded by an equal sign. Class boxes have a line drawn between the class name and its attributes. Object boxes do not have this line to further differentiate them from class boxes.

From our study of the MRP system, the most important attributes were collected and represented in Figure 2.22.

#### **[2.4.5] OPERATIONS AND METHODS**

An operation is a function or transformation that may be applied to or by objects in a class. Each operation has a target object as an implicit argument. The behavior of the operation depends on the class of its target. A method is an implementation of the an operation for a class. Operations are listed in the lower third of the class box. Each operation name may be followed by optional details, such as argument list and result type. An argument list is written in parentheses following the name. The result type is preceded by a colon.

For the above defined classes in the MRP system, relative operations appended to the class diagram are shown in Figure 2.23.

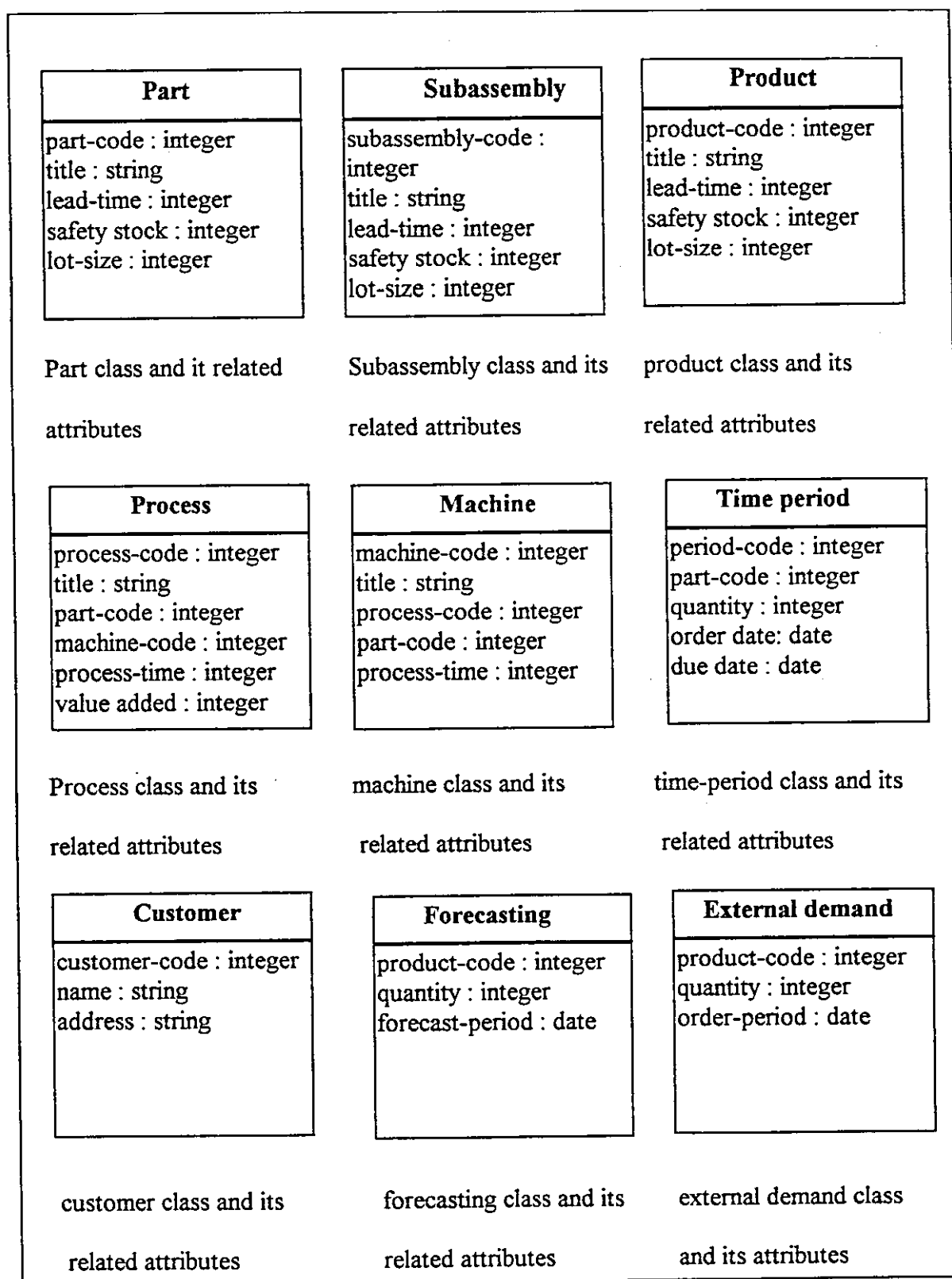


Figure 2.22 Representation of classes with their attributes.

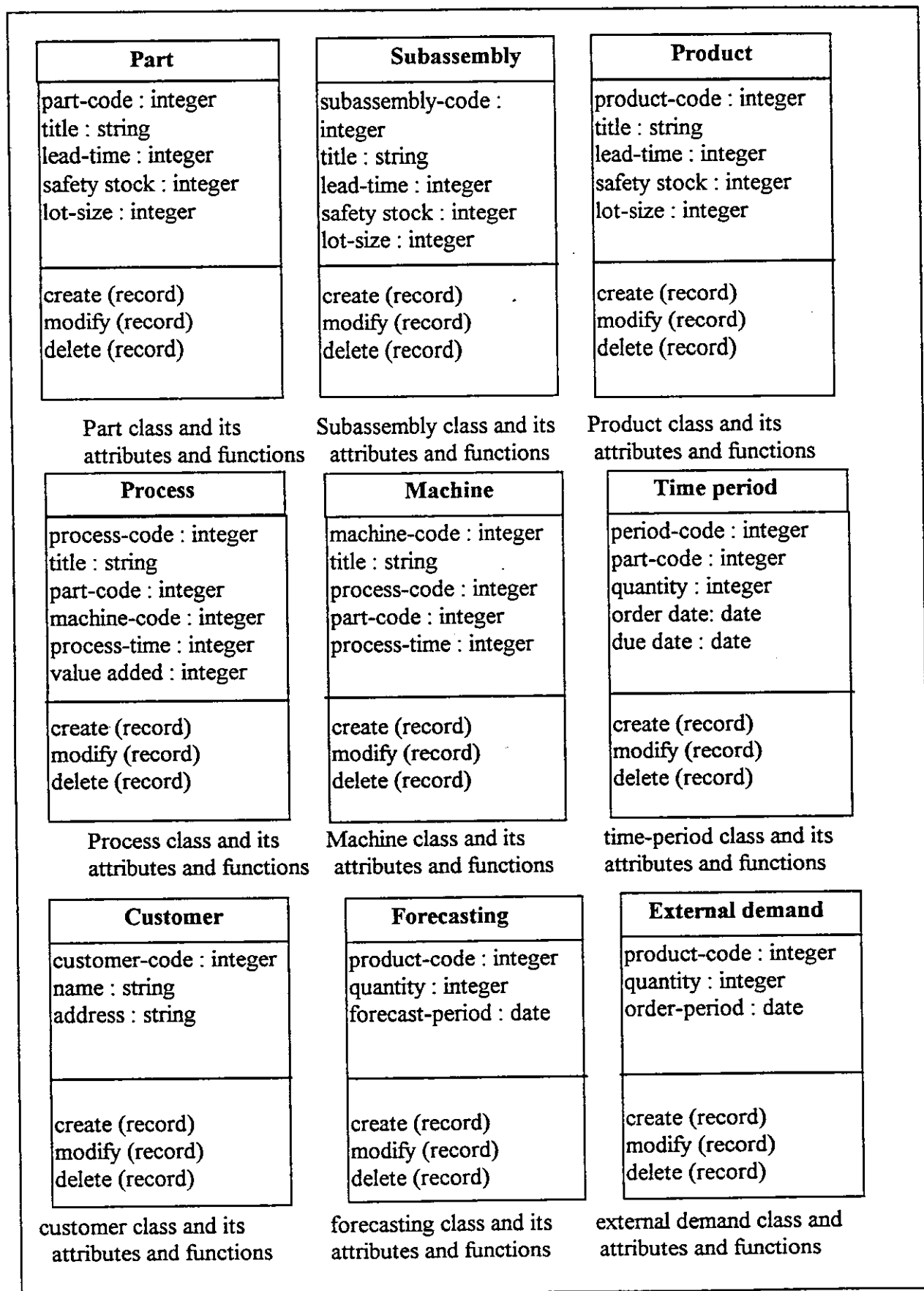


Figure 2.23 The MRP classes with their operations.



<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

<b>Flavor</b>	<b>Mixture</b>	<b>Cheese E.</b>
part-code : 5100 title : Flavor lead-time : 2 periods safety stock : 100 kg. lot-size : 100 kg.	subassembly-code : 6100 title : Mixture lead-time : 3 periods safety stock : 50 kg. lot-size : 50 kg.	product-code : 7100 title : Cheese E. lead-time : 4 periods safety stock : 100 kg. lot-size : 100 kg.
create (record) modify (record) delete (record)	create (record) modify (record) delete (record)	create (record) modify (record) delete (record)
Part class and its attributes and functions	Subassembly class and its attributes and functions	Product class and its attributes and functions
<b>Backing</b>	<b>Mixer</b>	<b>Period 1</b>
process-code : 101 title : baking part-code : 5200 machine-code : 3100 process-time : 10 min. value added : 30%.	machine-code : 3200 title : Mixer process-code : 102 part-code : 5300 process-time : 30 min.	period-code : 1 product-code : 7100 quantity : 100 kg. order date: 05/08/95 due date : 10/08/95
create (record) modify (record) delete (record)	create (record) modify (record) delete (record)	create (record) modify (record) delete (record)
Process class and its attributes and functions	Machine class and its attributes and functions	time-period class and its attributes and functions
<b>Ahmad Ali</b>	<b>Paprika E.</b>	<b>Onion E.</b>
customer-code : 33 name : Ahmad Ali address : Amman	product-code : 7100 quantity : 100 kg. forecast-period : 10/08/95	product-code : 7500 quantity : 200 kg. order-period : 12/08/95
create (record) modify (record) delete (record)	create (record) modify (record) delete (record)	create (record) modify (record) delete (record)
customer class and its attributes and functions	forecasting class and its attributes and functions	external demand class and attributes and functions

Figure 2.24 MRP classes with their operations case example.

## [2.4.6] DATA DICTIONARY

The data dictionary is a precise description of all the classes. The scope of the class within the MRP problem is described including any assumptions or restrictions on its membership use. The data dictionary describes associations, attributes, and operations.

Figure 2.25 shows the complete data dictionary of the MRP system.

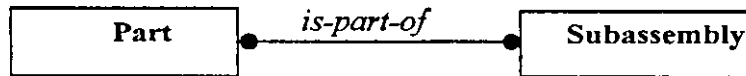
## [2.4.7] LINKS AND ASSOCIATIONS

A link is a physical or conceptual connection between object instances. Mathematically, a link is defined as a tuple, that is, an ordered list of object instances. A link is an instance of association. An association describes a group of links with common structure and common semantics. All the links in an association connect objects from the same class. An association describes a set of potential links in the same way that a class describes a set of potential objects. The OMT notation for an association is a line between classes. A link is drawn as a line between objects. Association names are italicized. It is good to arrange the classes to read from left-to-right, if possible. If the association is one, then a straight line is just drawn. If the association is many, then a line with solid balls at its ends is drawn. It is important to note that the term association here, is synonymous with the term relation used in RDBMS. Typical associations among MRP classes are shown in Figure 2.26.

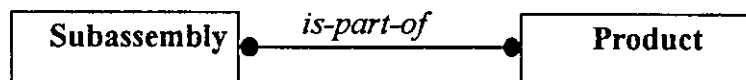
- **Part** : This is the raw material used in different processes to produce different subassemblies and products. It is not a result of any manufacturing process. Every part has a value, lead-time, safety stock, and a lot size. This value is necessary to compute the ABC-classification of inventory. The lead-time is necessary to phase the requirements in the different time periods. The safety stocks and the lot size are planning factors required to minimize the total cost. The quantity of each part should be known. Each part could participate in building different products and the same products can use the different parts.
- **Subassembly** : This is the semi-finished product or, as it is called, the work-in-process (W.I.P.) found in the intermediate stages of the factory. It results from some manufacturing processes done on the parts (raw material), but it cannot be sold directly to customers. For every subassembly there is a lead time, safety stock, and a lot size. The subassembly could have one or many parts and it could be used to produce one or many products.
- **Product** : This is the finished product ready for shipment to the customers. No further operations is needed and it can be sold directly to customers. Customer orders are expressed in the terms of products. For this product, lead-time, safety stock, and lot size should be kept in its records. A single order may have single or multiple products and a single product may be ordered by single or multiple orders.

- **Process** : This is the manufacturing process which is done on parts or subassemblies to convert them to assemblies or finished products. Each process adds a value to the part being processed. The process has a time which is added to the lead time. Each process may have one or many parts/subassemblies and each part/subassembly may require one or several processes.
- **Machine** : This is the tool required to perform the manufacturing processes on parts or subassemblies.
- **Time-period** : This is the time in which different products are ordered
- **Customer** : This is the source of demand for end products. The customer may place an order for a single product or for multiple products.
- **Forecasting** : This is the source of demand generated from the sales department. This plan determines the expected demand for different products in certain time-periods.
- **External demand** : This includes service-part orders, interplant orders, original equipment manufacturer (OEM) orders by other manufacturers who use these components in their products, components may be ordered for purposes of experimentation, destructive testing, promotion, equipment maintenance or other special-purpose orders not related to the regular production plan.

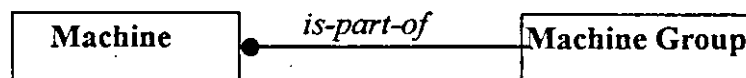
Figure 2.25 Data dictionary for the MRP system.



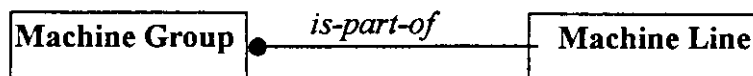
Component part *is-part-of* the subassembly. Different parts can be used to construct one subassembly and different subassemblies can use the same part, so the association between part and subassembly is *many-to-many*.



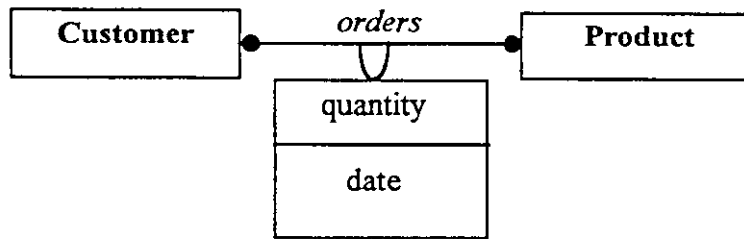
Subassembly *is-part-of* the product. Different subassemblies can be used to construct one product and different products can use the same subassembly, so the association between subassembly and product is *many-to-many*.



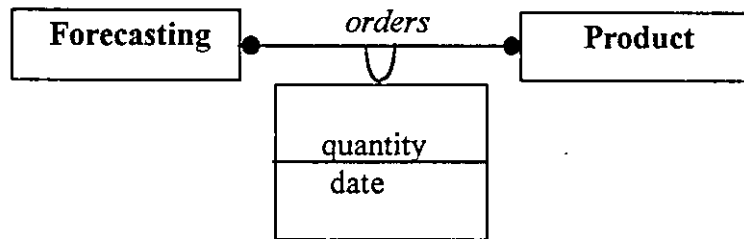
Machine *is-part-of* the machine group. Different machines can be used to form one machine group, but no more than one machine group can utilize the same machine, so the association between machine group and machine is *one-to-many*.



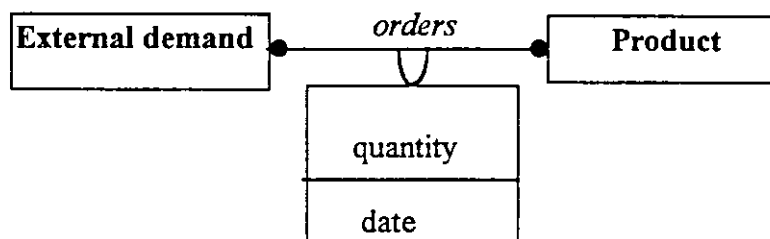
Machine group *is-part-of* the machine line. Different machine groups can be used to form one machine line, but no more than one machine line can utilize the same machine group, so the association between machine line and machine group is *one-to-many*.



Customer makes an order for one or different products and the product can be ordered from one or different customers, so the relation between the customer and the product is *many-to-many*



Forecasting puts an order for one or different products and the product can be ordered from only one forecasting module, so the relation between the forecasting and the product is *one-to-many*



External demand makes an order for one or different products and the product can be ordered from one or different external demand sources, so the relation between the external demand and the product is *many-to-many*

Figure 2.26 Linkages among MRP classes.

### [2.4.7.1] MODELING ASSOCIATION AS A CLASS

It is useful to model an association as a class when links can participate in associations with other objects or when links are subject to operations. For the association *order* found between each of customer, forecasting, and external demand and product, it is necessary to add attributes to this association and to model it as a class. The order class, then should be represented as follows:

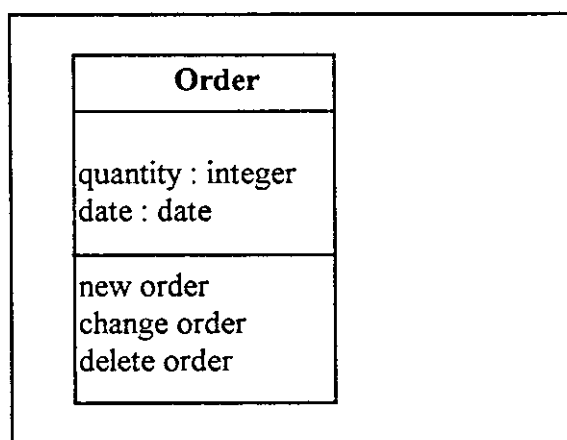


Figure 2.27 Order class with its related attributes and functions

## [2.4.8] OBJECT-ORIENTED ABSTRACTIONS

Abstraction is defined as the ability to focus on the most important considerations in the problem. In all approaches to hierarchical planning, abstraction is the crucial step in the design of the model. *The purpose of abstraction in programming is to separate behavior from implementation.* This makes the large software projects more intellectually manageable since it allows for separate and independent development of different parts of the system.

For the scope of our work, five abstraction concepts will be implemented in the design of our object-oriented data model. These concepts are:

- Identification.
- Aggregation.
- Generalization/specialization.
- Selection.

### [2.4.8.1] IDENTIFICATION

Each object class and object instance in an object-oriented database should have a unique identity that is external to the data values stored within the object. Thus, there is no concept of primary key used as a unique identifier, as in other database models. Instead, the object-oriented database management system (OODBMS) maintains an external identifier for each object that is not accessible to and cannot be modified by any other object or application. As a result, the identity of an object does not change, spite the fact that the values of any of its attributes may change.



Regarding the MRP system, the following identifications are given for the classes and their objects:

<i>Class Identification</i>	<i>Object Identification</i>
Part	Part-code
Subassembly	Subassembly-code
Product	Product-code
Process	Process-code
Machine	Machine-code
Period	Period-number
Customer	Customer-code
Forecasting	Forecasting-code
External demand	External demand-code

Figure 2.28 Identification for the MRP classes.

#### [2.4.8.2] AGGREGATION

In the object-oriented model, aggregation is a powerful concept in that, it permits the combination of objects that are related via some particular relationship into a higher level which is the aggregate object. Usually a meaningful name is assigned to this aggregate type and one may use this name without reference to the underlying properties of the type. This concept is particularly useful if the aggregate object has additional properties or is itself to be related to another object.

For MRP, the following aggregations can be found:

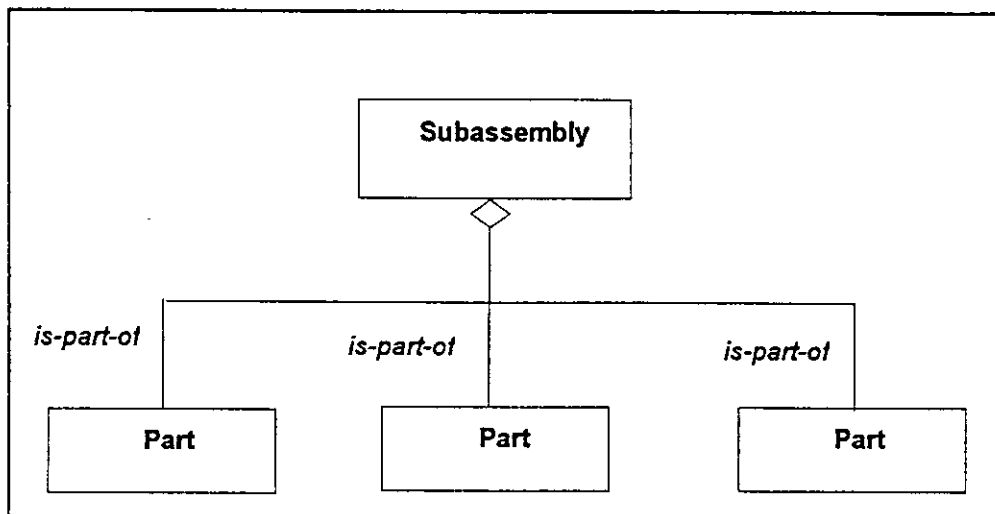


Figure 2.29 Aggregation of parts into subassembly.

Raw material parts can be found in huge numbers and diversity, but only certain parts can be used to produce subassemblies. The parts which constitute the subassembly are aggregated by that subassembly.

**WHITEROSE** CASE EXAMPLE

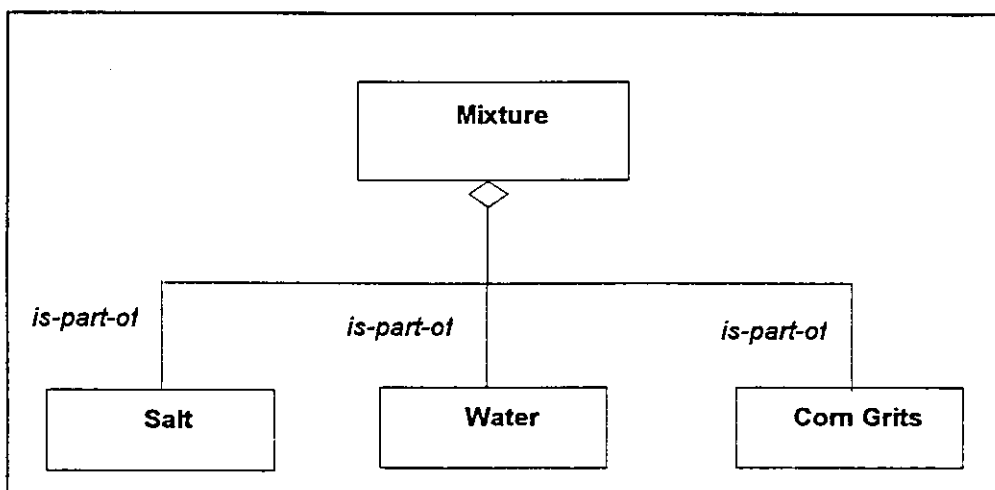


Figure 2.30 Aggregation of salt, water, and corn grits into mixture case example.

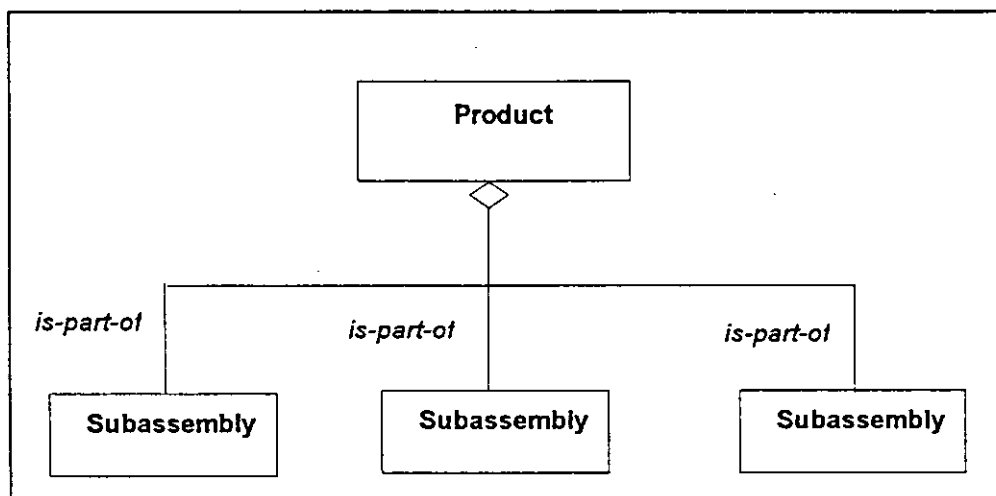


Figure 2.31 Aggregation of subassemblies into product.

Subassemblies which take part to form the product are aggregated by that product.

**WHITEROSE** CASE EXAMPLE

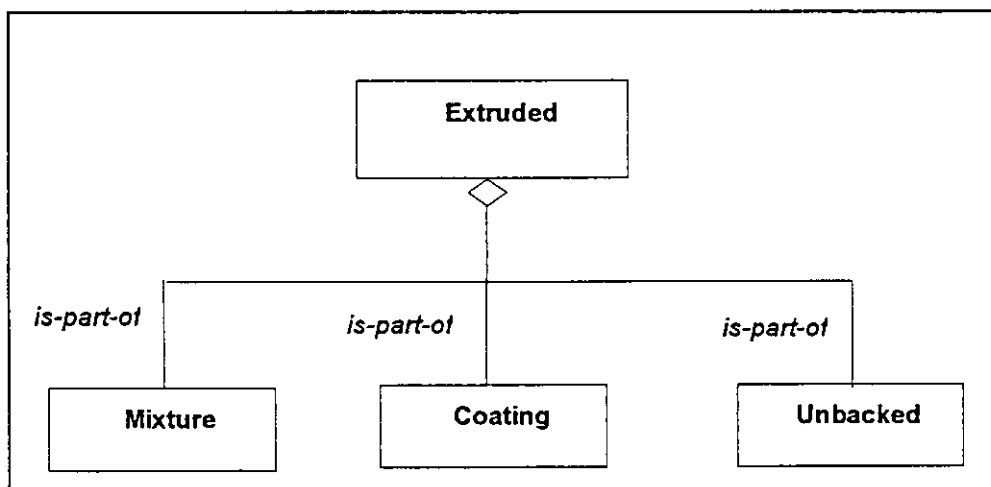


Figure 2.32 Aggregation of mixture, coating, and unbaked into extruded case example.

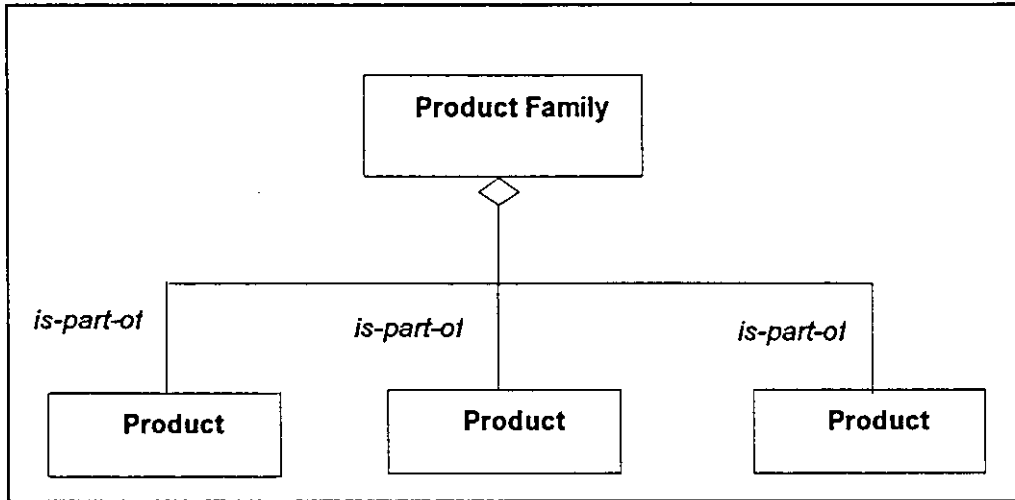


Figure 2.33 Aggregation of products into product family.

Products that belong to the same product family are aggregated by that family

**WHITEROSE** CASE EXAMPLE

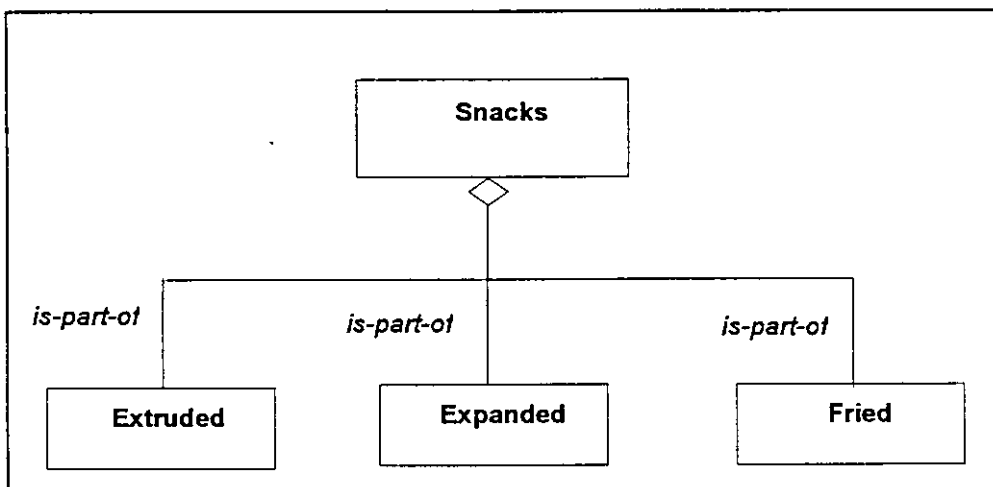


Figure 2.34 Aggregation of extruded, expanded, and fried product into snacks family.

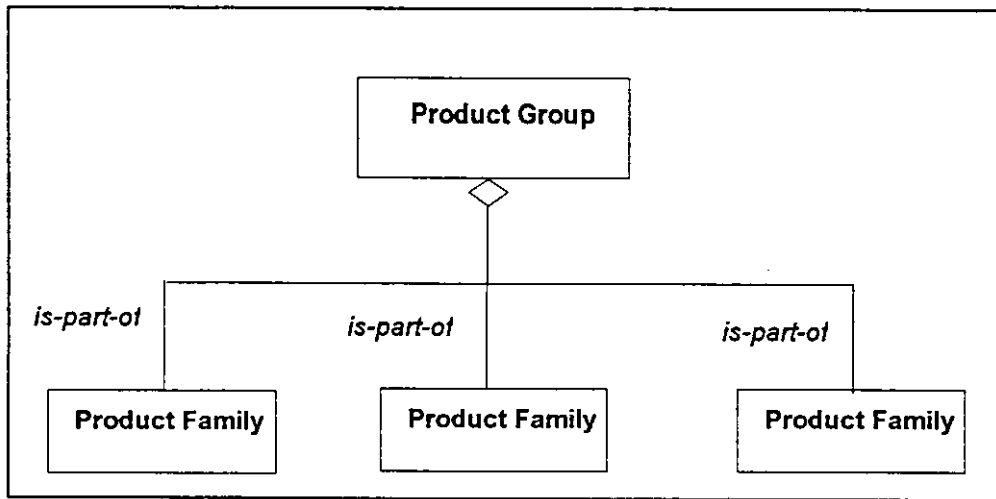


Figure 2.35 Aggregation of product families into product group.

Product families which take part to form product group can be aggregated by the product group.

**WHITEROSE** CASE EXAMPLE

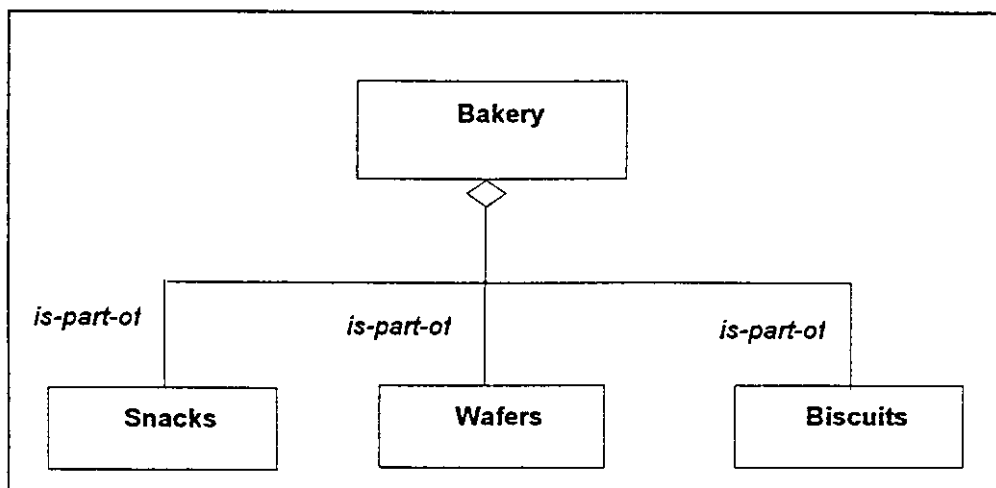


Figure 2.36 Aggregation of snacks, wafers, and biscuits into bakery product group.

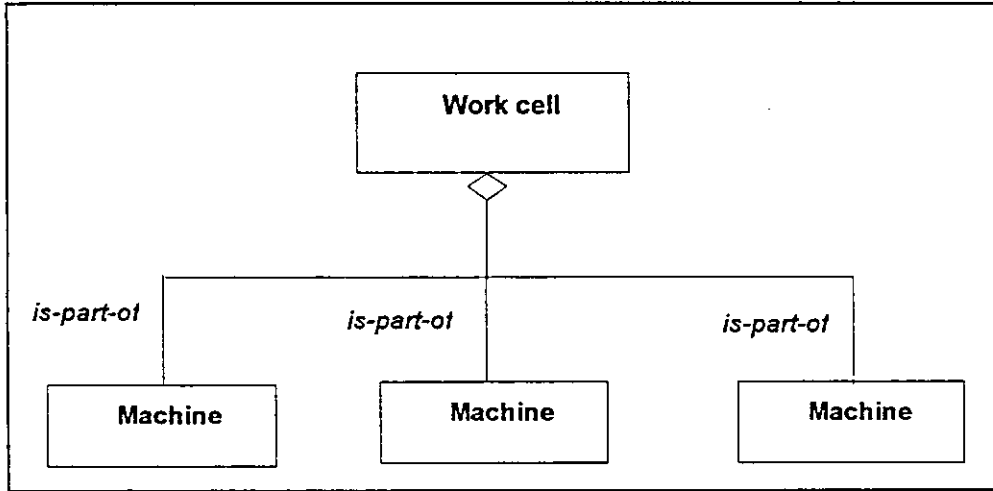


Figure 2.37 Aggregation of machines into work cell.

Machines which take part in a work cell are grouped by that work cell.

**WHITEROSE** CASE EXAMPLE

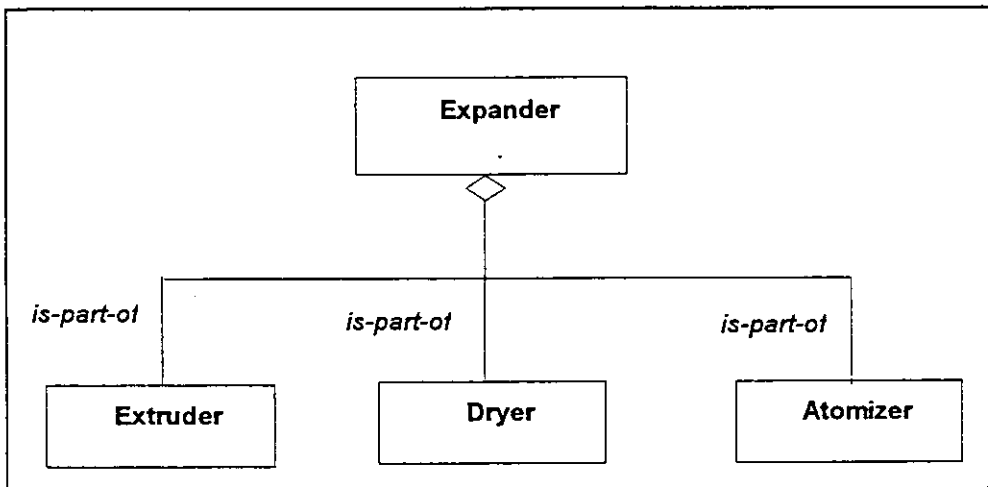


Figure 2.38 Aggregation of the Extruder, Dryer, and Atomizer into the Expander work cell.

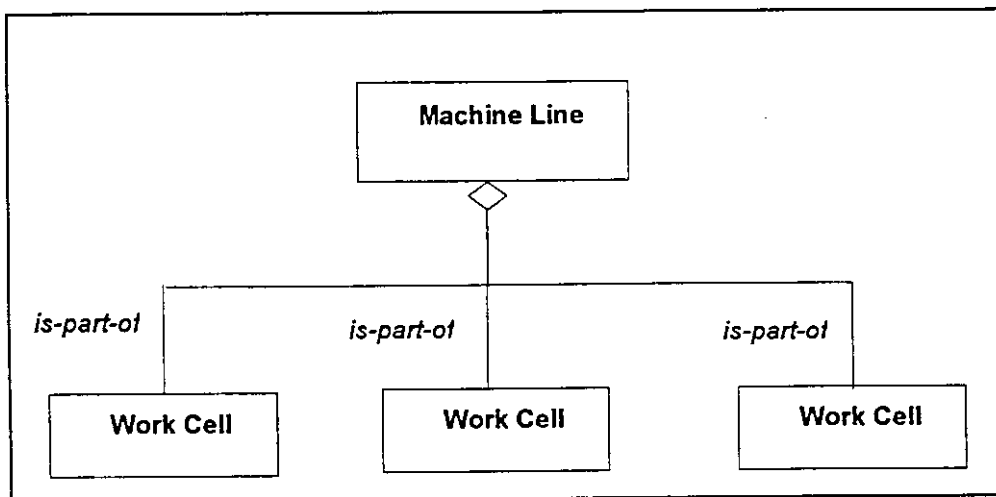


Figure 2.39 Aggregation of work cells into machine line.

Work cells which take part in machine lines are grouped by machine line.

**WHITEROSE** CASE EXAMPLE

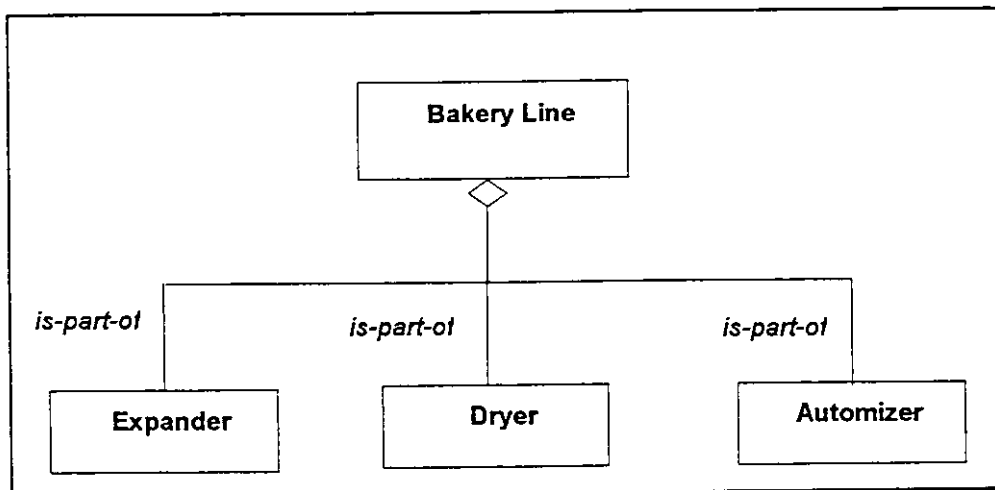


Figure 2.40 Aggregation of the Expander, Dryer, and Atomizer into the Bakery line.

### [2.4.8.3] GENERALIZATION/SPECIALIZATION

Generalization is an abstraction in which the common properties and functions of a group of similar object types are grouped together to form a new generic object type.

Whereas, specialization permits subtypes of a given object type to be defined using predicates to constraint the values of attributes.

Regarding the MRP system, Figure 2.41 shows the possible generalization.

### [2.4.8.4] SELECTION

Selection is the dimension of abstraction that is based on different criticalities assigned to different objects. Based on any set of elementary objects, only these objects whose value of one or more attributes exceeds certain threshold values are selected. The selected objects are treated as *an abstract set of objects*.

It is a design problem to select the criticality parameters of the subset and this task needs lot of experience and knowledge of the system. It is also, a situation dependent process which needs to be customized according to the problem.



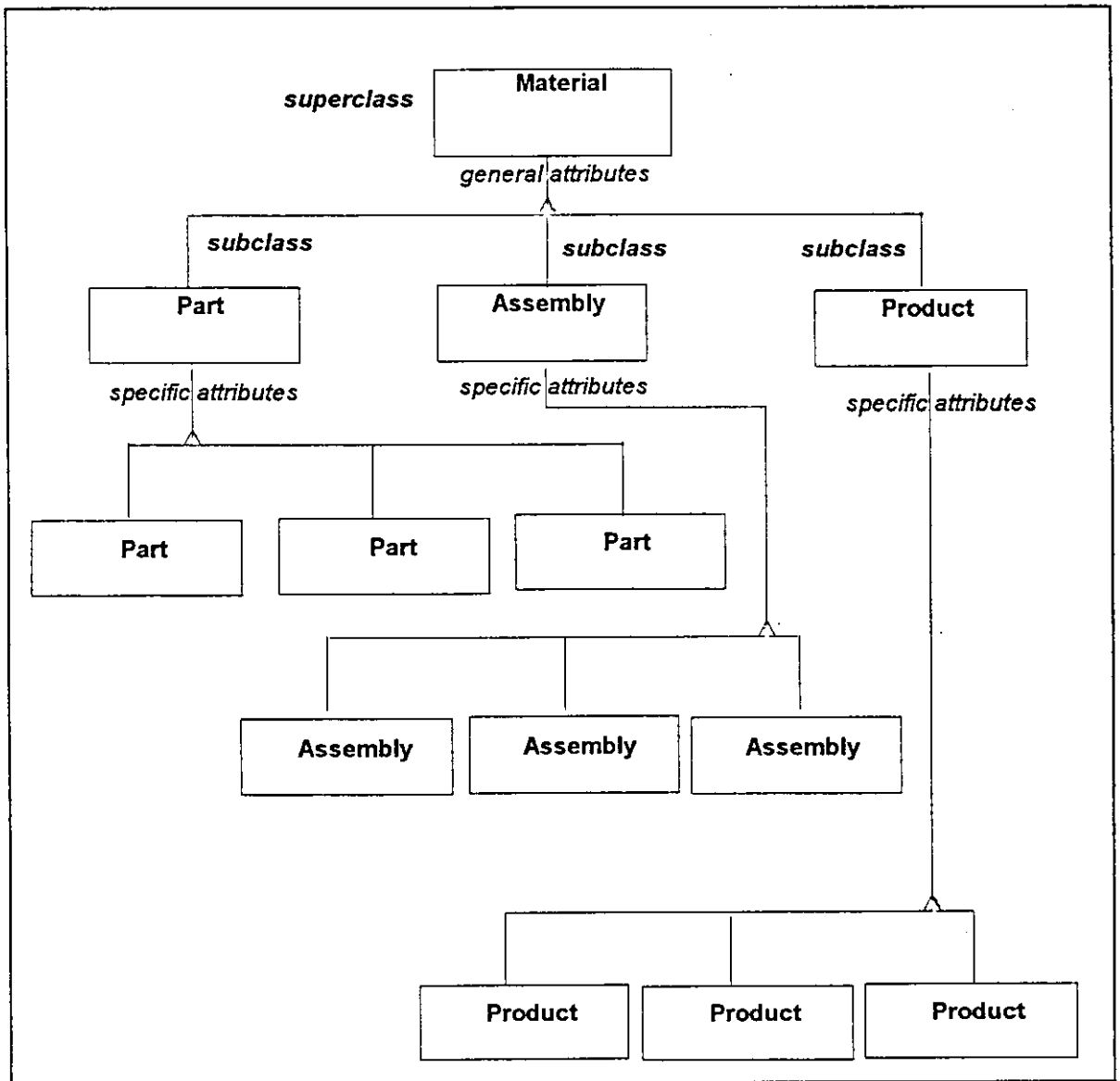


Figure 2.41 Generalization / Specialization of MRP objects

<b>WHITEROSE</b>	<b>CASE EXAMPLE</b>
------------------	---------------------

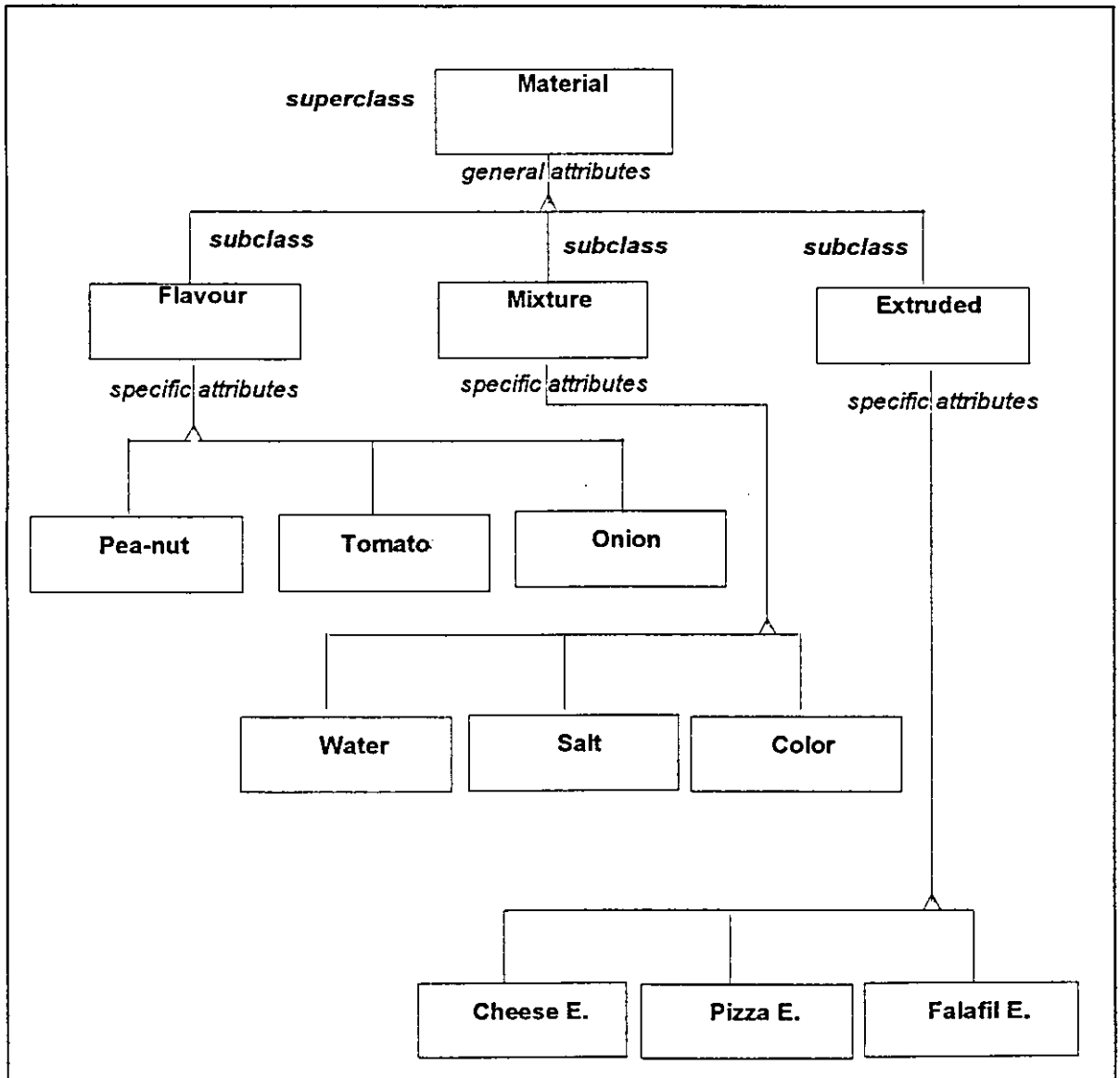


Figure 2.42 Generalization /Specialization of MRP objects case example.

Followings are some of the selection issues related to MRP system:

- Parts lead time: The lead time for all parts are known, and only parts with lead time exceeding a certain value are selected to be critical parts.

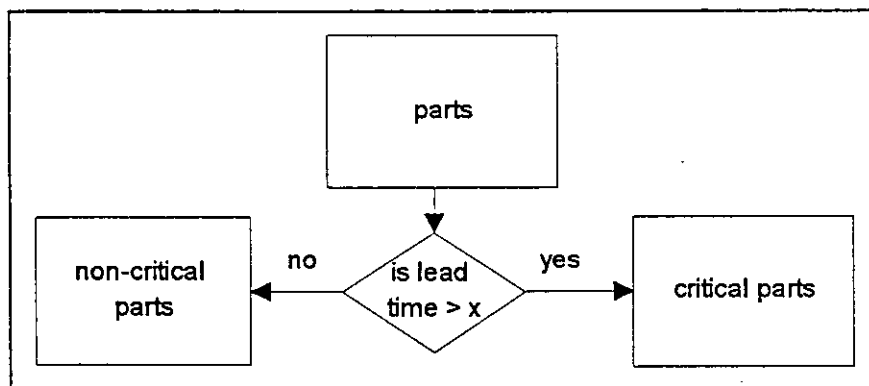


Figure 2.43 Selection based on critical lead-time.

### WHITEROSE CASE EXAMPLE

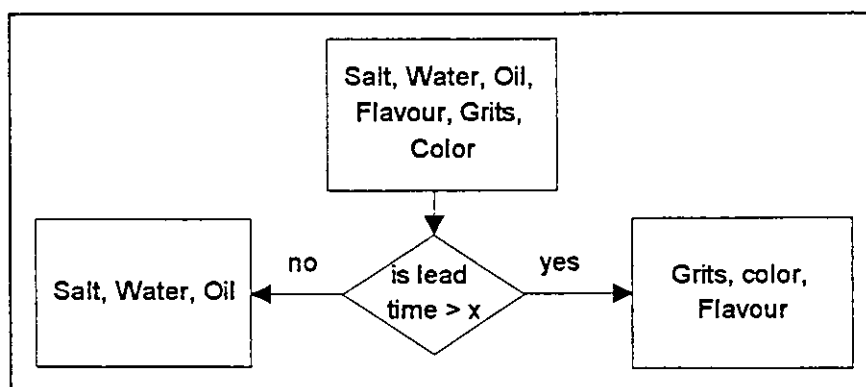


Figure 2.44 Selection based on critical lead-time case example.

- Part value: The value of each part is known, and only parts with values exceeding a certain value are selected to be critical parts.

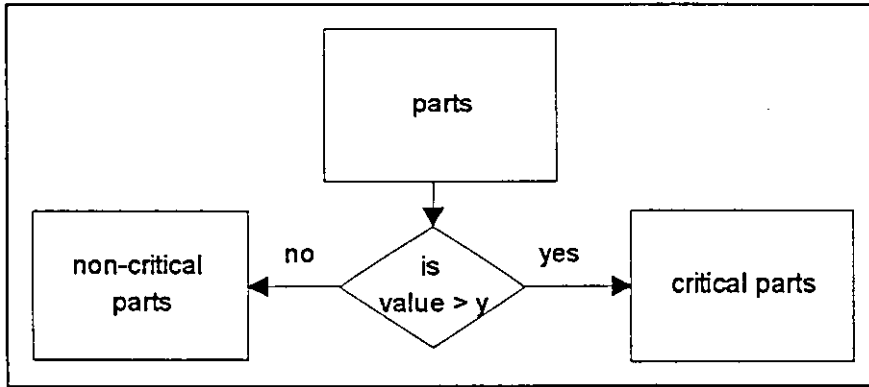


Figure 2.45 Selection based on critical value.

### WHITEROSE CASE EXAMPLE

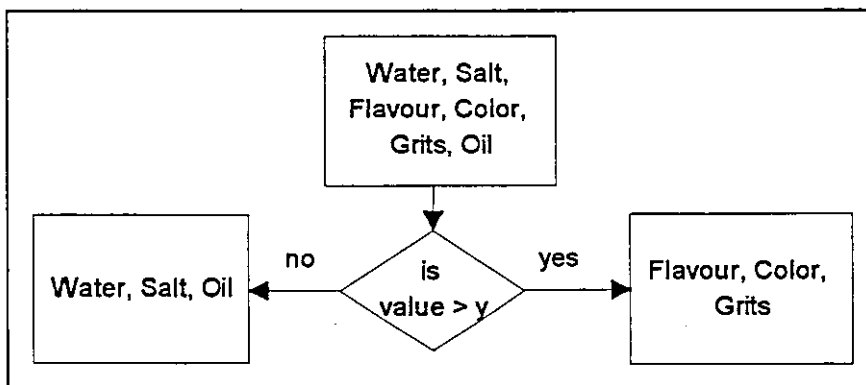
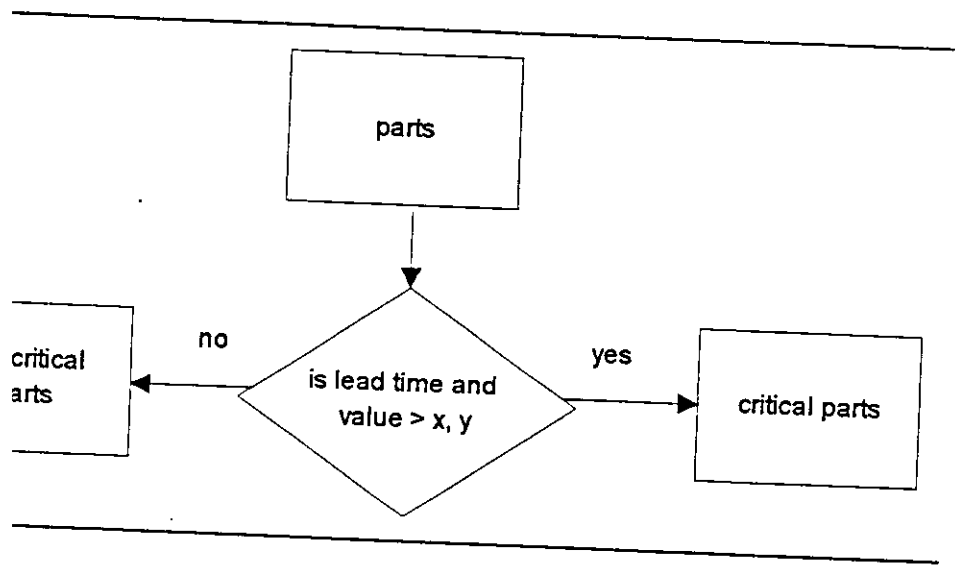


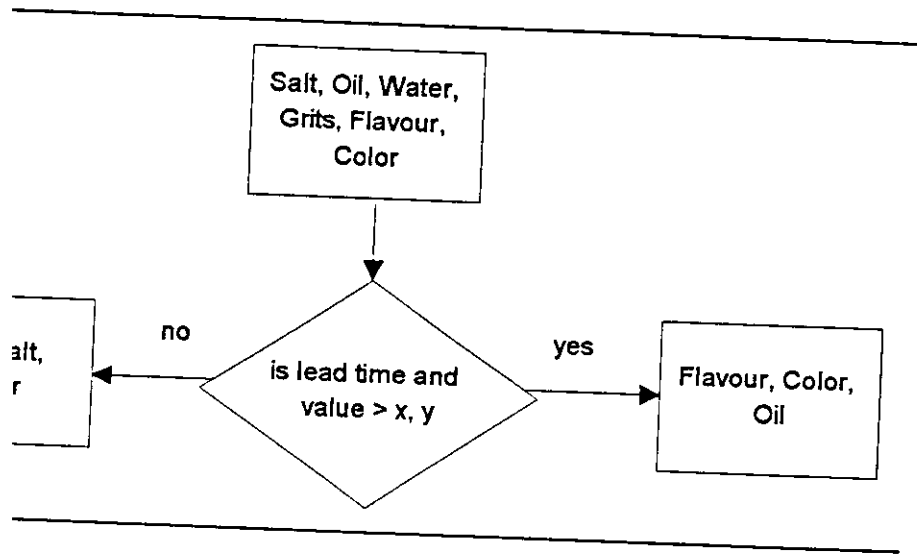
Figure 2.46 Selection based on parts value case example.

lead time and value: parts with both lead time and value exceeding a certain value selected to be critical.



Selection based on both parts value and lead-time.

**HITEROSE CASE EXAMPLE**



Selection based on both critical lead-time and part value case example.

the  
of  
ontrol,  
without  
mented.  
Diagram  
acts. State  
nd to  
tions in

ed:  
rio.  
ior.  
e diagrams.

## [2.5.1] EVENTS

An event is something that happens at a point in time, but has no duration; i.e. an event is an occurrence that is fast compared to the granularity of the time scale of a given abstractions. One event may logically proceed or follow another, or the two may be unrelated. An event is a one-way transmission of information from one object to another. An object sending an event to another object may expect a reply, but the reply is a separate event under the control of the second object, which may or may not choose to send it.

Every event is a unique occurrence, but we group them into event classes and give each event class a name to indicate common structure and behavior. Some events are simple signals, but most event classes have attributes indicating the information they convey. The time at which the event occurs is an implicit attribute of all events. Attributes are shown in parentheses after the event class name.

For the MRP system, the following event classes may occur:

- Customer makes an order for certain products.
- Forecasting places an order for certain products.
- External demand makes an order for certain products.
- Customer changes order quantity for certain products.
- Forecasting changes order quantity for certain products.
- External demand changes order quantity for certain products.
- Customer changes order due-date for certain products.
- Forecasting changes order due-date for certain products.

- External demand changes order due-date for certain products
- Scheduled receipt quantity is changed.
- Scheduled receipt date for a certain item is changed.
- Projected available balance is changed.
- Product lead time is changed.
- Product lot size is changed.
- Product safety-stock quantity is changed.
- Product structure is changed.

### **[2.5.2] SCENARIOS AND EVENT TRACES**

A scenario is a sequence of events that occur during one particular execution of a system. The scope of a scenario varies; it may include all events in the system, or it may include only those events impinging on or generated by certain objects in the system.

The following diagram shows a certain scenario in the MRP system, which is the for the order class:

- Customer makes an order for certain products.
  - Orders are scheduled in different time periods.
  - Orders are exploded using the B.O.M. structures.
  - Requirements are generated for orders using MRP record processing.
  - Orders are released.

Figure 2.49 Normal order scenario.

This scenario is the normal order processing cycle, but suppose that the customer determines to change, for example, the ordered quantity, or the ordered product-code, or the ordered due-date. The way to handle these changes is to delete the old order and to reenter his order as a new one with the same scenario as the above one.

- Customer changes the ordered quantity, or the ordered product-code, or the ordered due-date.
- Old order is deleted from orders.
- New customer order with the new parameters is entered.
- Orders are scheduled in different time-periods.
- Orders are exploded using the B.O.M. structures.
- Requirements are generated for orders using MRP record processing.
- Orders are released.

Figure 2.50 Changes of order parameters scenario.



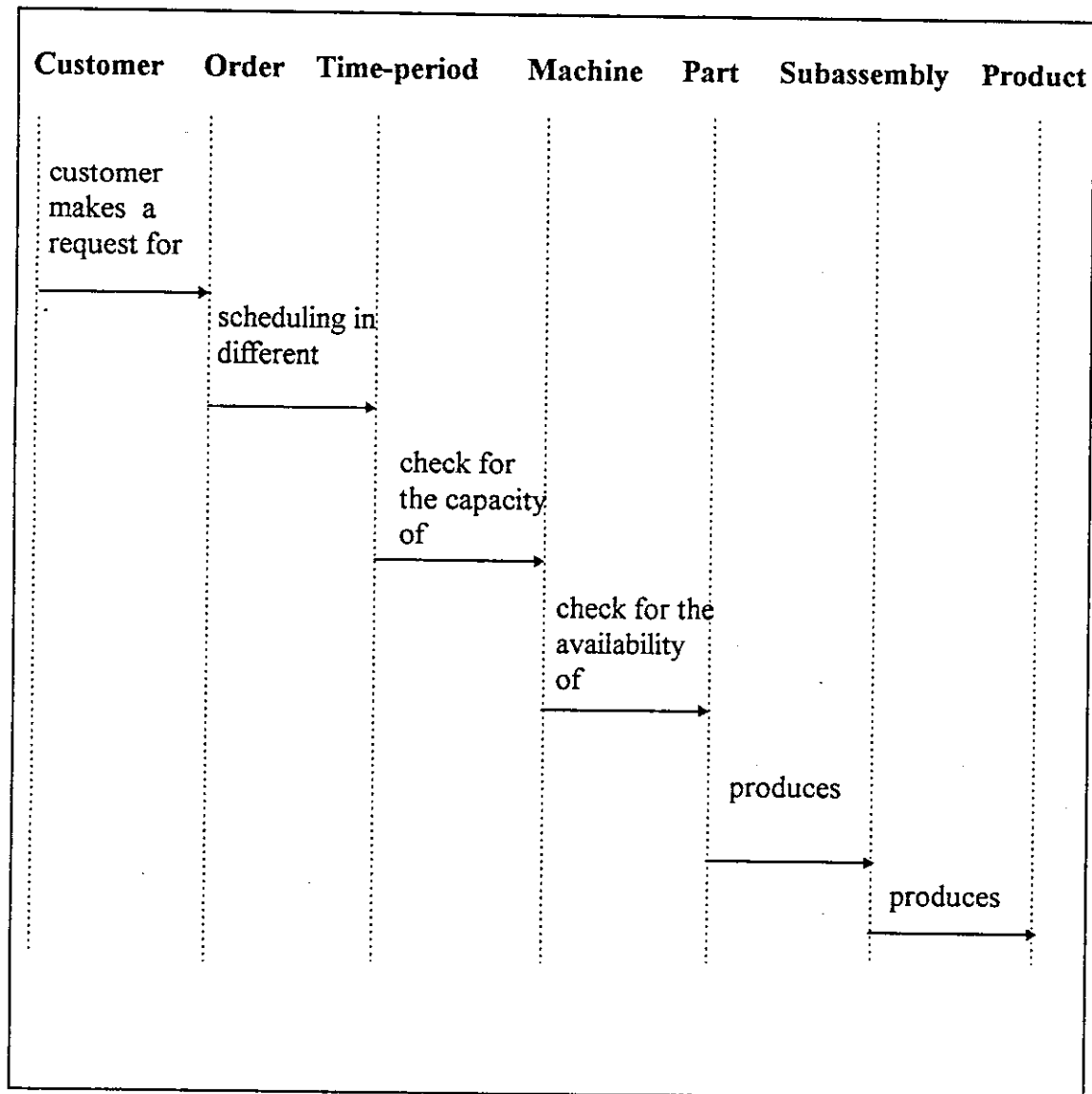


Figure 2.51 Event trace for an order.

### [2.5.3] STATES

A state is an abstraction of the attribute values and links of an object. Sets of values are grouped together into a state according to properties that affect the gross behavior of the object. A state specifies the response of the object to input events. The response to an event received by an object may vary quantitatively depending on the exact values of its attributes, but the response is qualitatively the same for all values within the same state, and may be qualitatively different for values in different states. The response of an object to an event may include an action or a change in state by the object. A state corresponds to the interval between two events received by an object. Events represents points in time; states represents intervals of time. Events and states are dual of one another; an event separates two states, and a state separates two events.

A state is often associated with the value of an object satisfying some conditions. In the simplest case, each enumerated value of an attribute defines a separate state. The OMT represents the state by a suggestive name and a natural-language description of its purpose. Then the event sequence that leads to a state and a declarative condition for the state given in terms of parameters are stated. Finally, a stimulus-response table shows the effect of events, the action that occur, and the next state.

Following we show the characterizing of an order release state.

**State:** Order release

**Description :** An order for specific products is issued in order to satisfy the net requirements.

Event sequences that produce the state:

- Customers orders products or the customer changes the order quantity or the customer changes the order product code or the customer changes the order due-date.
- A schedule plan is issued for different products in different time-periods after checking the available capacity.
- The MRP system checks the quantity required against the available stock and the expected to arrive stock.

Conditions that characterizes the states:

Gross requirements > [available stock + scheduled receipts].

Required capacity < available capacity

Events accepted in the state:

event	action	next state
order is completed	feedback data	system is waiting for next order
order is not completed	feedback data	system is regenerating MRP plan

Figure 2.53 Various characterization of the order release state.

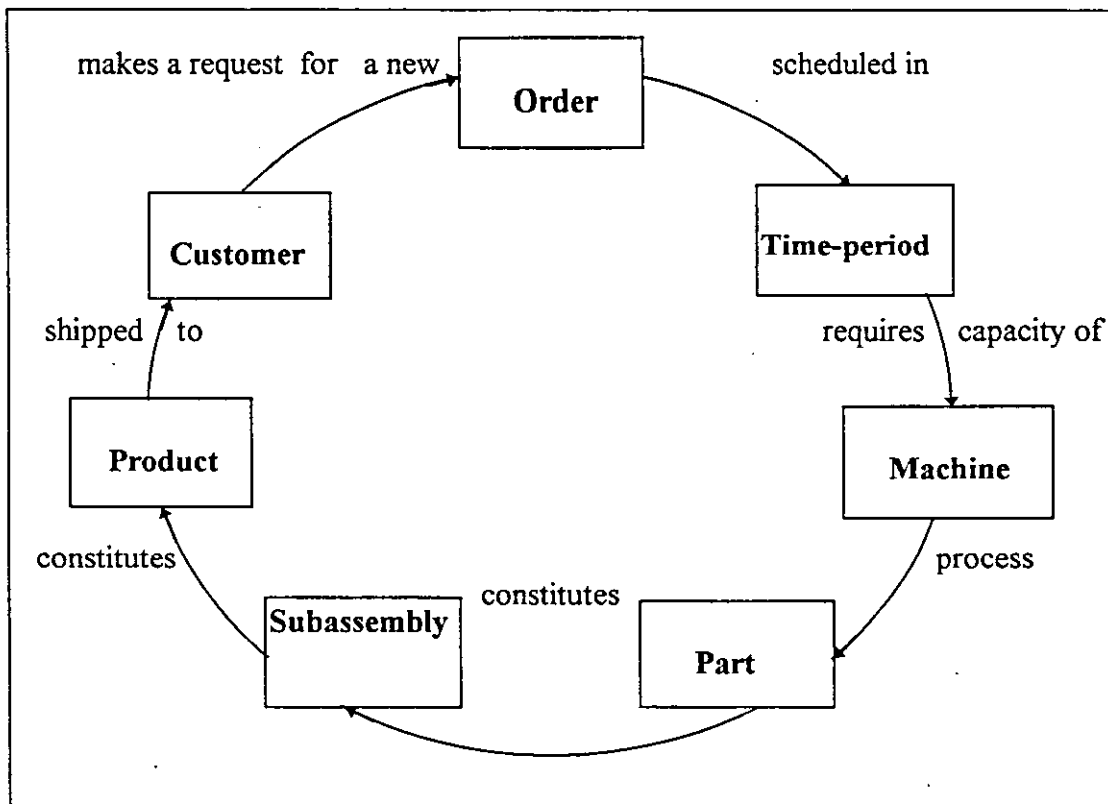


Figure 2.54 Life cycle of order release.

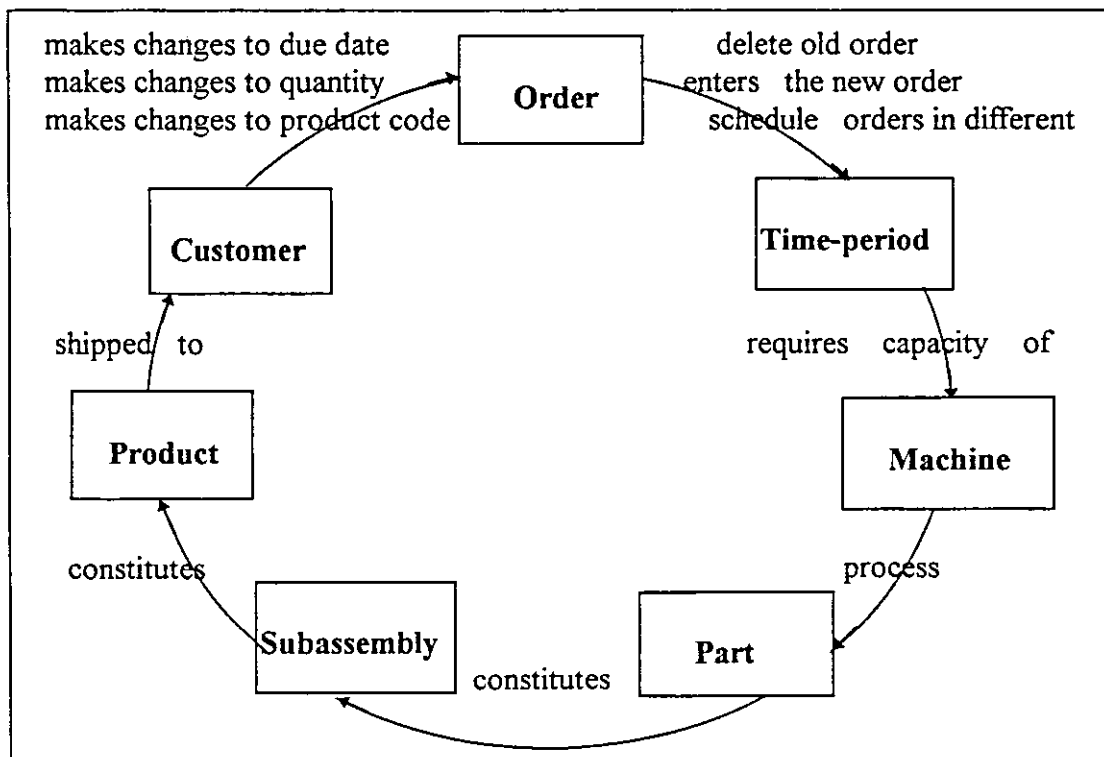


Figure 2.55 Life cycle for the change in order.

#### [2.5.4] STATE DIAGRAMS

A state diagram relates events and states. When an event is received, the next state depends on the current state as well as on the event; change of state caused by event is called a transition. A state diagram is a graph whose nodes are states and whose directed arcs are transitions labeled by event names. A state is drawn as a rounded box containing an optional name. A transition is drawn as an arrow from the receiving state to the target state; the label on the arrow is the name of the event causing the transition. All the transitions leaving the state must correspond to different events.

The state diagram describes the behavior of a single class of objects. Since all instances of a class have the same behavior (by definition), they all share the same state diagram, as they all share the same class features. But as each object has its own attribute values, so accordingly each object has its own state, which is the result of the unique event that it has received. The state diagram specifies the state sequence caused by an event sequence. If an object is in a state and an event labeling one of its transitions occurs, the object enters the state on the target end of the transition. The transition is said to *fire*. If more than one transition leaves a state, then the first event to occur causes the corresponding transition to fire. If an event occurs that has no transition leaving the current state, then the event is ignored. A sequence of events corresponds to a path through the graph.

Following we show the state diagram for the order class.

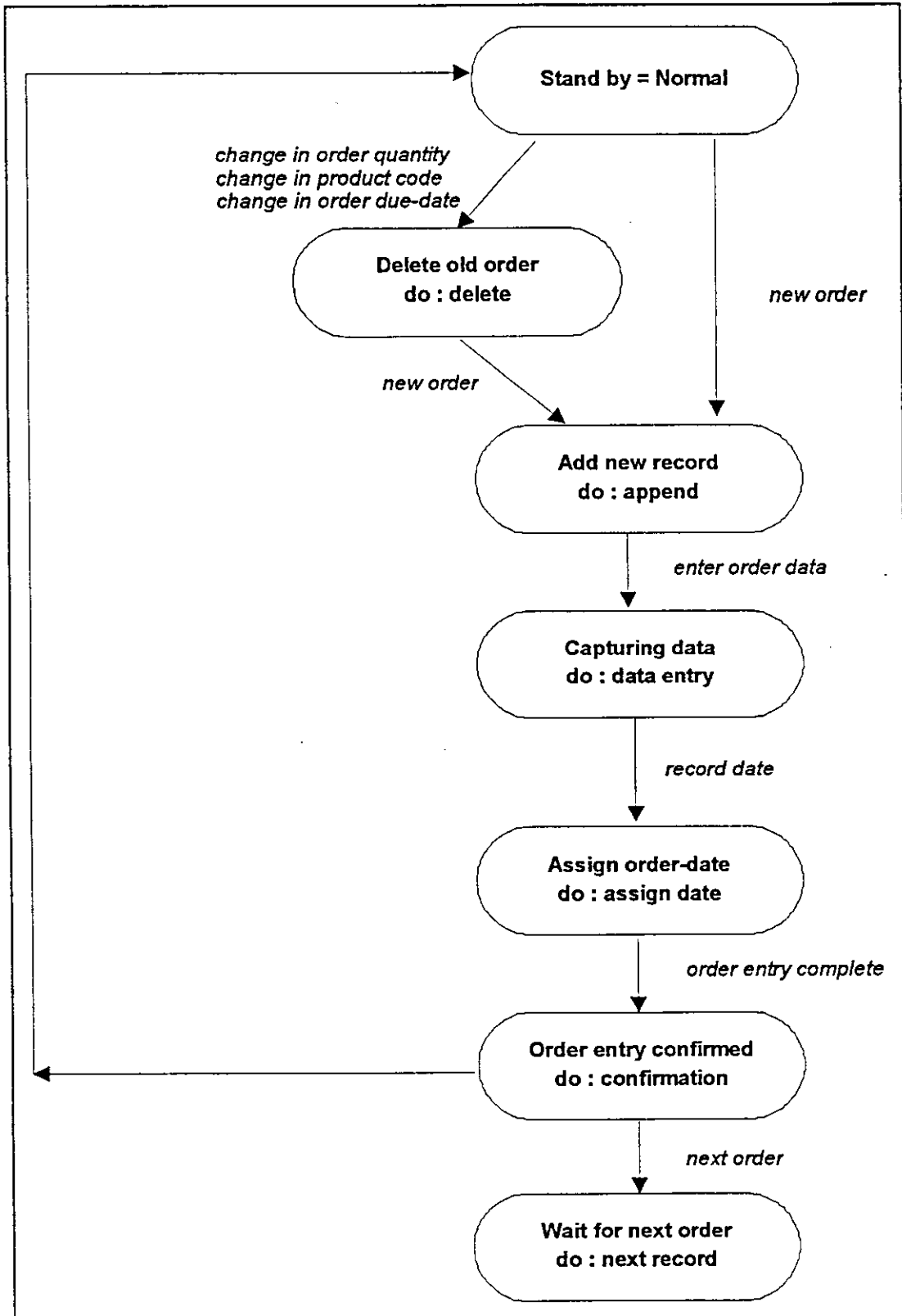


Figure 2.56 State diagram for the order class.

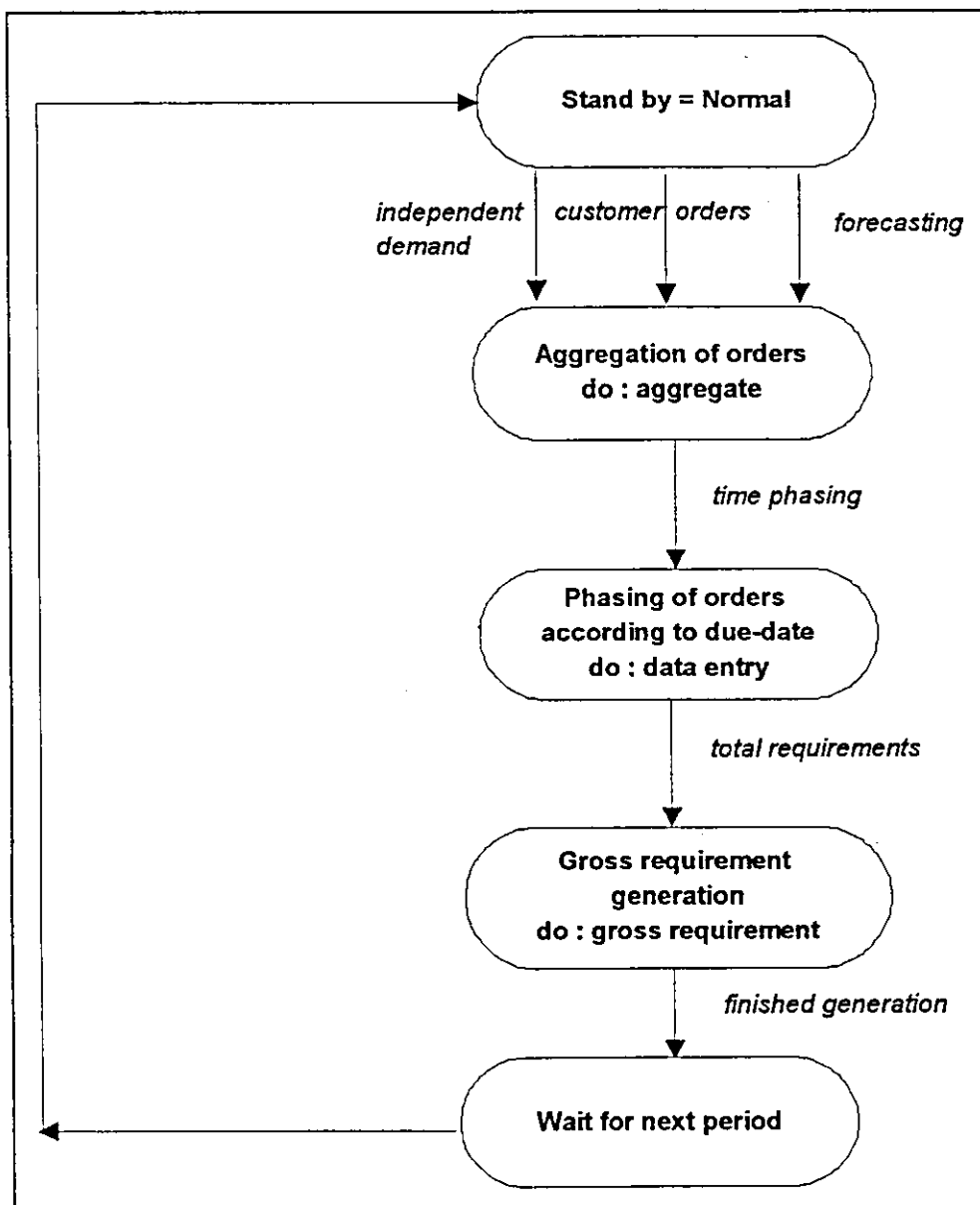


Figure 2.57 State diagram for the time-period class.

### [2.5.5] CONDITIONS

A condition is a Boolean function of object values and is valid over an interval of time.

It is important to distinguish condition from event, which has no time duration. A state can be defined in terms of a condition; conversely, being in a state is a condition.

Conditions can be used as guards on transitions. A guard transition fires when its event occurs, but only if the guard condition is true.

### [2.5.6] OPERATIONS

State diagrams would be of little use if they just described patterns of events. A behavioral description of an object must specify what the objects does in response to events. Operations attached to states or transition are performed in response to the corresponding states or events.

Operations can be classified into two classes:

- An activity, which is an operation that takes time to complete and is associated with a state. Activities include continuous operations
- An action, which is an instantaneous operation and is associated with an event. An action represents an operation whose duration is insignificant compared to the resolution of the state diagram.

An activity represented by the notation “do : A” within a state box indicates that activity A starts on entry to the state and stops on exit. The notation for an action on transition is a slash “/” and the name (or description) of the action, follows the name of the event that causes it.



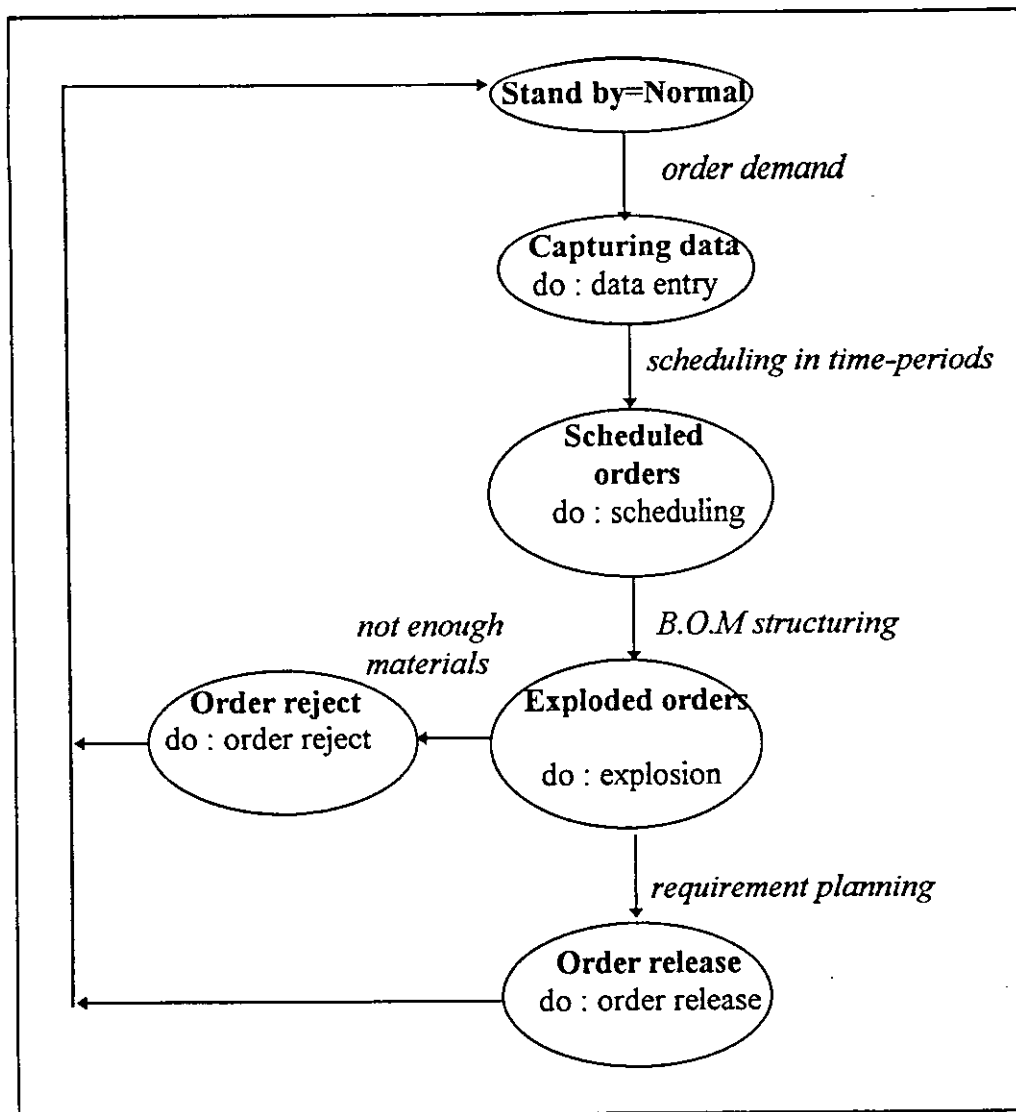


Figure 2.58 State diagram for the order class with activities.

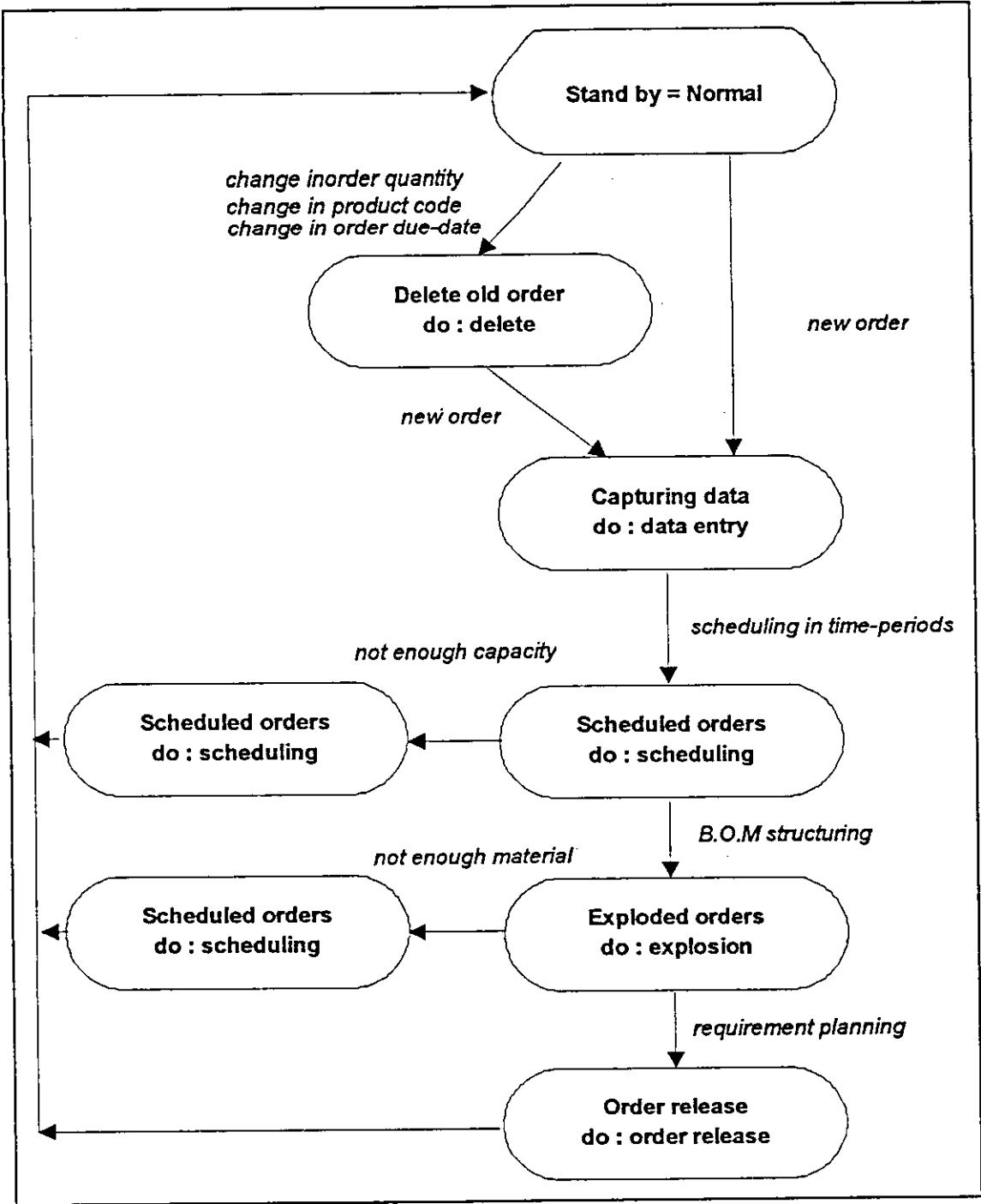


Figure 2.59 State diagram for the change in order class with activities.

## [2.6] FUNCTIONAL MODEL

The functional model describes computations within the system. The functional model is the third leg of the modeling tripod, in addition to the object model and the dynamic model. The functional model specifies what happens, the dynamic model specifies when it happens, and the object model specifies what it happens to. The functional model specifies the results of a computation without specifying how or when they are computed. The functional model specifies the meaning of the operations in the object model and the actions in the dynamic model, as well as any constraints in the object model.

For the scope of our work, we will continue using the OMT methodology. The OMT methodology for the functional model is:

- Identify input and output values.
- Use data flow diagram as needed to show functional dependencies.
- Describe what each function does.
- Identify constraints.
- Specify optimization criteria.

=> Functional Model = data flow diagrams + constraints.

## [2.6.1] IDENTIFYING INPUT AND OUTPUT VALUES

Input and output values are parameters of events between the system and the outside world. Following we illustrate the main input / output values in the MRP system.

### INPUTS

- Customer code
- Order code
- Order date
- Product code
- Order quantity
- Order date
- Order due-date
- Bill of material (B.O.M).
- Part-code
- Subassembly code

### OUTPUTS

- Order release notes.
- Change in schedule for open orders.

## [2.6.2] DATA FLOW DIAGRAMS

A data flow diagram (DFD) shows the functional relationships of the values computed by a system, including input values, output values, and internal data stores. A data flow diagram shows the data values from their sources in objects through processes that transforms them to their destinations in other objects. A data flow diagram contains the followings:

- **Processes** that transform data and are implemented as methods of operations on object classes. A process is drawn as an ellipse containing the process name.
- **Data flows** that move data; i.e. connects the output of an object process to the input of another object or process. It is drawn as an arrow between the producer and the consumer of the data value. The arrow is labeled with a description of the data, usually its name or type.
- **Actor objects** that drives the data flow diagram by producing or consuming data values. An actor is drawn as a rectangle to show that it is an object. Arrows between the actor and the diagram are inputs and outputs to the diagram.
- **Data store objects** that store data passively. Unlike an actor, a data store does not generate any operation on its own but merely responds to requests to store and access data. A data store is drawn as a pair of parallel lines containing the name of the store.

Figure 2.60 shows data flow in the MRP system.

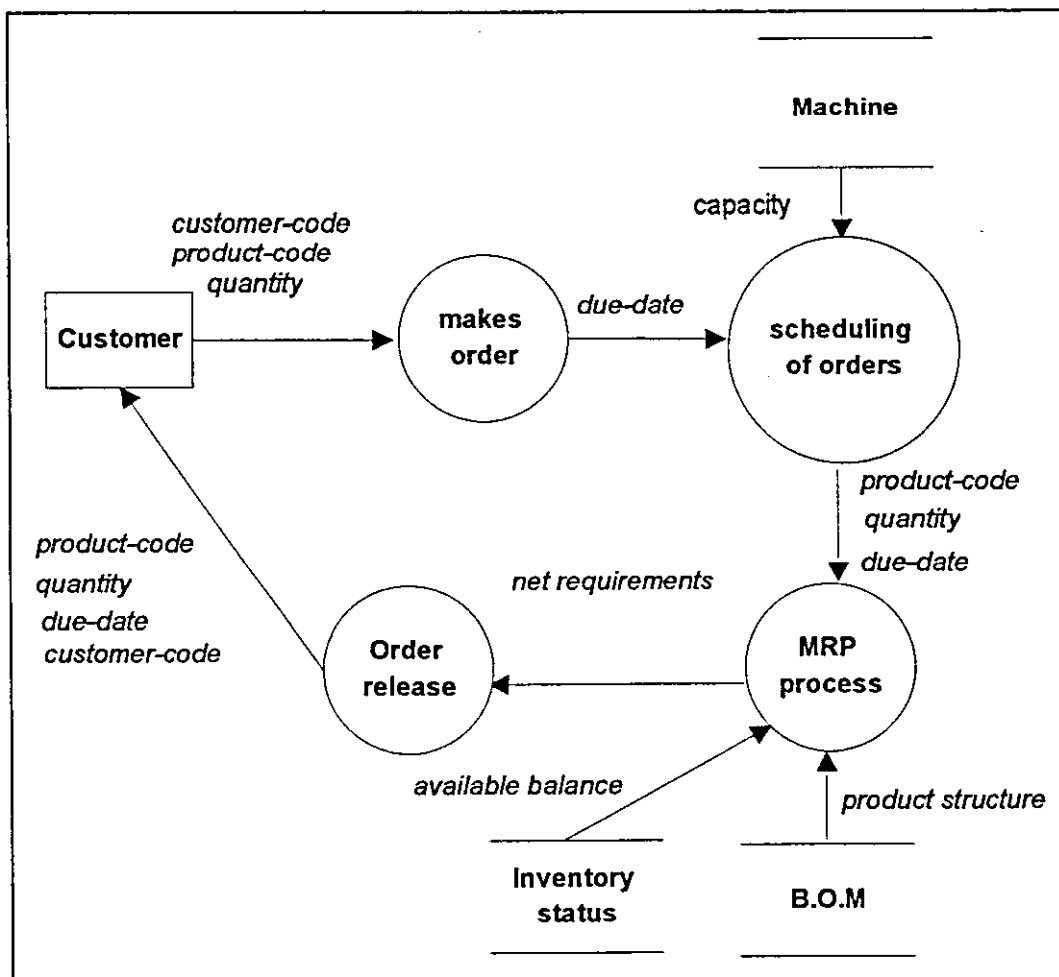


Figure 2.60 Data flow diagram for the MRP system

### [2.6.3] SPECIFYING OPERATIONS

Processes in the data flow diagram must eventually be implemented as operations on objects. Specification of operation includes a signature and a transformation. The signature defines the interface to the operation; i.e., the arguments it requires and the values it returns. The transformation defines the effect of an operation; i.e., the output value as a function of in the input values and the side effects of the operation on its

operand objects. Each operation may be specified in various ways, including the followings :

- Mathematical functions, such as trigonometric functions.
- Table of input and output values (enumeration) for small finite sets.
- Equations specifying output in terms of input.
- Pre- and post-conditions.
- Decisions tables.
- Pseudo code.
- Natural language.

As for simplicity, we will choose the natural language (English type) approach to describe each function.

- The order entry function

Open new order record .

Enter customer code.

Show all customer details.

Enter requested product code.

Locate for product code.

Show all product details.

Enter requested product quantity.

- The scheduling process

Assign due date for each order.

Add all orders for all products at the end of period.

Choose scheduling criteria such as (shortest processing time, tardiness.....)

Check for capacity availability in each scheduled period.

if capacity available    schedule is feasible and continue

else    schedule is infeasible and a reschedule is required.

- The MRP process

Different orders for each product in each time period are added (gross requirements).

For every product in demand, the available quantity at the beginning of the periods is calculated.

For every product in demand, the expected quantities to arrive in different time periods are calculated (scheduled receipts).

For every product, the product lead time is calculated.

For every product, the product lot-size is calculated.

For every product, the product safety stock is calculated.

For every time-period, the net requirement is calculated as follows:

net requirement = gross requirement + safety stock - available stock -  
scheduled receipts.

Repeat those steps for other products



- Order release process

Planned order is released for every product in every time-period if net requirement  $> 0$

if planned order released for certain product  $<$  lot size, then

planned order release for the product = lot size for that product.

else

planned order release = any positive integer  $>$  lot size.

For every net requirement in the time period, the actual order release time period is calculated as follows:

order release period = net requirement period - lead time for this product.

After orders are released for the product in different time periods, the gross requirements of the parts constituting that product are generated as follows:

gross requirement for the part = order release quantity for the product \* the required quantity from this part in different time periods.

Repeat the order release steps that were followed for other products

## **[2.6.4] SPECIFYING CONSTRAINTS BETWEEN OBJECTS**

A constraint shows the relationship between two objects at the same time, or between different values of the same object at different times. Functional constraints specify restrictions on operations. Constraints on functions could be preconditions that the input value must satisfy, or postconditions that the output values are guaranteed to hold. For the MRP system the following constraints exist:

### **[2.6.4.1] PRECONDITIONS**

The product ordered by the customer must exist in the production plan.

The order due-date can not be before the current date.

The order due-date can not be less than the current date + the product lead time.

Order quantity should be positive integer.

### **[2.6.4.2] POSTCONDITIONS**

Order release quantity should be greater than or equal the economical lot-size.

Available stock for any part should not be less than the safety stock.

# CHAPTER THREE

## SYSTEM DESIGN

During analysis, the focus is on *what* needs to be done, independent of the *how* it is done. During design, decisions are made about how the problem will be solved, first at a high level, then at increasingly detailed levels. System design includes decisions about the organization of the system into subsystems, the allocation of subsystems to hardware and software components, and major conceptual and policy decisions that form the framework for detailed design.

The objective of this chapter is to obtain a system design document which consists of the structure of the basic architecture of the system as well as high level strategy decision. In the following sections, we will try to satisfy this objective.

### **[3.1] BREAKING THE SYSTEM INTO SUBSYSTEMS**

The first step in system design is to divide the system into a small number of components. Each major component of the system is called a subsystem. Each subsystem encompasses aspects of the system that share some common property-similar functionality, the same physical location, or execution on the same kind of hardware. A subsystem is not an object nor a function but a package of classes, associations,

operations, events, and constraints that are interrelated and they have a reasonable well-defined and small interface with other subsystems and to the rest of the system. The interface specifies the form of all interactions and the information flow across subsystem boundaries but does not specify how the subsystem is implemented internally. Each subsystem can then be designed independently without affecting the others.

Subsystems should be designed so that most interactions are within subsystems, rather than across subsystem boundaries, in order to reduce the dependencies among subsystems. The relationship between two subsystems can be *client-supplier* or *peer-to-peer*. In a client-supplier relationship, the client calls the supplier, which performs some service and replies with a result. The client must know the interface of the supplier, but the supplier does not have to know the interfaces to its clients, because all interactions are initiated by the clients using the supplier's interface. In peer-to-peer relationship, each of the subsystems may call on the others. A communication from one subsystem to another is not necessarily followed by an immediate response. All subsystems found in our system should be of the client-supplier form.

The MRP system can be defined in terms of the following subsystems:

- The MRP record processing subsystem. In this subsystem, the core of calculations are performed. Taking its input from several subsystems.
- The Bill of material. In this subsystem, the product structure is known and the needs of each product from different parts are calculated.
- The lot-sizing subsystem. In this subsystem, the economical lot-size for each product is calculated.

- The lead-time subsystem. In this subsystem, the lead-time for each product is calculated.
- The safety stock subsystem. In this subsystem, the safety stock for each product is calculated.
- The MPS. In this subsystem, different sources of demand for products are scheduled in the different time periods available in the planning horizon.

## **[3.2] ORGANIZATION OF SUBSYSTEMS**

The above defined subsystems may be organized as a combination of both horizontal Partions and vertical Layers.

### **[3.2.1] VERTICAL LAYERS**

A layered system is an ordered set of a virtual world, each is built in terms of the ones below it and providing the basis of implementation for the ones above it. The following layered subsystem can be found in the MRP system:

- MPS subsystem.
- Bill of material subsystem.
- MRP subsystem.

### [3.2.2] HORIZONTAL PARTIONS

Partions vertically divide the system into several independent subsystems, each providing one kind of service. The following partitions are found in the MRP system:

- Lead-time subsystem.
- Lot-size subsystem.
- Safety stock subsystem.

The following diagram illustrates the different form of subsystems organization:

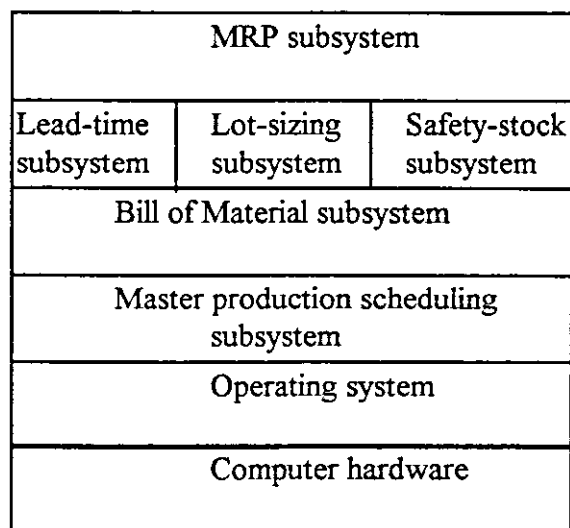


Figure 3.1 Organization of typical MRP subsystems.

### [3.3] SYSTEM TOPOLOGY

In the last section, the top-level subsystems were identified. Now, it is the time to show the information flow among subsystems using a data flow diagram. The following diagram shows the data flow among different MRP subsystems.

In this diagram we followed the *star* arrangement for the different subsystems. This design was adopted because the MRP subsystem is the central subsystem which controls the activities of all other subsystems.

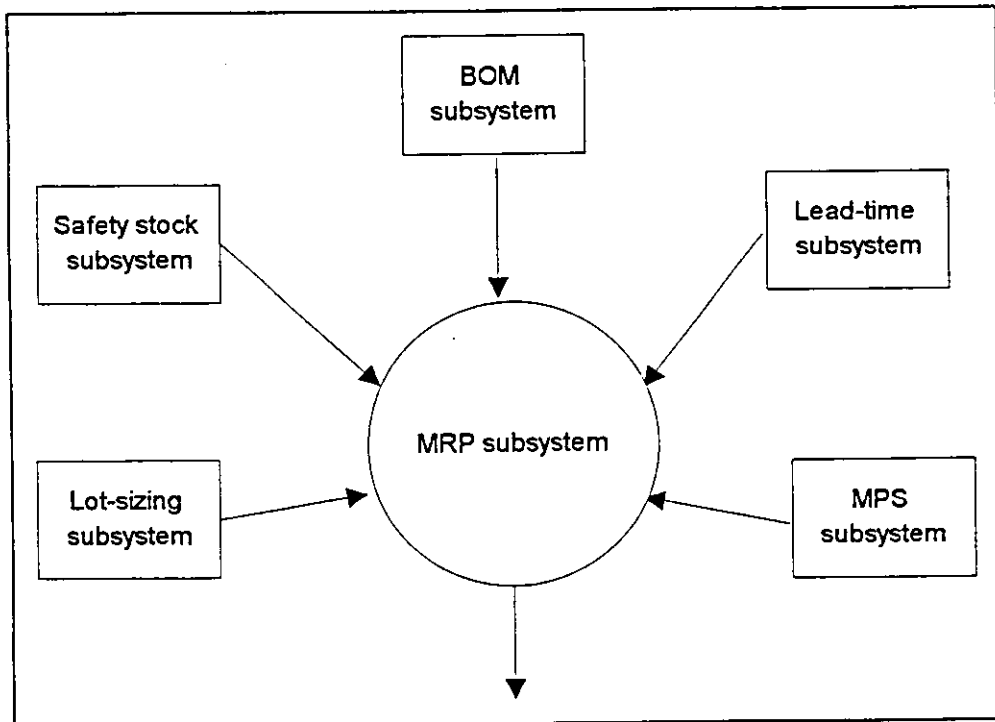


Figure 3.2 Data flow diagram for MRP subsystems

## [3.4] ESTIMATING HARDWARE & OPERATING SYSTEM REQUIREMENTS

It is difficult to estimate the exact hardware requirements needed to execute the MRP process. This is due to the differences in the amount of data manipulated and the processing power needed from one industry to another. However we can offer guidelines for the required specifications to be customized according to every industry's needs.

For the operating system, the choice should take into consideration the portability and multi-tasking problems. By portability, we mean that the operating system can be used by all ranges of computers starting from the microcomputers through the mini to the mainframes. By multi-tasking, we mean that the processor is capable of performing concurrent transactions.

In looking for such an operating system, UNIX stands the best to satisfy these requirements.

As for the hardware, the following guidelines can help in determining the hardware requirements:

- For small companies encompassing little amount of data, it is recommended to have a PC to run the MRP system.
- For medium to large size companies, it is recommended to go for the distributed data approach, which is the client-server approach. The client-server approach was



advent as a result of the revolution in data communication and software technology.

In this approach, the clients could be PC-workstations and the server to be a mini or mainframe according the capabilities needed. The server is to be located in the planning department, while the work stations are to be located according to their need in the different manufacturing departments.

### **[3.5] MANAGING OF DATA STORES**

The internal and external data stores in a system provide clean separation points between subsystems with well-defined interfaces. Data stores may be files or databases . As for our system we will choose databases managed by DBMS to be the form of data stores.

The reasons for that choice are :

- Databases are powerful and make applications easier to port to different hardware and software platforms.
- Databases provide many infrastructure features; such as crash recovery, sharing between multiple users, data integrity, extensibility, and data distribution.
- Databases provide common interfaces for all application.

### **[3.6] SYSTEM ARCHITECTURE**

As for the system architecture, the transaction manager architecture is chosen. There are many other architecture systems such as, batch transformation, continuous transformation, interactive interface, dynamic simulation, and real-time systems.

The transaction manager is a database system whose main function is to store and access information. The information comes from the application domain. Most transaction managers deal with multiple users and concurrency. A transaction must be handled as a single atomic entity without interface from other transactions. The transaction manager is the most suitable solution for the inventory problem.

### **[3.7] HARDWARE TRADE-OFF PRIORITIES**

In our choice and design of the hardware system, we must set priorities that will be used to guide trade-offs during the rest of the design. Setting trade off priorities is at best vague, because priorities are rarely absolute. According to our philosophy in designing this system, the following priorities must be satisfied:

- Performance in terms of integer point processing.
- Memory
- Disk storage
- Cost

All these parameters should be combined in an optimization process in order to obtain the desired performance.

## **CHAPTER FOUR**

### **RESULTS, DISCUSSION, & RECOMMENDATION**

#### **[4.1] RESULTS**

After carrying out the processes of analysis, system design and object design, we can now get a system that is integrated in such a manner that it is ready for implementation with not much effort. All the necessary objects were formulated and modeled including the static and dynamic structures. The different states of objects were also obtained and the operations done on the object were also stated. The resulting design of the system allows the implementor to code every operation using the code he prefers.

## [4.2] DISCUSSION

This project was intended to cover the order processing problem found in most industries. From the analysis, it was discovered that, the complete order processing cycle is a very complicated and huge problem to be handled in a single project. Actually, only a team of experts can do the whole thing. As for the scope of this project, only one stage was chosen and was extensively investigated this stage was the MRP stage. The choice of the MRP was made for its importance in every industry and if handled successfully, this solution will provide a strong infrastructure for other stages to be implemented using the same approach.

In the introduction, the MRP system was introduced and the problems facing its execution were highlighted. It is worth noting that these kind of problems become evident, especially if the MRP system has to implemented in a CIM environment. In such a case, the processing of the MRP becomes a real nightmare and the manufacturing system resources will not be utilized efficiently. As a solution to these problem, the theme of this project was originated. The philosophy behind attacking these problems was to approach the MRP system from another perspective; which is the object oriented approach. This decision was made after careful evaluation of the possible solutions. Although the decision is made in a time where the relational databases are the de facto, but all trends are now moving towards the object-oriented approach. The object-oriented paradigm is rising very fast and it will be the commercially dominating solution in the following years to come.

From the object-oriented approach, the object modeling technique (OMT) was chosen. This technique was first introduced in the early 90's and since that time, it has gained a great amount of attention and is widely used in the object-oriented world.

Using this technique, it was possible to model the MRP system as consisting of objects. Each object having a unique existence and interacting with other objects via messages. All necessary data and operations needed for that object are encapsulated inside it. Opposing to the relational approach, which tries to scatter the data and operations through relations. This approach makes the object self dependent having all what it needs inside it. In the object model the static structures of the objects were formulated. These structures came from real-world existence. After that attributes and operations were appended to objects. The behavior of the objects were stated in the dynamic model in order to identify the events and triggers, so that objects could perform actions. The functions required by the system were shown in the functional model.

In modeling the system in such a manner, the powerful features of the object-oriented approach were enjoyed. The feature of inheritance facilitated defining subclasses such as parts, subassemblies, and products in term of the superclasses; which were the materials. In this manner much of the common attributes will be inherited from the superclass to the subclasses and only especial attributes of the subclasses were in need to define them. This feature will be valuable when time comes to implement the code, where common attributes has to be coded only once. In this way, much of effort needed to write codes for every object can be saved.

The Encapsulation features represented in the objects made the problem of inconsistency not existent. All the necessary data and operations are defined inside the walls of the objects. This feature frees the implementor from enforcing external checking for constraints, because checking is done implicitly.

The extensibility feature which is presented in the object-oriented approach makes it easy for upgrading and maintain the system after being implemented. This feature is highly appreciated especially in the cases of large projects, such the MRP system.

Following, a brief comparison between the OODBMS and the RDBMS will be introduced.

#### **[4.2.1] DATA TYPES**

In RDBMS there is only one generic type for data structuring, namely the relational type. By definition, attributes of normalized relations are non-decomposable and in practice they tend to have relatively simple data types (integer, real, string, date, etc.). Operations on relations are restricted to retrieving and updating tuples identified by attribute values. Efforts have been made to extend the typing capabilities of relational systems and also to relax the requirement that attributes must be non-decomposable.

An object-oriented database stores class definitions and instantiations of these classes. Class definitions are the analogue of schemes in other database systems. but with the important additional feature that classes encapsulate the behavior of the object by packaging operations with the data structure. There is a class definition associated with

each object type and the operations that can be applied to instances of that class may be customized to the object. The properties of classes need not be simple data types but can be references to other classes of arbitrary complexity. Object-oriented systems typically provide a range of classes implementing frequently used data types. Users can tailor these to the needs of their own application using the facilities afforded by genericity and inheritance.

#### **[4.2.2] DATA INTEGRITY**

The normalization process required by the relational systems represents a tedious job to be done. Moreover, normalization tries to scatter data among different tables and relations in order to enforce referential constraints. Often this burden is placed on the application developer. Some systems do offer facilities for the explicit specification of referential integrity constraints, but even in these cases it is the responsibility of the database implementor to ensure that all such constraints are included. The system then assumes responsibility for enforcing the constraints at run-time, but this can be a significant overhead.

The relational model is incapable of expressing integrity constraints with greater semantic content than straightforward referential integrity. For example it is not possible to express the fact that a relationship is one-to-one or one-to-many. Such constraints must be built into the application code that manipulates the relational database. Since such code is not generally shared among all applications it is difficult to ensure that data will be updated consistently at all times. By contrast, in the object-oriented model a class defines a data abstraction and this abstraction includes a specification of the operations

(methods) that can be applied to instances of the class. By defining the database in terms of such abstractions a high degree of data independence is achieved. That is, it is possible to alter the way in which a class is implemented without affecting other classes or transactions that make use of the abstraction. Also, a user can derive new classes from existing classes by supplying a specification, and an implementation for any new properties or operations. These properties and operations are implemented by means of general-purpose programming language and so any feasible user-defined operation can be constructed and stored in the database. An operation on an object can take the place of many database operations and every transaction which invokes the operation uses exactly the same code - the procedure stored with the class definition. This has significant benefit for data integrity since consistency checks may be incorporated in the procedures. Also, by storing operations as part of the database a large part of the application code is under the control of the database administrator and can be managed by all of the usual database management utilities (i.e. facilities for concurrency control, recovery, version control and security).

Entity integrity is also handled rather differently in object-oriented systems. All objects (class instantiations) have a unique identity and other objects can refer to that identity. An object retains its identity through arbitrary changes to its own states. This is in contrast to the relational model where the properties of an entity must be sufficient to distinguish it from all other entities. That is, there must be an attribute or attributes whose values uniquely and immutably identify that entity. The properties of uniqueness and immutability are not always present in the real world and it is often necessary in relational databases to introduce artificial identifiers for entities.



### [4.2.3] SCHEMA EVOLUTION

Database systems are characterized by their trend to evolve around a number of fundamental object types which form the kernel of the system. As the system evolves, new views of the data are added by restricting or extending existing object descriptions, and new applications on these views are generated using existing code modules as a basis. RDBMS tend to offer very limited facilities for the expansion or modification of existing data structures. For example, commercial relational systems typically allow the dynamic creation and deletion of relations and the addition of new columns to a relation. Changes to domain types will usually involve the rewriting of the relations involved and changes to the applications that use them. This is largely due to the fact that the data structures compromising the database schema and the application programs are very loosely coupled. The tight coupling between applications and data in the object-oriented model offers considerably more scope for schema evolution through the extension and refinement of existing data structures and the effective reuse of application code. The richness of the model permits the semantics of schema evolution to be rigidly defined and validated.

Finally, it should be stated that the object-oriented approach was able to meet the requirements of an efficient solution for the MRP system from its analysis and design features, but a powerful commercial object-oriented database language is still in need to fully utilize the wonderful features of the object-oriented analysis and design.

### **[4.3] RECOMMENDATION FOR FUTURE WORK**

In our analysis of the MRP system, care was taken in regard to the integration of this module to other modules of the order processing life cycle. In the design phase, the concepts of extensibility and modularity made the system flexible for any change or upgrade. As for the scope of this project, only one part of the order processing cycle was fully investigated; which is the MRP. It is recommended that the followings be done in order to get a complete system that effectively handles the order processing problem:

- The analysis and design document presented in this project should be implemented using a commercial object-oriented database language and performance evaluation should be done to ensure that analysis and design meet their objectives.
- Other modules of the order processing life cycle should be also analyzed, designed, and implemented using the same techniques used in this project. After all modules are available, they should be integrated in one system.

## REFERENCES

- [1] Ajlouni, M. : "A Study Of The Frequency Of Dynamic Scheduling On Job Shop", MSc. Thesis, University of Jordan, Amman, Jordan, 1995.
- [2] Vollmann Thomas E., Berry, William, L., Whybark, D. Caly : Manufacturing planning and control systems, 3rd. edition, Irwin, Homewood, IL 60430, Boston, 1992.
- [3] Beit Shaweish, M. : "Hardware Requirements", CEB Company, personal communications, Amman, Jordan, 1995
- [4] Baid, N.K. Nagarur, N.N. : " Integrated decision support systems for FMS: using intelligent simulation", International Journal of Production Research, Vol. 32, n4, April 1994, pp. 951-965.
- [5] Muller, Daniel J. Jackman, John K. Fitzwater, Charles : " A simulation-based work order release mechanism for a flexible manufacturing system" Proceedings of Winter Simulation Conference, Piscatway, NJ, USA (IEEE cat. n. 90CH2926-4), 1990, pp. 599-602.
- [6] Mori, Masatoshi Kuriyama, Sennosuke : " Pull Logic Manufacturing based on CIM to approach J.I.T", Eleventh IEEE/CHMT International Electronics Manufacturing Technology Symposium, San Francisco, CA, USA, Sep. 26-18, 1991, p. 134-137.
- [7] Luk, Albert : "Fujikama goes Unix", Proceedings of the 1991 IEEE Pacific Conference on Communications, Computers and Signal Processing, Victoria,

- BC, USA, May 9-10, 1991, published by IEEE Service Center, Piscatway, NJ, USA (IEEE cat. n. 91CH2954-6), pp. 783-786
- [8] Li, Rong-Kwei Shyu, Yu-Tang Adiga, Sadashiv : Heuristic rescheduling algorithm for computer based production scheduling systems, International Journal of Production Research, Vol. 31, n8, August 1993.
- [9] Nakashima, Yusei Daito, Noriaki Fujita, Satoru : Integrated expert system with object oriented database management system, Systems & computers in Japan, Vol. 23, n11, 1992
- [10] Chung, Yonkung Fischer, Gary W. : "Illustration of object-oriented databases for the structure of a bill of materials", Computers & Industrial Engineering, Vol. 19, n3, June 1992, pp. 257-270.
- [11] Chaharsooghi, Sayed Kamal : "Manufacturing planning and control in small companies", Ph.D. Thesis, University of Hull, United Kingdom, 1987.
- [12] Chung, Yonkung Fischer, Gary W. : "Conceptual structure and issues for the object oriented bill of materials (BOM) data model", Computers & Industrial Engineering, Vol. 26, n2, April 1994
- [13] Hatfield, John Allen : Application of artificial intelligence in chemical processes, Ph.D. Thesis, Rutgers The State University Of New Jersey, New Brunswick, 1991.
- [14] Winter, Robert E. : "On the utilization of an active, integrated database for the vertical integration of production planning and control", Computer applications in production and Engineering : Integration Aspects, IFIP, 1991.

- [15] Hughes, J. G. : Object-oriented databases, 1st. edition, Prentice Hall international (UK) Ltd., 1991
- [16] Fred R. Mcfadden, Jeffrey A. Hoffer : Database management, 3rd. edition, The Benjamin/Cummings Publishing Company, Inc., 390 Bridge Parkway Redwood City, California 94065.
- [17] Rambaugh James, Blaha Michael, Premerlani William, Eddy Frederick, Lorensen William : Object-oriented modeling and design, Prentice Hall International, Inc.1991.
- [18] Orlicky, Joseph : Material requirement planning, McGraw-Hill Book Company, 1975.

## APPENDIX (A)

## GLOSSARY

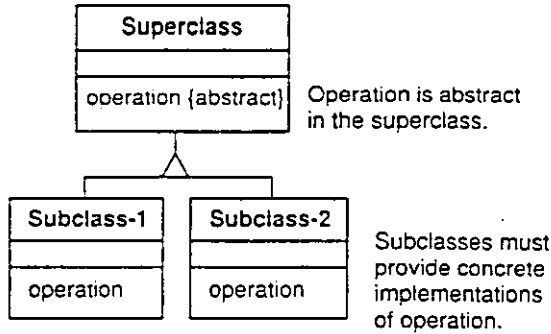
- **MRP** : Material Requirements Planning.
- **OMT** : Object Modeling Technique.
- **OOA** : Object Oriented Analysis.
- **OODBMS** : Object Oriented Database Management System.
- **RDBM** : Relational Database Management System.
- **BOM** : Bill Of Material.
- **OEM** : Original Equipment Manufacturer
- **OOBOM** : Object-Oriented Bill Of Material.
- **RM** : Relational model.
- **MPS** : Master Production Scheduling
- **CPU** : Central Processing Units
- **DFA** : Data Flow Diagram

## **APPENDIX (B)**

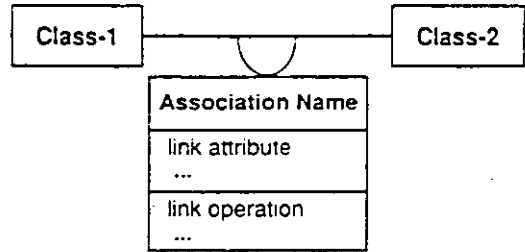


# Object Model Notation Advanced Concepts

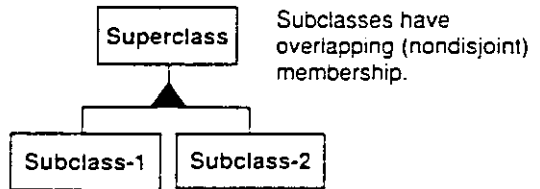
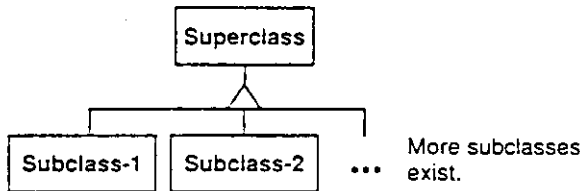
## Abstract Operation:



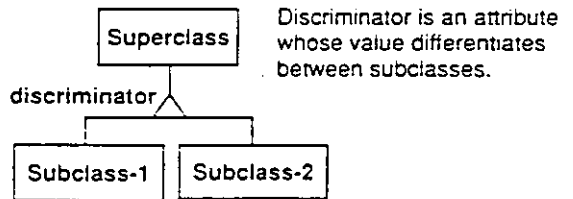
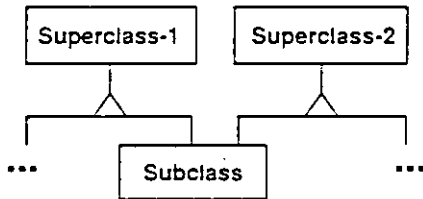
## Association as Class:



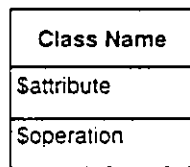
## Generalization Properties:



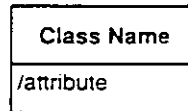
## Multiple Inheritance:



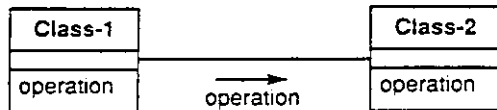
## Class Attributes and Class Operations:



## Derived Attribute:



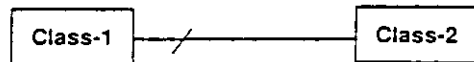
## Propagation of Operations:



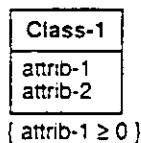
## Derived Class:



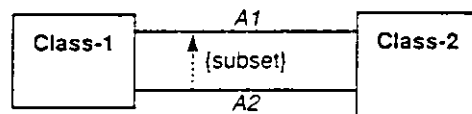
## Derived Association:



## Constraints on Objects:

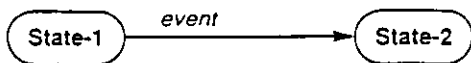


## Constraint between Associations:

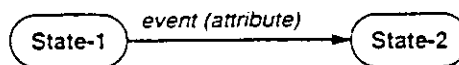


## Dynamic Model Notation

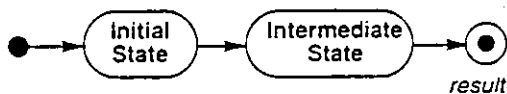
Event causes Transition between States:



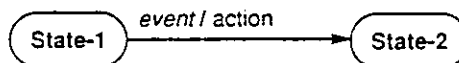
Event with Attribute:



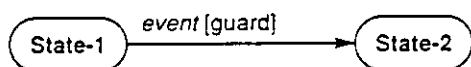
Initial and Final States:



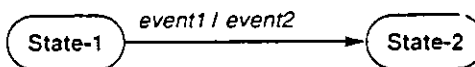
Action on a Transition:



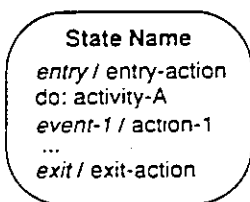
Guarded Transition:



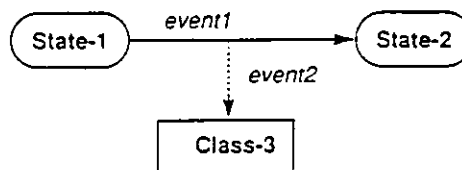
Output Event on a Transition:



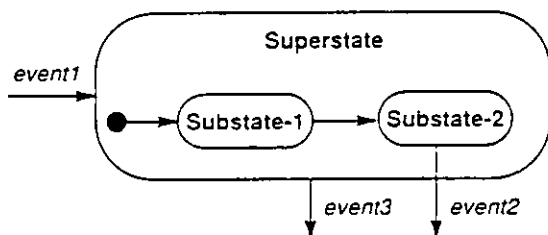
Actions and Activity while in a State:



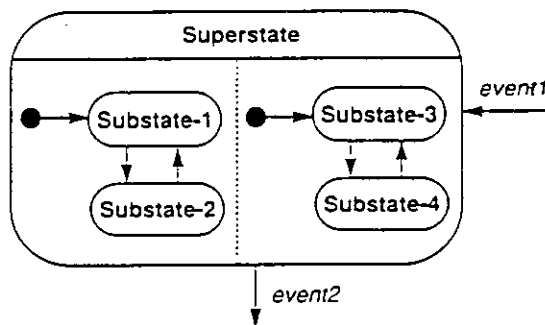
Sending an event to another object:



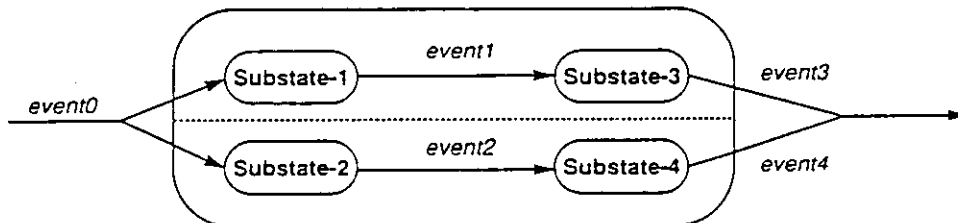
State Generalization (Nesting):



Concurrent Subdiagrams:



Splitting of control:



Synchronization of control:

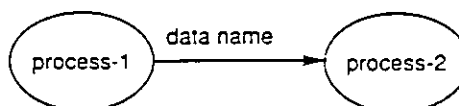
SOURCE: OBJECT-ORIENTED MODELING AND DESIGN, [15]

## Functional Model Notation

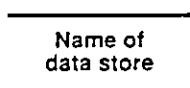
Process:



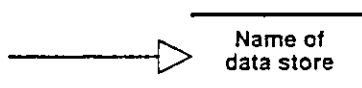
Data Flow between Processes:



Data Store or File Object:



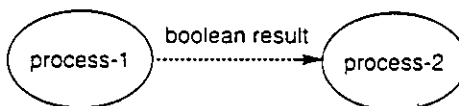
Data Flow that Results in a Data Store:



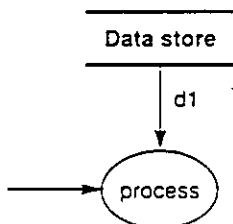
Actor Objects (as Source or Sink of Data):



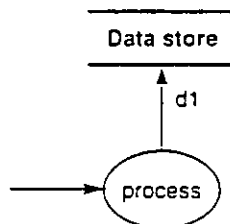
Control Flow:



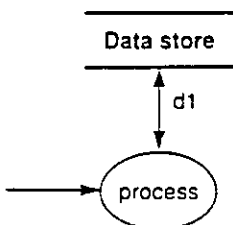
Access of Data Store Value:



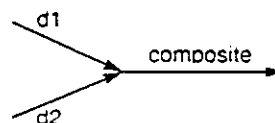
Update of Data Store Value:



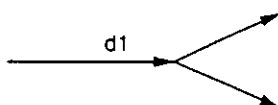
Access and Update of Data Store Value:



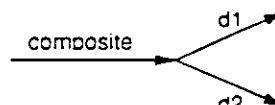
Composition of Data Value:



Duplication of Data Value:



Decomposition of Data Value:



**SOURCE: OBJECT-ORIENTED MODELING AND DESIGN. [15]**

**APPENDIX (C)**

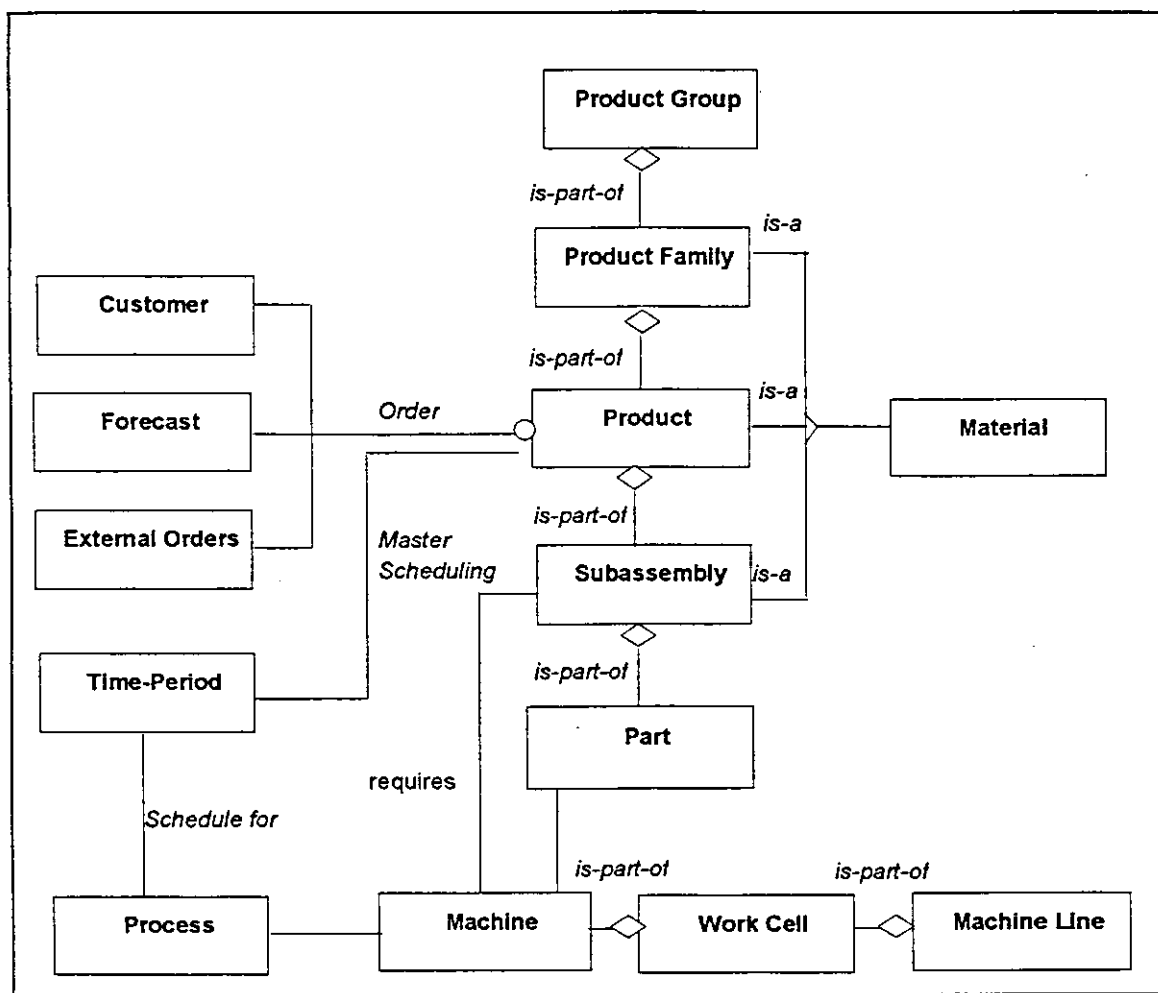


Figure C.1 Complete MRP schema