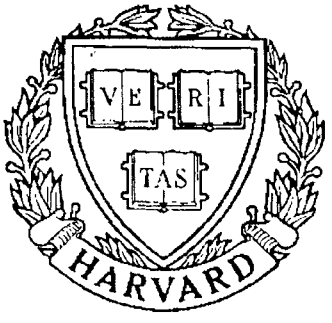
THESIS REPORT
Master's Degree

S Y S T E M S
R E S E A R C H
C E N T E R



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Integration of Manufacturing Resource Planning (MRP II) and Computer Aided Design (CAD) Through Database Interoperability

by: M.E. Bohse
Advisor: G. Harhalakis

M.S. 87-3
Formerly TR 87-159

Integration of Manufacturing Resource Planning (MRP II) and Computer Aided Design (CAD) Through Database Interoperability

by
Michael Edward Bohse

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Master of Science
1987

Abstract

Title of Thesis: Integration of Manufacturing Resource Planning (MRP II) and Computer Aided Design (CAD) Through Database Interoperability

Name of Candidate: Michael Edward Bohse, Master of Science, 1987

Thesis directed by: Dr. George Harhalakis, Assistant Professor, Department of Mechanical Engineering

The traditional, fragmented approach to increasing manufacturing efficiency has resulted in "islands of automation" in our factories. Computer Integrated Manufacturing (CIM) is the goal of tying together these islands into a single coherent system capable of controlling an entire manufacturing operation. The technical and organizational difficulties of such a massive undertaking require a modular approach to CIM implementation, with an initial nucleus being gradually expanded by allowing interaction between it and other systems' databases. Manufacturing Resource Planning (MRP II) is best positioned to serve as this nucleus. The suggested first system for integration is Computer Aided Design (CAD); the integration being centered around part specification, product structure, and engineering changes.

A functional model of the MRP II/CAD integrated system, detailing the logical interactions between the systems in these areas, is presented. In it, CAD serves as the center for design related decisions, while MRP II handles the task of executing and monitoring manufacturing activities. Integration is simulated using an interoperability system which allows for the specification of a set of rules to define and enforce the update and retrieval dependencies of the databases.

Acknowledgements

I wish to express my sincere gratitude to Dr. Harhalakis, whose guidance, support, and advice throughout this research were invaluable. I am also grateful to Dr. Mark and Bobbie Cochrane of the Computer Science Department for providing me with an Update Dependency Language interpreter so quickly, and for their considerable assistance with the language.

In addition, I wish to thank the members of my thesis committee, Dr. Harhalakis, Dr. Mark, and Dr. Ssemakula, for their time, and their comments and suggestions.

Further, I wish to express my appreciation to the Systems Research Center and the Department of Mechanical Engineering, especially Dr. Fourney, for providing financial support during this project.

Finally, I would like to thank my family and friends for their understanding and support. I would especially like to thank the other members of the "CAPS" Lab, whose help and friendship has meant a great deal: Tariq, Debbie, Phani, Susan, Ramesh, Philippe, Anshuman, and Lin.

Contents

1 Introduction **1**

2 A Proposed Model of CIM Architecture **6**

2.1 Manufacturing Resource Planning (MRP II) 7

2.2 Computer Aided Design 12

2.3 Computer Aided Process Planning 14

2.4 Computer Aided Manufacturing 15

3 A Proposed CIM Design Strategy **18**

3.1 Management of CIM Data 21

3.2 A Starting Point: MRP II/CAD Integration 25

3.3 The Fundamental Similarities of MRP II and CAD 27

4 Functional Design of a Model MRP II/CAD Integrated System **33**

4.1 Model Assumptions and Framework 33

4.1.1 Designation of Authority between MRP II and CAD 34

4.1.2 Control of Information Flow in the Model 35

4.2 Part Master Data 37

4.2.1 Data Division and Organization 41

4.2.2 Adding New Parts Via CAD 42

4.2.3 Adding New Parts Via MRP II 48

4.2.4 Adding New Revisions Via CAD 53
4.2.5 Adding New Revisions Via MRP II 57
4.2.6 Making Parts Obsolete 61
4.2.7 Deleting Parts 63
4.3 Product Structures 66
4.3.1 Adding Component Relationships Via CAD 72
4.3.2 Adding Component Relationships Via MRP II 76
4.3.3 Deleting Component Relationships 77
4.3.4 Substituting Components in Relationships 79
4.3.5 Modifying Component Quantities 81
4.3.6 Copying Component Relationships from one Assembly to Another 82

5 Database Interoperability through Update Dependencies 84
5.1 Database Interoperability 84
5.2 Update Dependencies 88
5.2.1 Syntax 90
5.2.2 Semantics 92
5.2.3 Implementation Strategy 94

6 Demonstration and Discussion of Results 97
6.1 Implementing the Model System Using Update Dependencies 97
6.1.1 Relations Used by the System 98
6.1.2 Programing the Interoperability System 104
6.1.2.1 Backtracking 105
6.1.2.2 "Failure Alternatives" 107
6.1.2.3 System Performance 108

6.1.3 Problems with the Current Interoperability Model 110
6.1.3.1 Duplication of Application Functions 110
6.1.3.2 Operation "Chains" 110
6.1.3.3 User Interface 112
6.2 A Sample Session with the Model System 113
6.2.1 Part Master Data 113
6.2.1.1 Adding New Parts Via CAD 113
6.2.1.2 Adding New Parts Via MRP II 117
6.2.1.3 Adding New Part Revisions Via CAD 119
6.2.1.4 Adding New Part Revisions Via MRP II 122
6.2.1.5 Part Supersession 124
6.2.1.6 Hold Functions in CAD and MRP II 129
6.2.1.7 Part Deletions 133
6.2.2 Product Structures 138
6.2.2.1 Adding Component Relationships Via CAD 139
6.2.2.2 Adding Component Relationships Via MRP II . . . 143
6.2.2.3 Adding Component Relationships Requiring a New Assembly Part Number 144
6.2.2.4 Deleting Component Relationships Via CAD 147
6.2.2.5 Deleting Component Relationships Via MRP II . . . 149
6.2.2.6 Mass Substitution of Components Via CAD 149
6.2.2.7 Mass Substitution of Components Via MRP II . . . 152
6.2.2.8 Modifying the Quantity of Components per Assembly 154
6.2.2.9 Copying Product Structures Via CAD 155
6.2.2.10 Copying Product Structures Via MRP II 157

7 Conclusions **160**

8 Recommendations for Further Work **164**

Bibliography **168**

A Update Dependency Operations for the Model MRP II/CAD System **171**

List of Tables

4.1 Part Data Maintained by MRP II and CAD 41

4.2 Revision Data Maintained by MRP II and CAD 42

4.3 Product Structure Data Maintained by MRP II and CAD in Relationship Records. 68

List of Figures

2.1 Potential CIM Architecture. 7

2.2 Typical MRP II Architecture. 8

2.3 MRP II from a Software View. 10

2.4 MRP II Linked to CAD. 13

2.5 CAPP Added to the MRP II/CAD System. 15

2.6 CAM Added to the MRP II/CAD/CAPP System. 17

3.1 A Systems Representation of Manufacturing. 18

3.2 The Single Database Concept of CIM Data Management. 23

3.3 The Multiple Database Concept of CIM Data Management. 24

3.4 System Representation of MRP II/CAD Information Flow. 26

3.5 A Typical Bill of Material. 29

4.1 Status Diagram for Adding New Parts Via CAD. 44

4.2 Status Diagram for Adding New Parts with Released Status Via MRP II. 50

4.3 Status Diagram for Adding New Parts with Hold Status Via MRP II. 51

4.4 Status Diagram for Adding New Revisions Via CAD. 54

4.5 Status Diagram for Adding New Revisions with Released Status Via MRP II. 58

4.6 Status Diagram for Adding New Revisions with Hold Status Via MRP II. 59
4.7 Status Diagram for Making Parts Obsolete Via CAD from Released Status. 61
4.8 Status Diagram for Making Parts Obsolete Via CAD from Hold Status. 62
4.9 Status Diagram for Deleting CAD-Generated Parts Via CAD. 64
4.10 Status Diagram for Deleting CAD-Generated Parts Via MRP II. . . 65
4.11 Status Diagram for the Deletion of MRP II-Generated Parts Via MRP II. 66
4.12 Transfer of Component Relationships from CAD to MRP II beginning with Assemblies having Working Status. 74
4.13 Transfer of Component Relationships from CAD to MRP II beginning with Assemblies having Hold or Released Status. 75
4.14 Transfer of Component Relationships from MRP II to CAD. 78

5.1 A Global Schema between the Databases and the System. 85
5.2 A Global Schema between the Users and the Databases. 86
5.3 Database Integration. 88
5.4 Database Interoperability. 89

6.1 A Demonstration of Backtracking. 105

Chapter 1

Introduction

It is no secret that United States manufacturing industries have, in recent years, experienced an ever-increasing level of foreign competition. Once the undisputed leader in the design and production of manufactured goods, the U.S. is now seeing many of its industries lost to foreign countries. Today there is only one American maker of television sets left; the vast majority of radio and small home appliance manufacturers went overseas years ago. The traditionally strong U.S. steel and auto industries are likewise under fire. By 1990, the U.S. will be importing a flood of new cars produced in countries such as Czechoslovakia, Greece, Korea, Spain, Taiwan, and Yugoslavia. Even the newer, high technology fields of computers and semiconductors feel the effects of foreign competition.

Though there are many complex reasons for the success of foreign manufacturers, one of the most often cited is that of cheaper labor, which seems to be justified by the following statistics. In 1984, the average hourly manufacturing wage in the United States [7] was $12.59. In West Germany, the rate is 76 percent of that amount; in Japan, 56 percent. Similarly, in Great Britian, the rate is only 46 percent of this amount; in South Korea, 11 percent; and in Brazil, 10 percent. To make matters worse, the productivity of American workers, measured by manufacturing output per worker-hour, now lags behind that of every other industrialized nation [7].

Further, the problem of foreign competition is compounded by a perception by a considerable number of American consumers that the quality of foreign products is higher than that of similar American-made products. Over the last several years, for example, consumers have been willing to pay premiums on imported automobiles, most notably those from Japan, whose numbers have been limited by trade agreements.

It is obvious that a major overhaul of American industry is necessary if the U.S. is to remain competitive in the world market. The labor figures mentioned above make it clear that the reliance on traditional, labor intensive manufacturing systems and techniques must be reduced. New methods, technologies, and systems must therefore be employed, not only to increase productivity, but also to improve product quality and responsiveness to changes in market demands and conditions.

To this end, manufacturers have been literally inundated with technologies promising improved productivity, reduced costs, and improved quality, all supposedly leading to the "factory of the future." These include some of the hottest buzzwords since "interchangeable parts": Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Flexible Manufacturing Systems (FMS), Manufacturing Resource Planning (MRP II), Group Technology (GT), Just-In-Time inventory control (JIT), Automated Materials Handling (AMH), Computer Aided Process Planning (CAPP), Artificial Intelligence (AI), and others. Given the potential benefits of these ideas, it is no wonder that the interest of manufacturing firms, large and small, has been aroused.

When properly planned, managed, and implemented, most of these systems have proved beneficial to their specific areas of application. But too often, out of fear of falling behind in the race towards automation and losing profitability, firms plunge head on into one or more of these techniques without clear goals or long term plans.

The result is usually "islands of automation," isolated areas that benefit from the latest technologies, without communicating or interacting with related activities.

In fact, experience has shown that some firms may independently automate as many as 50 different functional areas [2], often using unique hardware and software for each. Over the past several decades, this practice of implementing specialized activities and functions has been commonly accepted as the way to achieve efficiency. Recently, however, this approach has been questioned. Based on findings of the National Research Council [15], "In case after case, the absolute necessity of integrating each element of the manufacturing process from market analysis through design, fabrication, and sales to follow-on customer service has emerged as the key to achieving the productivity gains required."

The prospect of integrating the various elements of the production cycle requires the development of a global systems approach to manufacturing that has been lacking in the past [9]. While there has been a tremendous growth in the number of narrowly focused "solutions," both hardware and software oriented, there has been little tangible progress toward the solution of the overall manufacturing "problem." One impediment to a more global approach is the traditional specialization of hardware and software vendors. With technology evolving so rapidly and competition intensifying in each area of specialization, it is difficult for vendors to broaden their focus. There are vendors producing engineering packages such as Computer Aided Design and Finite Element Analysis, vendors selling manufacturing hardware and software for such things as Numerically Controlled machinery and robotics, vendors with operations management systems like Manufacturing Resource Planning, vendors marketing financial packages like General Ledger and Accounts Payable/Receivable, and so on. But none of these areas can exist in isolation. Engineering, manufacturing, operations, and finance are all unseparable components

of what has been called [16] the "Product Enterprise System."

With this in mind, it is now time for industry to integrate the individual tools into a single integrated manufacturing system. Because computers are generally recognized as the means through which integration can be achieved, such a concept is commonly referred to as Computer Integrated Manufacturing, or CIM [22], one of the latest buzzwords now circulating in industry.

But CIM means different things to different people. To many, it is the linking of CAD and CAM; to others, it means a well tuned, closed-loop MRP II system. These attitudes are, again, largely a product of specialization; the scope of CIM is much larger. The goal of CIM is to streamline the flow of information throughout a manufacturing business as much as it is to streamline the manufacturing processes themselves. In fact, the entire manufacturing cycle can be viewed as a series of data processing functions [10], involving creating, sorting, analyzing, transmitting, modifying, and storing data. The information involved in manufacturing is quite diverse, including geometrical data, tolerances, bills of material, costs, schedules, inventory levels, and much more.

CIM has been referred to as a system, a project, a product, a philosophy, a concept, and a program [2]. In the absence of a widely accepted definition, it can perhaps be viewed as any or all of these. It should combine hardware, software, and when appropriate, manual procedures to economically and efficiently provide disciplined interaction between every business activity, while remaining flexible enough to respond to changing business and market conditions.

In this work, a general functional model of CIM is presented, aimed at satisfying the above goals. An implementation strategy for integrating the various components of CIM in a modular fashion is described, and a starting point, the integration of Manufacturing Resource Planning (MRP II) and Computer Aided Deisgn (CAD),

is proposed. It is the design of the integrated MRP II/CAD system that is the focus of the work. The areas common to MRP II and CAD, part specification, part structure, and engineering change control, are then identified, and the flow of related information between the two systems is detailed. Finally, the concept of database interoperability as a means for achieving a model of such an integrated system is presented. The integrated MRP II/CAD model is then described and demonstrated using an Update Dependency Language, a tool for achieving interoperability of systems.

Chapter 2

A Proposed Model of CIM Architecture

Given the goal of CIM, to streamline the flow of information and the processing of that information throughout the production cycle, how can such a system be configured? Using the current technologies available, a suggested model of CIM is shown in Figure 2.1. Manufacturing Resource Planning (MRP II), which plans, schedules, and monitors production activities as well as maintaining financial accounts, serves as the central coordinator of CIM. Computer Aided Design (CAD) handles the task of aiding in the creation, updating, and transfering of design information to the rest of the system. Computer Aided Process Planning (CAPP) translates design data information into manufacturing data. And finally, Computer Aided Manufacturing (CAM) converts the product data into physical parts.

Supporting the four major functional areas of CIM are several important, task specific and proven systems, such as Artificial Intelligence (AI), Group Technology (GT), Automated Materials Handling (AMH), and robotics. The role of each of the major systems is described more fully in the following sections.

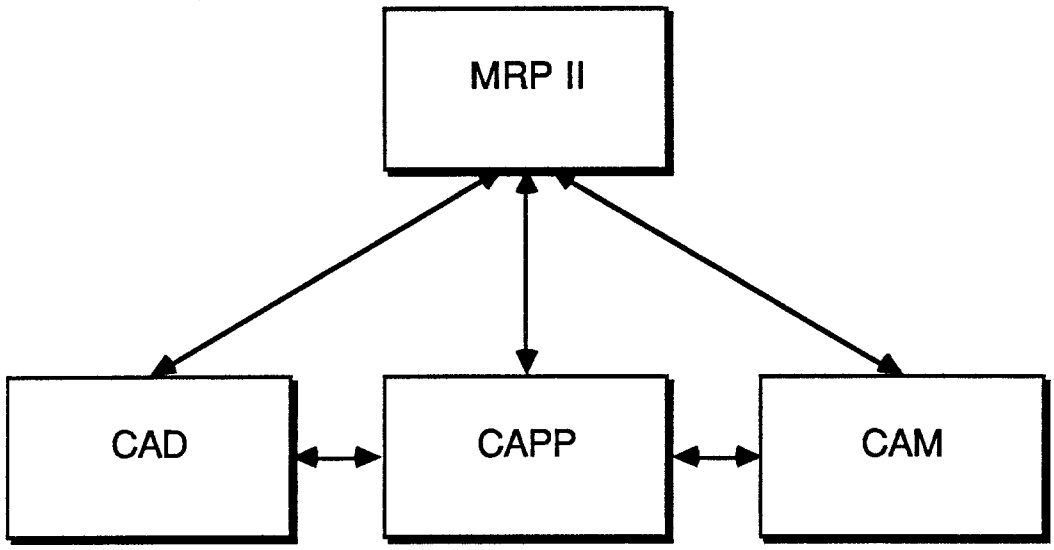


Figure 2.1: Potential CIM Architecture.

2.1 Manufacturing Resource Planning (MRP II)

From its beginnings in the crude inventory tracking systems in the late 1950's and early 1960's, MRP II has become a highly sophisticated closed-loop information system, one that is highly suitable to becoming the information center, or "hub" [8] of CIM. The typical MRP II Architecture is shown in Figure 2.2.

Positioned at the highest level of MRP II is the Master Production Schedule (MPS) module. Using independent demand, in the form of customer orders, and sales forecasts as determined by the marketing department, it is MPS which establishes and monitors the production goals of the organization. By tracking the accuracy of forecasts, MPS can adjust its plans to better suit business conditions. Forecasts of slow selling products can be decreased to free resources for those that are selling well. Production may also be increased to cover unanticipated orders.

At the MPS level, "rough-cut" capacity is explored. Many potential bottlenecks are therefore identified before they occur, by comparing the availability of the most

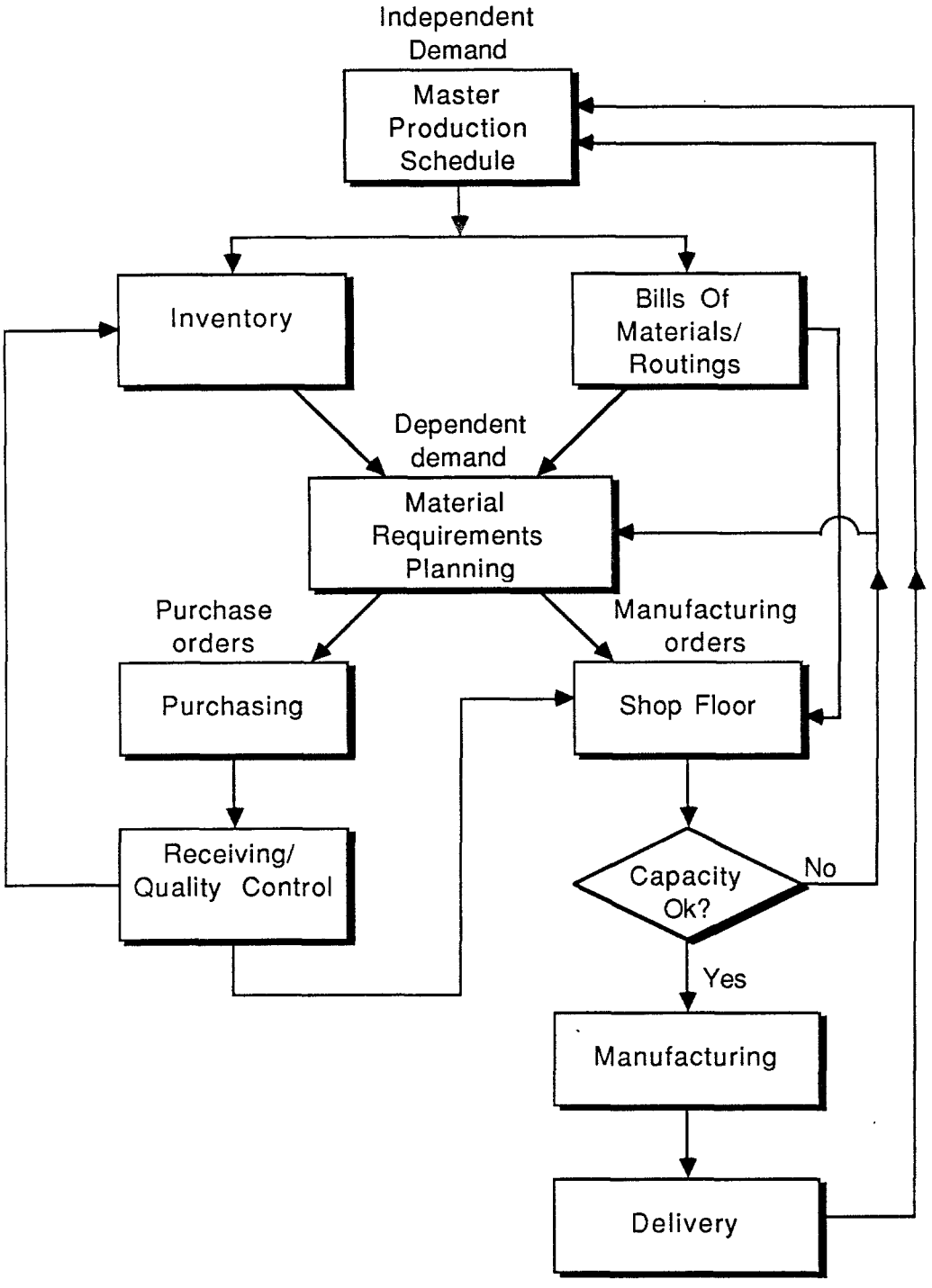


Figure 2.2: Typical MRP II Architecture.

critical resources with the approximate demand. Hence changes can be made, either by reducing the demand for the critical resource(s) or by somehow increasing the availibility of the resource(s).

Once established, the Master Production Schedule is executed by the other modules of MRP II. Materials Requirements Planning (MRP) translates the gross requirements for end products from the Master Production Schedule into time phased net requirements for individual parts. This is done with the help of information contained in the Bills of Material/Routings and Inventory modules. The Bills of Material/Routings module contains basic information about each part (part master data), its product structure (bills of material), and the processes required to produce it (Routings). Along with the inventory and on-order quantities available from the inventory module, MRP uses these data to produce a schedule of manufacturing and purchasing activities necessary to meet the Master Production Schedule.

The primary result of an MRP run is a series of suggested manufacturing and purchase orders. The purchase orders are then handled by the purchasing module, where orders are firmed, released, and tracked. Upon receipt, the Receiving/Quality Control module tracks the items through inspection and into stock, recording inventory transactions in the inventory module.

Manufacturing orders are monitored by the Shop Floor Control module, which produces a rough shop floor schedule using routing information. At this point, capacity is also examined, with greater detail than the rough-cut check at the Master Production Schedule level, and over- and under-utilization of resources is identified.

Orders are tracked as they move from work center to work center, and finally to delivery of the final product. Actual shop floor data (e.g., time and labor booked), for each order can be maintained for comparison to standard or expected values.

During the execution of the MRP generated purchase and manufacturing orders,

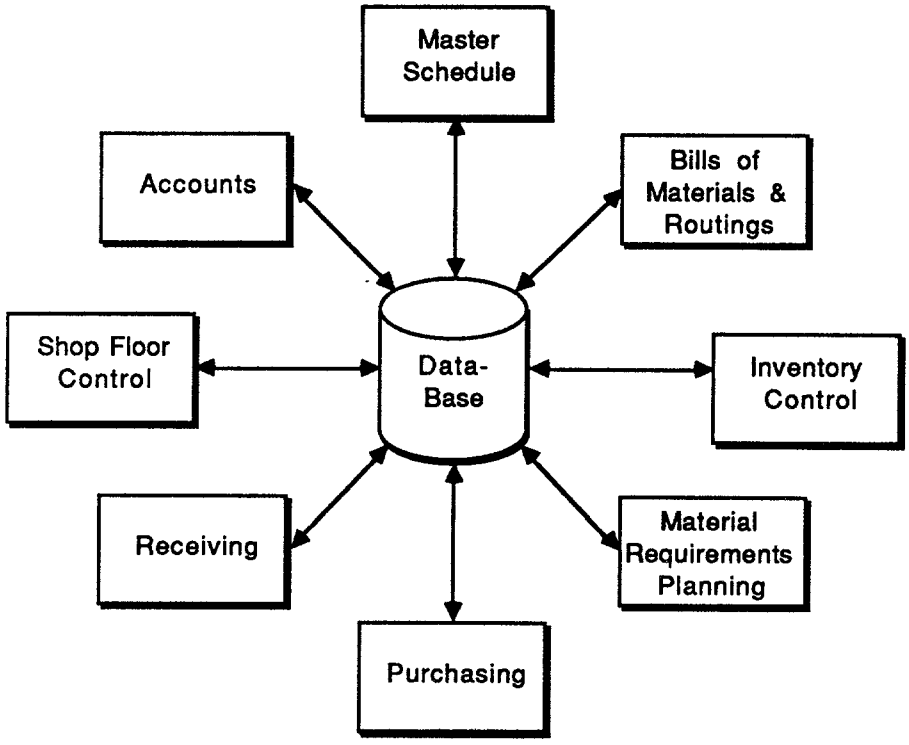


Figure 2.3: MRP II from a Software View.

MRP monitors the completion of individual activities, and alerts the users to late orders, insufficient quantities, and other problems that may delay the completion of the Master Schedule orders.

Not shown in Figure 2.2 but important to the complete operation of MRP II are the accounting functions. Using inventory information and product cost data from the Bill of Material/Routings module (part master data), the value of current inventory can be determined. Using cost data, product structure, and routing information, the manufactured cost of any item can be calculated. Accounts Payable is also coordinated with Purchasing and Receiving, and Payroll is used to record labor costs with data from Shop Floor Control.

Figure 2.3 depicts MRP II from a software standpoint. A central database stores the data required by each of the surrounding functional modules; it is the manage-

ment of this data that makes MRP II such a powerful system. All of the various modules have access to the same data, retrieving and updating it as necessary. The data maintained in the MRP II database includes static part and part structure information, part routings, inventory quantities and locations, production goals, activity schedules, manufacturing and purchase orders, vendor information, shop floor capabilities and schedules, accounting figures, and more.

As described in the preceeding paragraphs, these data allow MRP II to coordinate and monitor all the major functional areas of a manufacturing business. Perhaps this is why some people consider MRP II to be Computer Integrated Manufacturing in itself. But, although MRP II may be able to coordinate and monitor all the major functional areas, it cannot adequately control all of them. The weaknesses of MRP II are most apparent in the areas of design, process planning, and manufacturing. MRP II does not provide any of the design functions of a CAD system, but can only record the static (non-geometric) design data of a part. The same applies to process planning; MRP II has no facilities for creating process plans, only for recording them. In the area of manufacturing, MRP II does generate a schedule of shop floor activities, but it cannot control or monitor individual machines or download Numerically Controlled machining data.

Because there are already systems available to provide the functions MRP II does not, the logical approach seems to be to bring together MRP II and these systems to form CIM. And because the other systems are either providing data to or using data from MRP II and each other, MRP II plays the role of operations coordinator, or "hub" of the combined CIM system.

2.2 Computer Aided Design

Computer Aided Design (CAD) systems were developed to provide users with computer aided tools for designing, drafting, and analyzing parts. The primary goals of CAD are to reduce design time, increase flexibility and adaptability, and to assist the designers in performing tedious design calculations.

CAD systems have developed greatly since their introduction in the 1960's, when they functioned primarily as electronic drawing boards. With CAD (more appropriately called Computer Aided Drafting at that stage), drawings requiring minor, or even major changes could be easily modified instead of being completely redrawn. Beyond improving drafting efficiency, however, CAD did little to streamline the rest of the design function.

Today's CAD systems are much more capable than those early predecessors, though a suprisingly large number of CAD users continue to use them primarily for drafting only [10]. In addition to the traditional drawing capabilities, many of the current systems include design features such as automatic or semi-automatic mesh generation for finite element analysis, clearance/interference studies, solid modeling, and management of part information (i.e., part specifications and part lists).

The role of CAD in CIM is to serve as the center for design information for the entire organization. It is with the help of CAD that design ideas are translated into product data which, in turn, drives the remainder of the manufacturing cycle. The data can be divided into two categories: part specification/product structure data and geometric/design data. The part specification/product structure data represent much of the same information contained in the Bill of Materials/Routings module of MRP II; the linkage of CAD and MRP II through this module is shown in Figure 2.4. This interface allows CAD to control the static part information used

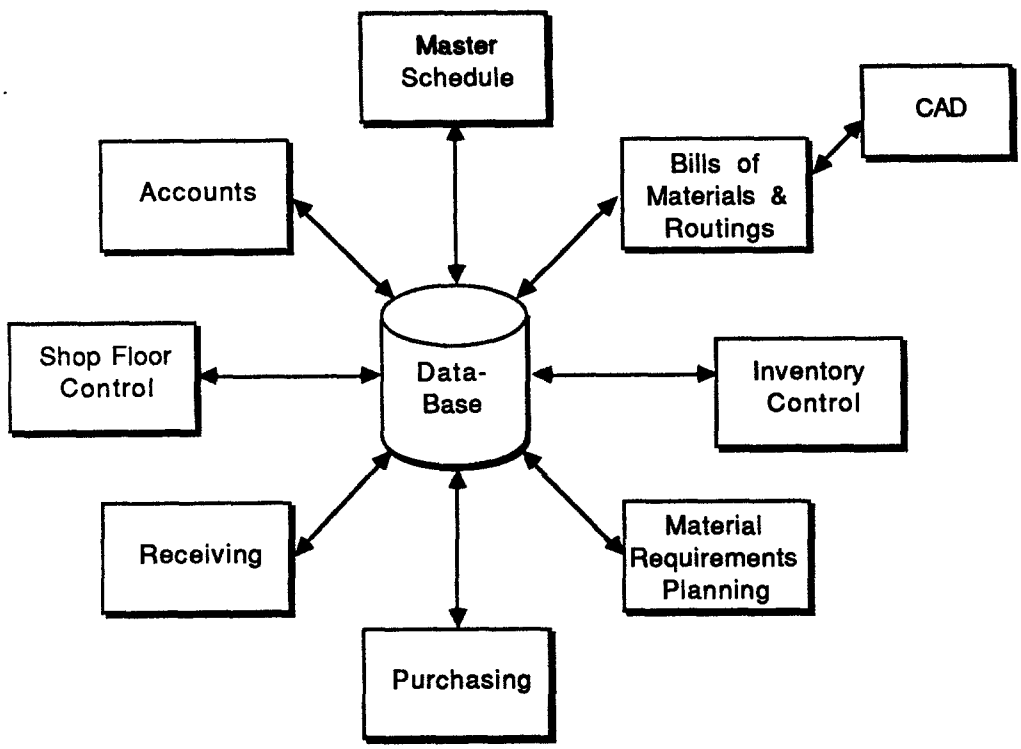


Figure 2.4: MRP II Linked to CAD.

by MRP II.

The geometric/design data maintained by CAD is to be interfaced with Computer Aided Process Planning, described in the following section.

2.3 Computer Aided Process Planning

Computer Aided Process Planning (CAPP) is a fairly recent tool that is used to either automatically or semi-automatically develop process plans for a part from the geometric information provided by CAD. More specifically, the purposes of CAPP are to [20]:

- Select tools and processes
- Determine process sequencing
- Determine cutting conditions and times
- Itentify non-machining elements and times
- Select jigs and fixtures

CAPP systems are generally divided into two classes: variant and generative [14]. Variant CAPP systems develop process plans for new parts with the help of Group Technology (GT) [3]. Parts are classified into families with common geometries, machining requirements, and so on. To create a process plan for a new part, a part similar in some manner to the part under consideration is identified, and the process plans for the old part are modified as necessary for the new part to account for the differences between them. Generative CAPP systems are designed to create process plans for a new part from a "clean sheet of paper", often using a rule based expert system to make decisions. The rules, based on the experience of

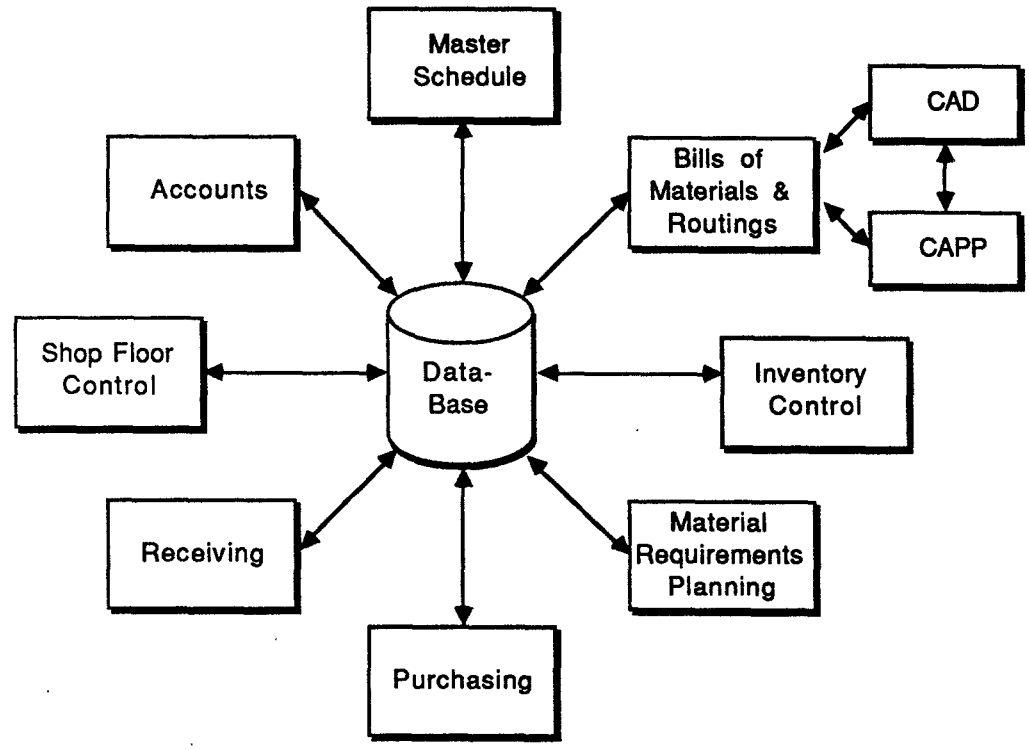


Figure 2.5: CAPP Added to the MRP II/CAD System.

human process planners, specify the details of the process plans given the geometry, features, surface finish, tolerances, and material specification of the part.

The suggested role of CAPP in CIM is shown in Figure 2.5. CAPP interfaces with CAD through the geometric part information generated in CAD. And once the process plans have been generated by CAPP, the information is shared with the Bills of Material/Routings module of MRP II so that shop floor activities can be scheduled. The third interface of CAPP is with Computer Aided Manufacturing, which is the subject of the next section.

2.4 Computer Aided Manufacturing

Computer Aided Manufaacturing (CAM) systems of one form or another have been in use since the late 1940's. CAM is the use of computers to automate specific

manufacturing tasks with the intention of decreasing manufacturing time and cost while increasing consistency and quality of parts. Early CAM systems consisted of numerically controlled (NC) machine tools which read paper tapes with punched holes to produce the required tool movements in much the same way a player piano works.

Today, the concept of Computer Aided Manufacturing is much grander. Instead of simply playing back pre-programmed movements, CAM should be capable of interacting intelligently with its surroundings [1]. Given the needed production quantities and process plans for each part, CAM should handle the detailed scheduling, assigning tasks to specific work centers, downloading the specific manufacturing instructions, and monitoring performance.

In support of these activities, work centers would be set up as Flexible Manufacturing Cells (FMC), consisting of basic tools and capable of producing a wide range of parts simply by downloading the appropriate process plans. Automated Material Handling (AMH) Systems would be used to facilitate the transfer of materials between work centers.

Figure 2.6 shows the addition of CAM to the CIM system. From CAPP, CAM receives the specific process plans for each part. The other CAM interface, with Shop Floor Control, includes the communication of work orders and rough scheduling to CAM, with an accompanying feedback to Shop Floor Control to allow MRP II to monitor progress and report delays or other problems.

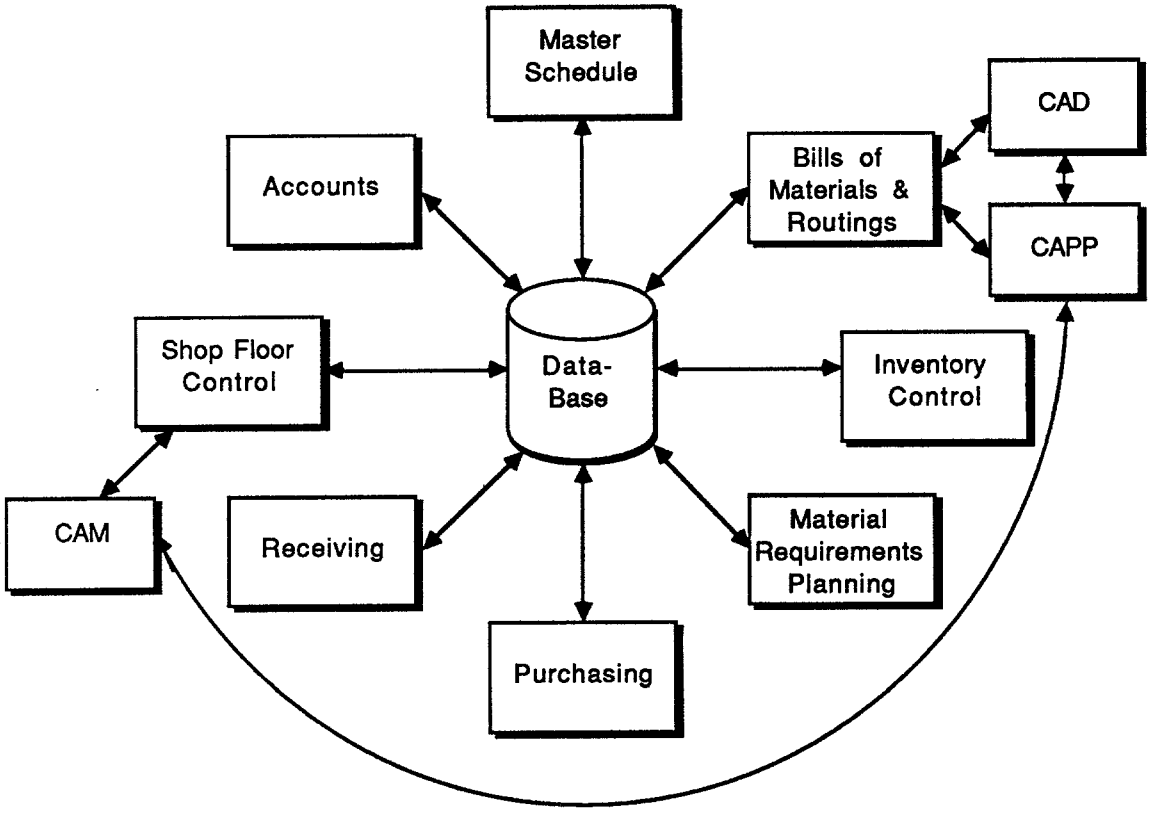


Figure 2.6: CAM Added to the MRP II/CAD/CAPP System.

Chapter 3

A Proposed CIM Design Strategy

Beyond the general concept of CIM, to automate the flow of information throughout a manufacturing organization, there remains the question of how such a system can be designed and implemented. While the model presented in Chapter 2 addressed the basic functional aspects of CIM and described the technologies that could provide the necessary features, the details of the integrated system were not considered.

To develop the detailed functional aspects of CIM, one must understand the flow of information among the various production functions. Perhaps the best way of analyzing this information flow to begin with a systems representation of manufacturing, as suggested in Figure 3.1. The input to the system is a particular consumer need, and the output is a particular product intended to fill the need.

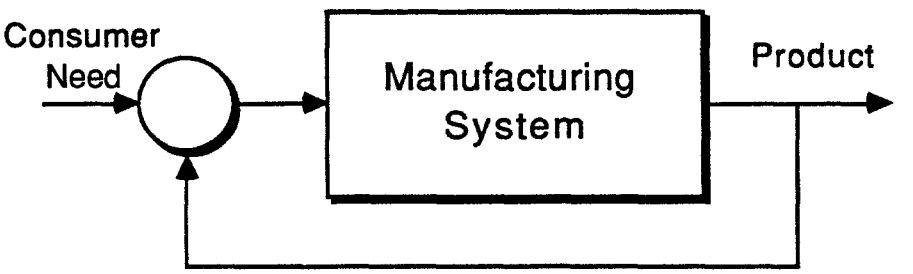


Figure 3.1: A Systems Representation of Manufacturing.

The "system" between includes all the components of a manufacturing business necessary to process the need and supply the end product: marketing, who inputs the need from the market; design, who creates a product design to satisfy the need; manufacturing, who determines how to produce the product and how much it will cost; suppliers, who provide the raw materials for producing the product; accounts, who pay the suppliers, designers, marketers, and so forth; and management, who oversees all activities, making decisions such as whether or not to produce the product, and so on.

The feedback in this system allows the comparison of the final product to the original need responsible for its creation. If the two do not match well, the system can "process" the difference between the two, with the result being modifications to the product definition and/or volume to improve its performance with respect to the original need.

While such an example is useful for demonstrating the system concept, it is of little value for representing the flow of information through an organization, since the entire organization is shown as being within the system. However, as with most systems, it is possible to decompose the representation of Figure 3.1 into smaller units, or sub-systems, each representing a system with its own inputs and outputs. If decomposed to a proper level of detail, the inputs and outputs of the various sub-systems can be linked to create an information flow diagram for CIM.

From the system information flow diagrams for the various components of CIM, formalized procedures for implementing the transfer of information are necessary. These will serve as the basis for the functional design of the integrated system. Each type of information should be placed under the control of a primary system, which means that, in general, only one system will be capable of freely modifying a piece of information, though any other systems requiring the data would have access to

it.

Information flow diagrams created in this way are not unique, however. Different organizations will have somewhat different informational needs, and different approaches to solving these needs. Therefore, the idea of creating a turn-key CIM system is not practical. The implementation of CIM will have to occur on an individual basis, allowing an organization to choose those technologies and tools most appropriate for solving its particular problems.

Research in the area of CIM should therefore be centered in two areas:

- Improvement of the individual manufacturing tools and techniques
- Development of tools for the integration of CIM components

Though there are some who believe that implementing CIM is only a matter of applying technology that is currently available, this attitude is short-sighted. Changes and improvements are required in virtually all aspects of manufacturing, including product design, material handling, processing, assembly, and management [1]. Particularly weak areas requiring research include machine tool technology, programming languages, hardware and software architectures, robotics, and geometric modeling.

The development of tools for integrating CIM components is equally important. It is crucial that these tools be generic, and applicable to virtually any particular systems being integrated. Given the functional details of a proposed integrated system, the tools should provide the means for accomplishing the necessary operational capabilities.

3.1 Management of CIM Data

Perhaps the most complex aspect of developing generic CIM integration tools is the necessity of managing large amounts of data generated in different systems with different formats. Some of the major data management concerns are [18]:

- **Size.** The combined data from several large computer systems may easily run into hundreds of millions of pieces of information, even for a medium size manufacturer, requiring large amounts of storage and increasing access time.

- **Heterogeneity of data and users.** Throughout a manufacturing organization, there are many different types of data, including text, numerical, and graphical information, which must be handled by an integrated system. To complicate matters, different users requiring similar information often require different data formats and arrangements to meet their special needs. Thus an integrated system must provide a number of user interfaces to cater to the various user requirements.

- **Update difficulties.** One of the major concerns of data managment in an integrated system is the problem of updating data within system and data integrity constraints. Since a given piece of information may be modified by users of several systems, means must be provided not only for ensuring that all users will have immediate access to the new information, but also for evaluating the impact of changes on other information stored throughout the system. Thus the data management system must be aware of the detailed data relationships such that either changes to related information can be made automatically (e.g. changing a production schedule when an order due date is changed) or messages sent to appropriate personnel for further action. In

such cases, in which the integrity of one or more systems may be temporarily violated due to incomplete or inconsistent data, a freeze on related activities may be necessary until the conflicts are resolved.

- **Performance requirements.** In addition to satisfying the above technological problems, the database must satisfy the users' needs for performance. Data access time must be reasonable, even when the data is distributed over several geographic locations. The system must be able to handle the special loads induced by graphical information as well as heavy numerical calculations required for engineering analysis. For an extensive integrated system, these requirements will necessitate large-scale computing power, perhaps in one large system, or more likely, a network of smaller machines. When the organization's needs change somewhat, the database should provide a means for adapting to the change without extensive modification.

- **Security.** Finally, the database must provide a means of protecting data. Because the whole organization will be using the same data, it is imperitive that the data be secure. A detailed authorization structure, perhaps with several levels of accessibility, should be maintained within the system to provide the users with only the appropriate access to the information they need to perform their work (read only, or read/modify).

There are two primary schools of thought concerning the management of CIM data [18]. The first proposes the use of a single database to handle all CIM data, while the second suggests the use of multiple databases, with an external "interoperability" system to manage the transfer and updating of data between the systems as necessary.

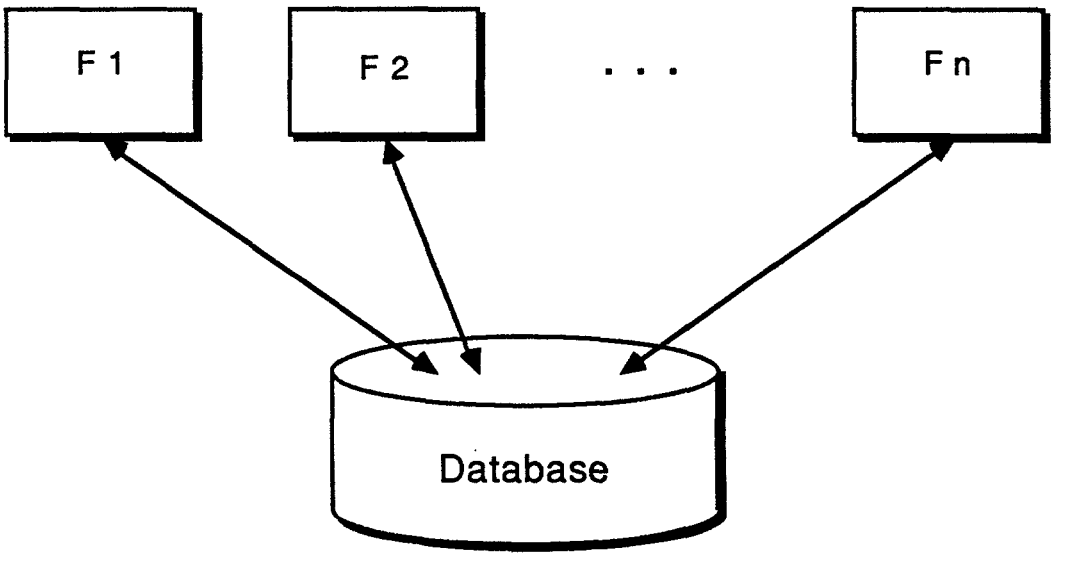


Figure 3.2: The Single Database Concept of CIM Data Management.

The concept of a single database surrounded by a number of application systems is depicted in Figure 3.2. In this configuration, each piece of information is maintained only once in the system, eliminating the possibility of inconsistent data for the same piece of information existing in different applications. The multiple database concept is shown in Figure 3.3. The external interoperability system is positioned between the individual applications and their databases. The interoperability system monitors the application functions, and, when an event occurs in one that affects key data, executes a preprogrammed response, involving related applications, in order to maintain consistency between the applications involved. Because copies of data shared by different applications are maintained in each of these application, it is crucial that the interoperability operations be complete and accurate, or inconsistent data will result.

The idea of a single database has some definite advantages over the muli-database alternative. First, the single storage of each piece of information eliminates the redundancy and consistency problems inherent in separate databases. And sec-

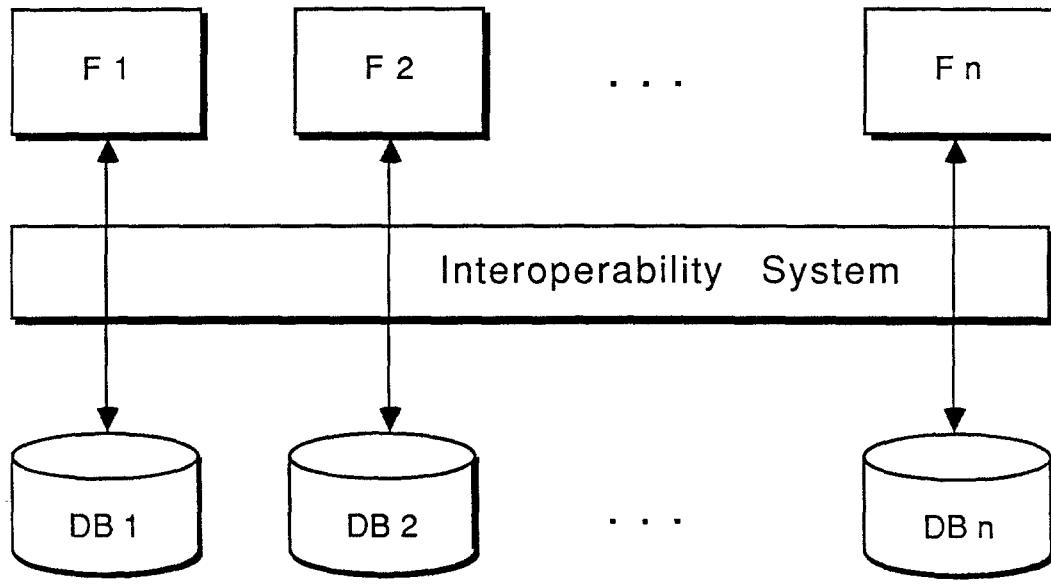


Figure 3.3: The Multiple Database Concept of CIM Data Management.

ond, a single database avoids the communication overhead necessary to transfer information between different databases.

In terms of response time, however, it is not clear which alternative would perform better. With a single database, the response time for all functions would be increased as compared to their performance as isolated systems, due to the size of the combined database. With separate databases, the response time of functions that affect only their local database would be low, whereas the response time of functions that affect other databases as well could be very significant, depending on the amount of traffic (number of inter-database operation calls) needed to maintain consistency.

A major limitation of the single database approach, however, is that it requires a "ground-up" approach to CIM. The databases of existing systems would generally have to be rebuilt in the process of creating a single database. Such extensive modifications are expensive, and would limit, if not eliminate, the support provided

by the original system vendors. Further, the design of a single database requires the consideration of all the data it will contain; future additions or modifications required by the addition of new application may be very difficult.

In constrast, the multiple databse approach is suitable for the linking of existing systems, protecting committed investments in software and hardware, and even more importantly, in user training and familiarity. Further, using multiple databases allows systems to be integrated one by one, by enhancing the interoperability system to include links to the additional systems. With this approach, it is possible to begin implementing CIM with a small set of integrated systems, and to add on additional modules as the implementation progresses. Being such a large undertaking, a modular approach such as this is perhaps the only practical way of developing CIM, at least in the near future.

For these reasons, it is proposed that the multiple database approach be utilized in implementing CIM, and further, that implementation be performed in stages. In the following section, a starting point for this process is proposed: the integration of Manufacturing Resource Planning and Computer Aided Design.

3.2 A Starting Point: MRP II/CAD Integration

The first systems to be integrated into the CIM framework should have a well defined interaction and should be functionally mature in the overlapping areas to the extent that integration is possible. Of the four major systems described in Chapter 2, MRP II, CAD, CAPP, and CAM, this criterion is best satisfied by MRP II and CAD. While the interactions of CAPP and CAM with the other CIM areas are conceptually understood, these technologies are not yet mature enough to handle their CIM roles. Significant improvements in each of these are required before integration into CIM may be considered.

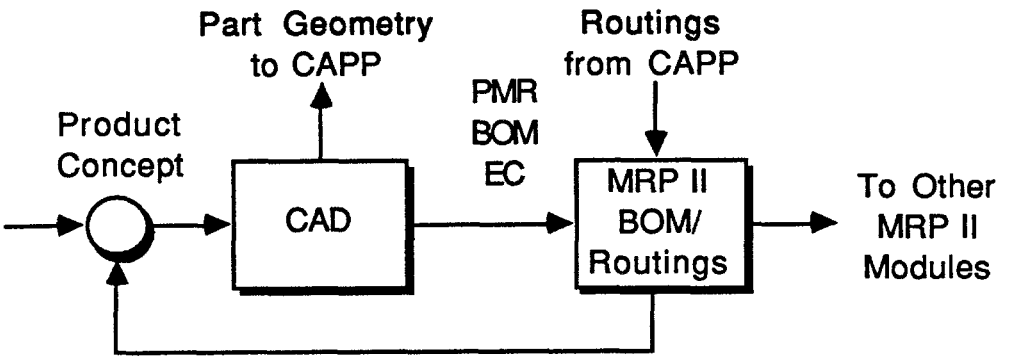


Figure 3.4: System Representation of MRP II/CAD Information Flow.

The information shared by MRP II and CAD is static part and product structure information. Since part information management is so fundamental to MRP II and CAD, these two systems represent an ideal starting point for integration, allowing attention to be focused on the interoperability functions rather than on the details of how each system can use or generate the data. The latter issues have already been addressed by researchers working in these particular areas.

The integration of MRP II and CAD offers the addition of the design area of manufacturing to the central hub of CIM, making design data directly accessible to MRP II users. Figure 3.4 depicts a system representation of the MRP II/CAD information flow. In addition to the "forward loop" transfer of static part information from CAD to MRP II, there is a feedback loop from MRP II back to CAD. The design of a part is an iterative process, and often the experiences of MRP II users in manufacturing or purchasing a part enable them to propose design changes that would improve manufacturability or reliability.

The area of interaction between MRP II and CAD, part information management, is a logical starting point for CIM. Until recently however, there has been little attention focused on it. Perhaps the reason for this is once again specialization of software and hardware vendors, with MRP II being operations management

oriented while CAD is engineering oriented. Complicating this problem is the fact that until recently, most MRP II systems were designed for mainframe computers, while most CAD systems were oriented torwards minicomputers.

Still another problem is the reluctance of CAD advocates to getting involved with MRP II based on its fairly low success rate of about 25 percent [6]. Though MRP II is a sophisticated and powerful system capable of providing significant benefits, inadequate implementation strategy, lack of top management support, or insufficient user education and training can all contribute to its failure [12], particularly in organizations not accustomed to formalized systems.

Only an estimated 2 percent of MRP II users [21] have actually achieved the closed-loop information system made possible by the system. The majority of MRP II users, perhaps 55 percent, are using the system primarily for open-loop Materials Requirements Planning (MRP).

It is interesting to note, as mentioned earlier, that the majority of CAD installations also fail to take full advantage of the current technology [10]. However, such cases are generally not considered as failures.

With the proper committment and a thorough systems analysis of the informational needs and flow through the organization, it is likely that integrated system failures would be minimal. The transition to a formal system is always difficult, but it is essential for CIM. With a modular implementation of CIM, this transition can be planned and achieved gradually.

3.3 The Fundamental Similarities of MRP II and CAD

The data elements common to MRP II and CAD include part specification and bills of material [5]. Though these serve somewhat different functions in each, much of

the same data is used by both. Part specification involves the documentation of parts, both purchased and manufactured. In MRP II, the collection of information documenting each part is called a part master record (PMR). It contains information used in the procurement, manufacture, and assembly of components, such as:

- Part Number
- Revision Number
- Part Description
- Drawing Number
- Make or Buy Code
- Unit of Measure
- Vendor Information (for purchased parts)
- Leadtimes
- Standard Cost
- Required Quality Code

Part master record information is maintained in the Bill of Material/Routings module of MRP II. In addition, many of these same information fields are also maintained in the CAD system as a means of documenting and cataloging design drawings.

The bill of material (BOM) for a product is a family tree structure identifying the component parts, their quantities, and their relationships. Parts of assemblies on a given level in a BOM are said to be the "parents" of the "children" on the next lower level. A well structured BOM models the logical sequence of manufacturing

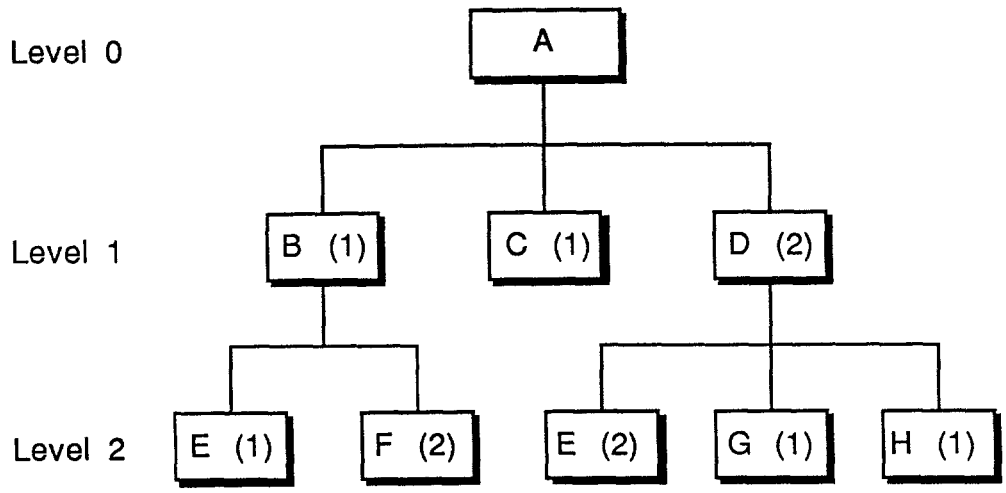


Figure 3.5: A Typical Bill of Material.

operations for the assembly it represents. An example of a bill of material is shown in Figure 3.5.

To MRP II, the BOM serves as the guide to production activities and material purchases. Within the Material Requirements Planning (MRP) module of MRP II, the requirements for end product production (as determined by the Master Schedule of MRP II) are carried through the various levels of the BOM in order to determine the quantity of each assembly and component, and the quantity of raw material, and the timing of these requirements (using the leadtime information in the PMR file). Inventory records are then checked for current stock and pending orders of items to determine the net requirements of each. For purchased items, purchase orders are generated. For manufactured items, work orders are created.

Within CAD, the BOM, commonly referred to as a parts list, is used for documentation purposes, representing the explosion of an assembly into its components and subassemblies. It is the CAD BOM that introduces the product structure to the rest of the organization. Ideally, the CAD and MRP II BOMs should be identical. Unfortunately, however, MRP II users often find it necessary to modify the

CAD BOM to better represent the manufacturing sequence of activities of an item. For this reason, organizations often have two versions of bills of materials: The Engineering BOM used by designers; and the Manufacturing BOM, used by manufacturing. As part of the transition to formal manufacturing systems, it is crucial that organizations maintain only one type of BOM, the Manufacturing BOM. To do so, designers must be aware of the general manufacturing activities, procedures, and facilities available for producing a part, so that the Manufacturing BOM may be created during the design process.

By the time a part is released from a CAD system, it is given a number of attributes, such as:

- Part Number
- Revision Level
- Drawing Number
- Description
- Unit of Measure
- Tolerances
- Surface Finish

Much of this information forms the basis of the part master record. Typically, this information from CAD is printed out, transferred to MRP II users, and manually entered into MRP II. From this "skeletal" information, the manufacturing department determines whether the part will be manufactured or purchased, and what its approximate cost will be. If it is to be purchased, vendors are sought and

price and time quotes requested. If it is to be manufactured, master and alternative routings are explored, based on existing in-house manufacturing capabilities. The information resulting from these activities is then added to complete the PMR information.

The transfer of BOM information has traditionally followed a similar procedure [4]. BOMs are generally constructed by the designer, who traditionally reviews each detail and assembly drawing to generate a list of needed components and materials, including consumable items such as welding rods, adhesives, and protective finishes. This process is extremely time consuming, considering that even a relatively simple product often requires reviewing more than 100 individual drawings. Complex assemblies require more time because the drawings have to be reviewed repeatedly to achieve accurate quantity counts, appropriate dimensional tolerances, and to ensure that no items are overlooked. While some CAD systems available today better facilitate BOM construction, considerable effort is still generally required.

Once created by the designer, a paper copy of the BOM information is typically manually entered into MRP II so that production control and purchasing may begin the activities required to manufacture the new product.

The transcription of PMR and BOM data from CAD to MRP II is an extra step involving an unnecessary delay and increasing the chance of errors. From the very entry of the information into MRP II, there exists the possibility of having data inconsistent from that in CAD. But PMR and BOM data must serve the organization throughtout a part's life [19], from its design (which follows concept and feasibility studies), through production, post production, and termination phases. Over its lifetime, there may be numerous engineering changes to a part requiring modification to either the PMR or the BOM, or both [11]. Keeping both CAD and

MRP II up-to-date with respect to these engineering changes is a major task, as they must be entered and maintained independently on each system.

The obvious solution to this problem is for CAD and MRP II to share PMR and BOM data. In the integrated MRP II/CAD system, common data can be maintained and made available to users of either system, eliminating transcription errors resulting from keying in the same data to both. The common data in both systems can therefore always be up-to-date and consistent. Part specification data from CAD drawings can be used at the time of a drawing's release to establish a part master record for the part, which will be completed by MRP II users. Bills of material from CAD can likewise be transmitted to MRP II. Engineering change control may be simplified, since modifications to the shared data will automatically be propagated between the two systems.

The goal of the MRP II/CAD integrated system is to achieve accurate and timely exchange of information between MRP II and CAD. In the following chapter, the functional design of this system is presented.

Chapter 4

Functional Design of a Model MRP II/CAD Integrated System

The design of the interoperability functions for integrating MRP II and CAD are based on the flow of information between the two systems, as depicted in Figure 3.4. The model to be presented addresses the transfer of part master data and bills of material between the two systems; modifications in the form of engineering changes are likewise considered.

The purpose of this model is to demonstrate the feasibility and practicality of integrating MRP II and CAD by exploring the interoperational capabilities required to do so. However, to remain as general as possible, the model is not based on any particular MRP II or CAD system, but instead relies only on the basic functions and capabilities fundamental to most commercial systems. Further, this model is not a unique solution to MRP II/CAD integration, but only an example of how such an integrated system could operate.

4.1 Model Assumptions and Framework

The model is intended for application in a discrete parts environment, where end products are made to stock; such an environment is well suited for the application of MRP II as well as CAD. It is assummed that in this environment, only one version of

a given part is active at any given time throughout the organization. This contrasts with a make-to-order environment, where planning is done on a contract basis, and individual contracts may require different versions of the same part at the same time [11].

Further, to remain as general as possible, the model is somewhat simplified; the data represented consist of the most basic MRP II and CAD part information. Most full-scale MRP II and CAD packages maintain additional data. It is, of course, possible to extend the model to consider these additional data.

It is also assumed that the integrated system is to be implemented with an initially empty database. Thus the model will maintain consistency of data entered through the system, but has not been designed to control data already present in either CAD or MRP II.

4.1.1 Designation of Authority between MRP II and CAD

The roles of MRP II and CAD in the integrated system are crucial to the operation of the model. CAD, being the center of design activity, is the primary controller of design information. The evaluation of design alternatives, creation of new parts, and the modification of existing parts is performed within CAD, though of course using input from other departments. Marketing, for example, identifies the features desired in a given product through marketing surveys, and analyzes what potential customers may be willing to pay for them, an important design consideration. The manufacturing department frequently requests design changes to improve a part's manufacturability. Unfortunately, too often this occurs only after designs have been finalized, complicating the process. By improving the communication of information between manufacturing and design, the "wall" that has traditionally existed between these two departments can be lowered, increasing the role of manufacturing aspects

in the design process.

MRP II plays an execution role, planning for and monitoring the actual procurement and manufacture of items. With the CAD design data as a starting point, the decision whether to make or buy the part is made, and any changes to the bill of material or part master record made necessary by the decision are performed. For purchased parts, MRP II users seek and record information regarding vendors, prices, terms, and leadtimes. For manufactured parts, routing information, manufacturing leadtimes, and estimated manufacturing costs are recorded. It is this manufacturing and procurement data that is controlled by the MRP II system.

4.1.2 Control of Information Flow in the Model

The flow of information between the two systems is regulated by a series of status codes assigned to each set of data within each system. The status codes are designed to provide for the efficient transfer of information between the systems while accomodating the timing of various design and manufacturing activities. For the CAD system, four status codes are suggested:

- **Working.** The "working" status is assigned to CAD data related to designs that have not yet been finalized. Such parts may still be under development, or may be within the review process.

- **Released.** CAD part data with a "released" status indicates parts that are currentlty active in the organization. Parts are generally released only after some type of formal review procedure.

- **Hold.** Part data in CAD is given a "hold" status when the part is being reviewed for possible revision or replacement and when the current design should not be used during this process. This code would likely be used only

when the review for revision is triggered by safety or performance concerns. Generally, therefore, parts being reviewed as part of their normal "evolution" would not require the "hold" status.

- **Obsolete.** Data related to parts and/or revisions that have either been superseded by other parts and/or revisions or simply eliminated from the list of active parts are given an "obsolete" status, meaning they are inactive.

In contrast, the MRP II system has only two possible statuses for part data:

- **Released.** As in the CAD system, "released" MRP II part data relates to active parts.

- **Hold.** A "hold" status on part data in MRP II indicates one of two situations. The first is a part with an incomplete part master record, which should be completed before its release. This corresponds roughly to the "working" status used in CAD. The second use of the "hold" status is to temporarily suspend the MRP II activities related to a part. This may be due to a review for revision necessitated by safety or performance problems, as mentioned in the description of the CAD "hold" status, or it may be due to manufacturing, supplier; or other problems falling within the responsibility of MRP II users.

The "working" status is not used in MRP II, since by the time a part is released from the CAD system to MRP II, it has been established as an finalized part. Thus, the "tentative" nature of a "working" CAD drawing is not applicable. The "obsolete" status used in CAD is likewise not necessary in MRP II, which monitors "obsolescence" automatically, through the use of effectivity start and end dates for each part and revision.

The detailed design of the model and the use of these status codes to regulate information exchange between the two systems are presented in the following sections.

4.2 Part Master Data

The part master information maintained by the model includes the following fields:

- **Part Number.** In this model as well as in any formal manufacturing system, a part number must uniquely identify a single part throughout the organization; two or more parts cannot share a single part number, and a single part number cannot represent two or more physical parts. The most efficient part numbering system is one utilizing nonsignificant numbers and some form of error checking algorithm [13]. Part numbers are assigned and maintained by CAD users.

- **Revision Level.** During the life cycle of a part, it may undergo many design changes. The different versions are tracked with a revision level. In a make-to-stock environment, typically only one version per part (the latest) is active (i.e., "released" status) at any given time. CAD users are responsible for assigning revision levels.

- **Effectivity Start Date.** This is the date at which a specific version becomes active, replacing older versions, if any, for MRP II planning. Generally, MRP II users determine this date based on the status of activities related to both the old revision if applicable (e.g., current stock and outstanding orders) and the new revision (e.g., leadtimes, tooling requirements, and anticipated orders).

- **Effectivity End Date.** This is the date at which a particular revision is made inactive. If not explicitly entered, it is assumed that the revison is to be kept active until further notice. When a new revision of the same part is made active, the effectivity start date of the new revision is made the effectivity end date of the old revision.

- **Status Code.** This is the status code as described in the previous section, which differs between CAD and MRP II. These are maintained by the system, though users of both systems may change the status subject to the constraints of the model.

- **Drawing Number.** The drawing number is used to identify the specific drawing associated with each version of a part, and is the responsibility of CAD users. Typically the drawing number is the same as the part number; this field is provided in the event that it is not. Such a case would occur when a single drawing has more than one part represented, such as a single drawing depicting a series of similar bolts varying in length only. Each of these bolts should have a different part number but all will share the same drawing number.

- **Drawing File Name.** This field is used to identify the name of the file containing the CAD drawing of the part; it is maintained by CAD users.

- **Drawing Size.** The drawing size is a code to represent the physical size of the drawing. Its primary use is to aid in locating drawings, since different sized drawings may be stored in different locations (e.g., different file cabinets or different rooms). CAD users are responsible for maintaining this information.

- **Description.** The part description field is maintained by CAD users. The description should be concise and as meaningful as possible. The use of generic key words is recommended to facilitate retrieval of part information by description searching.

- **Bill of Material Unit of Measure.** This is the part's unit of measure (i.e., how it is measured) in the bill of materials (BOM). Typical BOM units of measure include each, foot, pound, box, and so on. The quantity per assembly field in the bill of material of any assemblies using the part is based on this unit. This field is maintained by CAD users.

- **Purchasing/Inventory Unit of Measure.** Often a part is purchased and/or inventoried in a unit of measure different from that used in the bill of materials. For example, a type of steel bar may be measured in feet in the bill of materials but may be purchased and stored by the pound. Many MRP II systems have separate units of measure for purchasing, inventory, and the bill of material; in this model only two are used: one for the bill of material and one for purchasing and inventory. MRP II users are responsible for this data.

- **Unit of Measure Conversion Factor.** If the bill of material unit of measure differs from the purchasing/inventory unit of measure, the unit of measure conversion factor relates the two, allowing MRP II to determine the proper purchase order quantity, given the quantity required based on the analysis of the bills of materials. This field is also the responsibility of MRP II users.

- **Source Code.** The source code identifies the orgin of the part, either manufactured (source code M) or purchased (source code P). This is generally determined by MRP II users. Most MRP II systems allow for other part types,

such as family or phantom parts, but these do not represent physical parts and therefore they are not considered in this model.

- **Standard Cost.** The estimated cost of manufacturing or purchasing an item is maintained by MRP II users. For purchased parts, this information comes from vendor quotes. For manufactured items, this is determined by a combination of the cost of the components (i.e., sub-assemblies and raw materials) and the operations (i.e., routings required to make the part).

- **Leadtime.** The leadtime of a manufactured item is the estimated total or "standard" elapsed time required to produce it, assuming all of the items necessary for its manufacture are already available. This includes time required for paperwork and time spent idle on the shop floor as well as the actual times of the routing activities. Similarly, the leadtime of purchased parts is the total standard elapsed time required to order and receive parts from a vendor. MRP II users are responsible for this information. Many MRP II systems break down the leadtime into several time "fences," such as "firmed planned" and "ready for release," which is useful for scheduling pre-order-release activities.

- **Supersedes Part Number.** This field is used when the current part is to replace a previously released part. The part number to be replaced is entered, indicating that it is being made obsolete by the new part. This allows the tracking of part supersession in much the same manner that the revision level field tracks the engineering changes made to specific parts. Being part of the design process, this field is controlled by CAD users.

MRP II	CAD
Part Number	Part Number
Drawing Number	Drawing Number
Drawing Size	Drawing Size
Description	Description
BOM Unit of Measure	BOM Unit of Measure
Purch/Inv Unit of Measure	
UOM Conversion Factor	
Source Code	
Standard Cost	
Leadtime	
Supersedes Part Number	Supersedes Part Number
Superseded by Part Number	Superseded by Part Number

Table 4.1: Part Data Maintained by MRP II and CAD

- **Superseded by Part Number.** For an obsolete part, this field is used if the part was made obsolete because it was replaced by another part; the field contains the number of the part replacing it. This field is likewise under the control of CAD users.

4.2.1 Data Division and Organization

The data described above have been divided between CAD and MRP II based on their relevance to each system. Some fields are maintained by both systems, while others are unique to one or the other. Further, some of the data is maintained on a part-by-part basis, while other data must be maintained separately for each version of a part.

Each system therefore maintains two sets of part data: one to record data common to all versions of a part and one set to record data specific to each revision level. Table 4.1 shows the data maintained in the part data set in each system, while Table 4.2 presents the data contained in the revision data set for each system. Note that for each part number, there will be at least one set of revision data.

MRP II	CAD
Part Number	Part Number
Revision Level	Revision Level
Effectivity Start Date	Effectivity Start Date
Effectivity End Date	Effectivity End Date
MRP II Status Code	CAD Status Code
	Drawing File Name

Table 4.2: Revision Data Maintained by MRP II and CAD

In the following sections, the exchange of information during the creation, deletion, and modification of new parts and revisions is detailed. Status diagrams, showing the states of the data at each point in the operations, are used to pictorially represent the operations.

4.2.2 Adding New Parts Via CAD

In the model of the integrated system, Computer Aided Design controls the creation and modification of design data, introducing new parts as well as originating engineering changes. However, the design of the model allows new parts to be entered into the integrated system from either CAD or MRP II. It is assumed that, in general, all manufactured parts including end products, assemblies, components, and raw materials will enter the system through design activity in CAD. The majority of purchased items will likely be entered into CAD as well, particularly if they require design drawings. The capability of adding new parts into MRP II is provided to handle the case of commonly used items with no drawings. Office supplies is a typical example; many companies enter these as parts into their MRP II systems to facilitate ordering, yet there is obviously no need for design drawings. Another example is standard, or "catalogued" items such as motors, which are not designed or drawn up, but are only represented as a specification sheet. Again the primary

need for information related to these items is within MRP II, although some firms treat the specification sheet as a drawing. In this case, CAD will initiate the part as in the general case.

Consider first the case where a new part is initiated within CAD. The basic sequence of operations for the addition of a new part to the integrated system from CAD is shown in the status diagram in Figure 4.1, where both the new part and the previously released part being superseded by the new part (if applicable) are shown.

While the part is still under development, it is entered into the integrated system. To be entered, it must be assigned the following for the establishment of the part record:

- Part Number
- Description
- BOM Unit of Measure

Further, to establish the first revision record, the system requires:

- Revision Level of first version (usually 0, 1, or A)
- Drawing File Name

The designer may also choose to assign a separate drawing number, if applicable, and/or complete the drawing size or supersedes part number fields, though these are not required. With this information, the system performs a series of consistency checks to ensure that the part number being added does not already exist in either CAD or MRP II. If it does, an error message is generated and the part record is not added. Using the data entered by the designer, a CAD part record is established

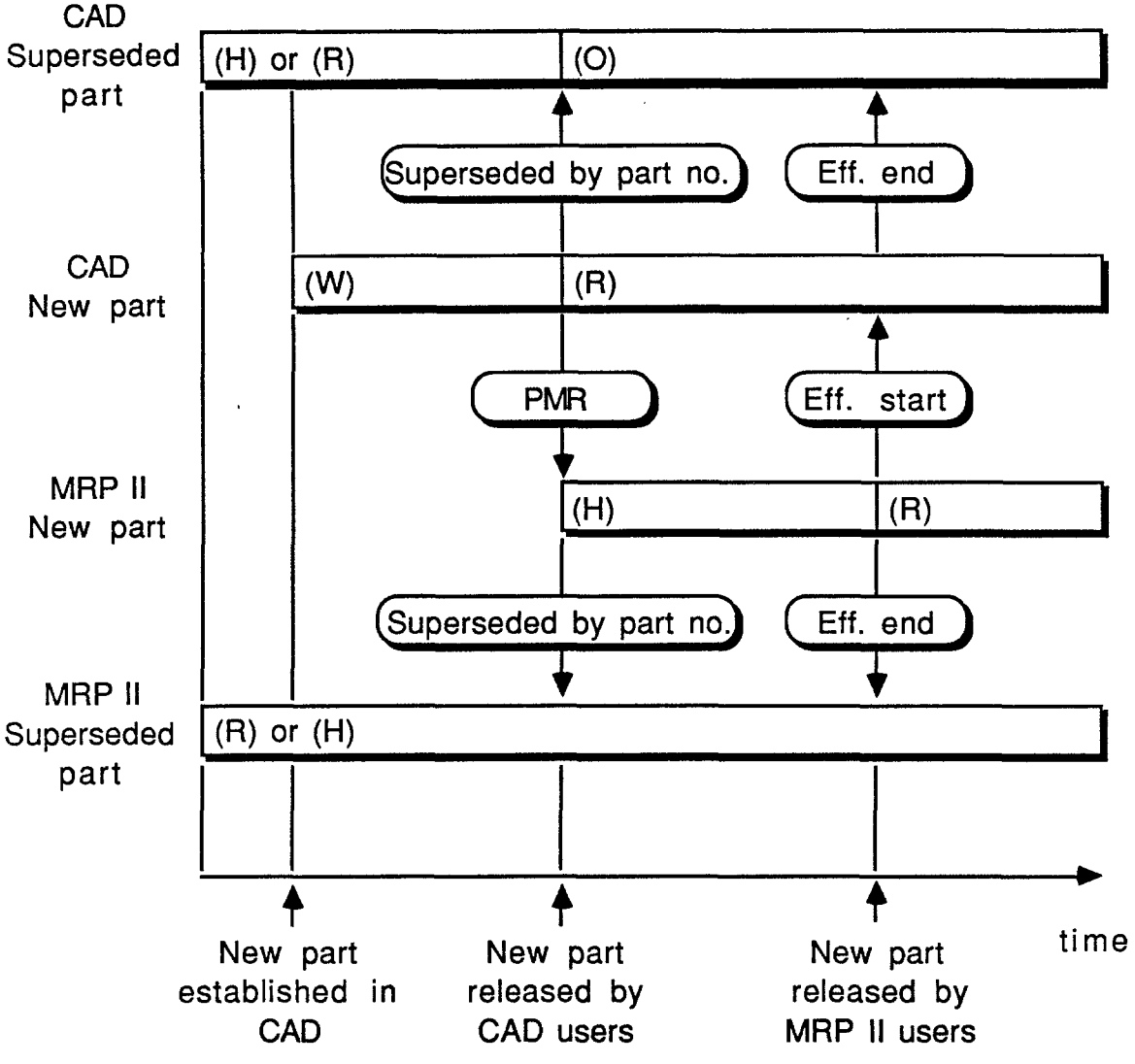


Figure 4.1: Status Diagram for Adding New Parts Via CAD.

for the new part by the system, with any unspecified optional fields given the value "unknown."

Though MRP II is checked by the system at this stage, no data about the new part is created in MRP II. While a part is under development, all information about the part remains local to CAD.

In addition to the part record, a revision record for the part's first revision level is created within CAD. The status of this revision is set to "working" to indicate that the design is still in progress, as shown in Figure 4.1. The name of the drawing file containing the design is entered in the drawing file name field, and the effectivity start and end dates are left as "unknown." As with the part data, no revision information is transferred to MRP II at this point.

When the design of the new part is complete, it undergoes a formal review process according to organizational procedures. When the design is approved, perhaps after further modification, it is released. At this point several actions are initiated. First, a "skeletal" part master record for the new part is established in MRP II, containing the same information as recorded in the CAD part record; those data fields maintained in MRP II but not in CAD (e.g., standard cost, leadtime, source code) are assigned the value "unknown" until supplied by MRP II users.

Second, a revision record is also established in MRP II for the new part, as indicated in Figure 4.1,using the part number and revision level from the CAD revision record; the effectivity start and end dates for the revision are left as unknown. The status of the MRP II revision is set to "hold," since many of the fields required by the MRP II system have been initialized to unknown and must be completed before MRP II can consider the part to be active. A message is generated within MRP II indicating that a new part has been created and that its part master record requires attention.

Third, within CAD, the system checks for a value in the "supersedes part number" field of the new CAD part record. If it finds a valid part number in this field, the part number of the new part is inserted into the "superseded by part number" field of the CAD part record of the part number found. The MRP II part master record of the superseded part is likewise modified to reflect the supersession. The latest revision of the part being superseded, which may have a "released" or "hold" status, is then given an "obsolete" status in CAD, as shown in Figure 4.1.

Finally, if the preceding steps are successfully completed, the status of the new CAD revision is changed from "working" to "released" to complete the CAD release function, as indicated in Figure 4.1.

As noted, however, the data in MRP II related to the newly released part is initially given a status of "hold." After the part has been released by CAD users, it is the responsibility of MRP II users to research the part and supply the data fields necessary for the release of the part within MRP II.

It must be decided whether the part is to be manufactured or purchased. If this choice differs from that originally implied by the CAD bill of material (or lack thereof) for the part, CAD users should be notified so that the required changes can be made. For example, if CAD users plan for a part to be purchased, it will not have a bill of material; if MRP II users later determine that it is most efficient to manufacture the part, a bill of material identifying the necessary components of the item should be created, most likely within CAD (and then automatically transferred to MRP II by the system).

If it is to be manufactured, routings for the part must be developed. If it is to be purchased, potential vendors must be contacted and evaluated. In either case, a standard, or approximate cost should be computed, either from vendor quotes (for purchased parts), or from the routings information, labor rates, and cost of

component parts (for manufactured parts). Also, the leadtime of the part should be estimated for use by Materials Requirements Planning (MRP).

Depending on the value of the part and its criticality to the organization, an inventory policy should be established, considering such aspects as buffer stock. Finally, an effectivity date must be chosen so that MRP can begin planning for the production or ordering of the new part. If the new part does not supersede any other part, the choice of an effectivity start date is based largely on the anticipation of the need for the new part; since MRP will not plan orders for the part until it is needed, the choice of the effectivity start date is generally not critical.

If the part supersedes another however, the selection of an effectivity start date is more significant. As of the effectivity start date, the new part replaces the superseded part in all bills of material requiring it (see section 4.3.4). The effect of this is that MRP (Material Requirements Planning) begins to generate requirements for the new part, while discontinuing the generation of requirements for the superseded part. Therefore, unless significant safety or performance issues are involved, it is desirable to choose an effectivity start date far enough into the future to allow for the consumption of existing stock and pending orders for the part being superseded.

Once these part data are determined and entered into the system, the part and revision records can be released to MRP II. In addition to the data required when the part was first entered into CAD, the model requires the following data to complete the part record:

- Purchasing/Inventory Unit of Measure
- Unit of Measure Conversion Factor
- Source Code
- Leadtime

And to complete the revision record:

- Effectivity Start Date

The standard cost may also be entered into the part record and the effectivity end date may be entered into the revision record, although neither is required.

As a further requirement for the release of a part within MRP II, the system checks the status of the new revision in CAD. If the revision has been placed on "hold" in CAD due to safety or performance problems, it cannot be released; a message to this effect is given to the user and the data is not released.

Assuming the part data is still active in CAD, the status of the MRP II data is changed from "hold" to "released," as depicted in Figure 4.1, and the MRP II part and revision records are updated to reflect the additional information. As the final step in the release process, the effectivity start (and end, if supplied) date of the first version of the new part is transferred back to CAD to update the CAD revision record.

Once the part is released to the MRP II system, users of that system have the ability to place the data on hold, if desired, to temporarily suspend MRP activities involving the part. Such a hold is local to MRP II, involving no interaction with CAD, and is intended for use in the event of manufacturing or purchasing problems, such as broken equipment or a vendor's inability to ship, which might interrupt production activities.

Once part data is placed on hold, it may be rereleased at any time, but only after the system has verified that the data is not on hold in CAD.

4.2.3 Adding New Parts Via MRP II

For purchased parts with no design drawings associated with them, the ability to add parts directly into MRP II is included in the model. Parts may be entered with

either a "released" or "hold" status. The status diagrams for this procedure are shown in Figures 4.2 and 4.3, depending on the desired MRP II status of the new part. Each of these figures depicts both the new part data and the data related to the part being superseded by the new data, if applicable.

Regardless of the desired part status, the user must enter information into all the required part record fields, namely:

- Part Number
- Description
- Bill of Material Unit of Measure
- Purchasing/Inventory Unit of Measure
- Unit of Measure Conversion Factor
- Source Code
- Leadtime

The user may also specify in the part record the standard cost and/or the supersedes part number fields.

The information required for the establishment of a revision record depends on the desired status of the new part. If the part is being entered with a "released" status, as in Figure 4.2, the system requires:

- Revision Level
- Effectivity Start Date

The effectivity end date may also be specified.

If the part is being entered with a "hold" status, as in Figure 4.3, the only value required is:

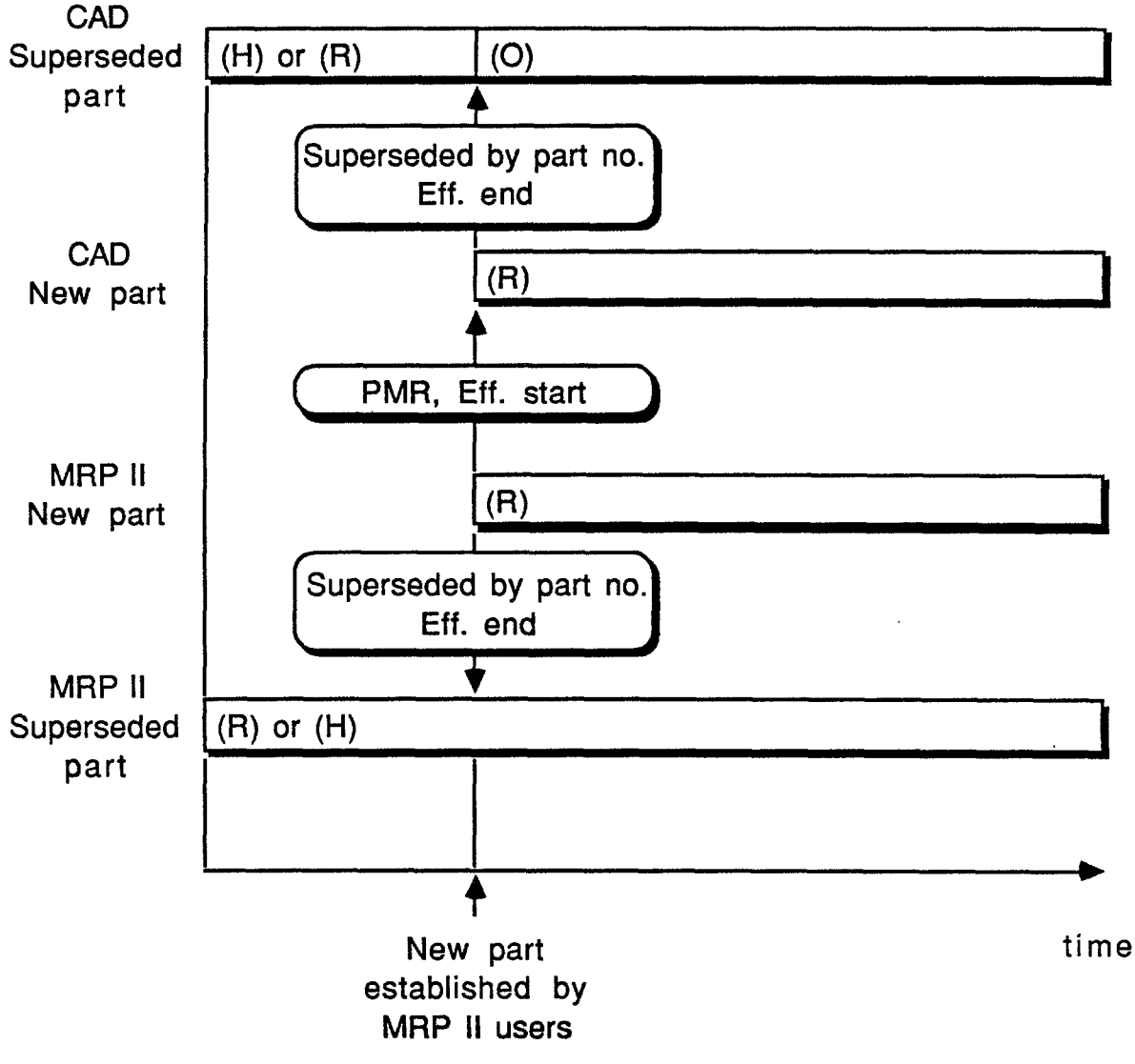


Figure 4.2: Status Diagram for Adding New Parts with Released Status Via MRP II.

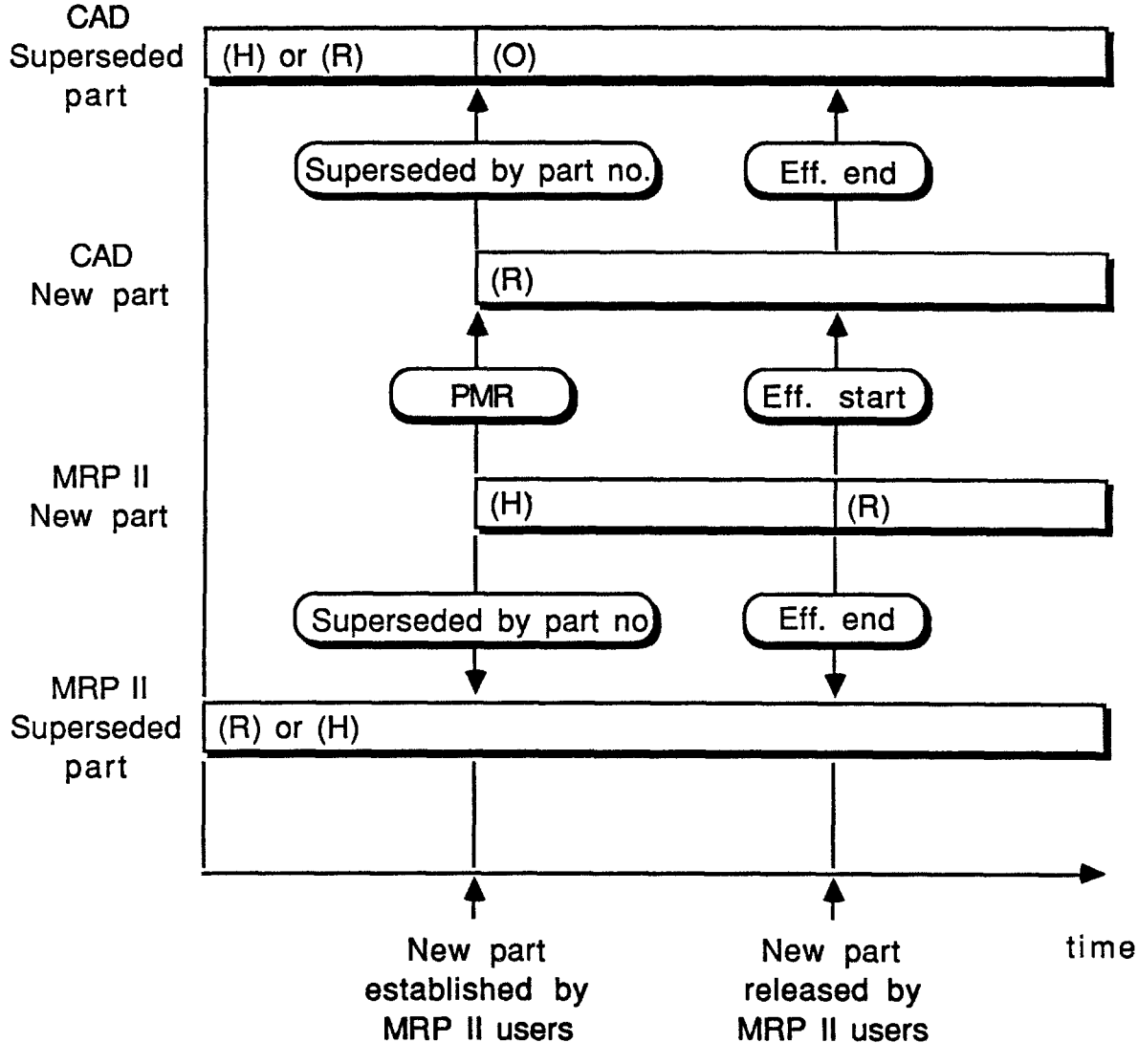


Figure 4.3: Status Diagram for Adding New Parts with Hold Status Via MRP II.

- Revision Level

This allows for the entry of parts before plans for procuring them are complete.

In either case, prior to inserting the part, the system performs a consistency check in both the CAD and MRP II systems to verify that the part number does not exist already, either as a part in both databases, or as a working part in CAD. If it does not, the part and revision records are added to MRP II.

Since it is assumed that parts added to the MRP II area of the integrated system do not have drawings, the two part record fields, "Drawing Number" and "Drawing Size," are given the value "inapplicable."

To maintain consistency in both areas of the integrated system, CAD part and revision records are also established for the new part. The part record consists of the appropriate information from the MRP II record, including the value "inapplicable" for the drawing number and size. The revision record uses the MRP II revision record data, plus the value "inapplicable" for the drawing file name. The status of the new CAD data is automatically set to "released" regardless of the status entered by the MRP II user, allowing MRP II users to enter a part with a "hold" status and subsequently release it without requiring any intervention from CAD users.

If the part has been assigned a "released" status (Figure 4.2), then several other activities occur. If the new part supersedes another, this fact is used to update the record of the superseded part in both the MRP II and CAD databases. Further, the effectivity start date of the new part's revision, which was transferred to the new revision record in CAD, is also used to update the effectivity end date of the latest revision of the superseded part in both MRP II and CAD, and the same revision is given an "obsolete" status in CAD. No further operations are necessary to make the part active.

if the part has been assigned a "hold" status (Figure 4.3), making the part fully

active is a two step process. As part of the insersion of the part, the "supersedes part number" is used to update the "superseded by part number" of the old part, if applicable. Also, the latest revision of the superseded part is given an "obsolete" status in CAD. Because the effectivity date of the new part was not entered however, it is necessary to release the part as a separate operation. The only required information for the release of the part is:

- Effectivity Start Date

When this operation is invoked, the status of the MRP II revision record for the part is changed from "hold" to "released." In addition, the effectivity start date is transferred to the CAD revision record for the new part, and is used to update the effectivity end date of the latest revision of the superseded part in both MRP II and CAD.

4.2.4 Adding New Revisions Via CAD

As with new parts, the creation of new revision levels of existing parts is assumed to be primarily the responsibility of CAD users, though there are cases when MRP II users require this capability (revising the parts created via MRP II). The creation of a new revision level represents an engineering change to the part which is intended to improve its design in some aspect, such as performance, safety, reliability, or manufacturability. The status diagram for the exchange of data during the creation of a new revision level initiated in CAD is shown in Figure 4.4, which depicts the states of the previously active revision as well as the states of the new version.

The process begins when the CAD designers responsible for a particular part are notified that a change in the part's design is desired. This notification may come from Manufacturing, concerned about manufacturing problems possibly attributed to the design, such as a high scrap rate, or extensive machining time and/or cost.

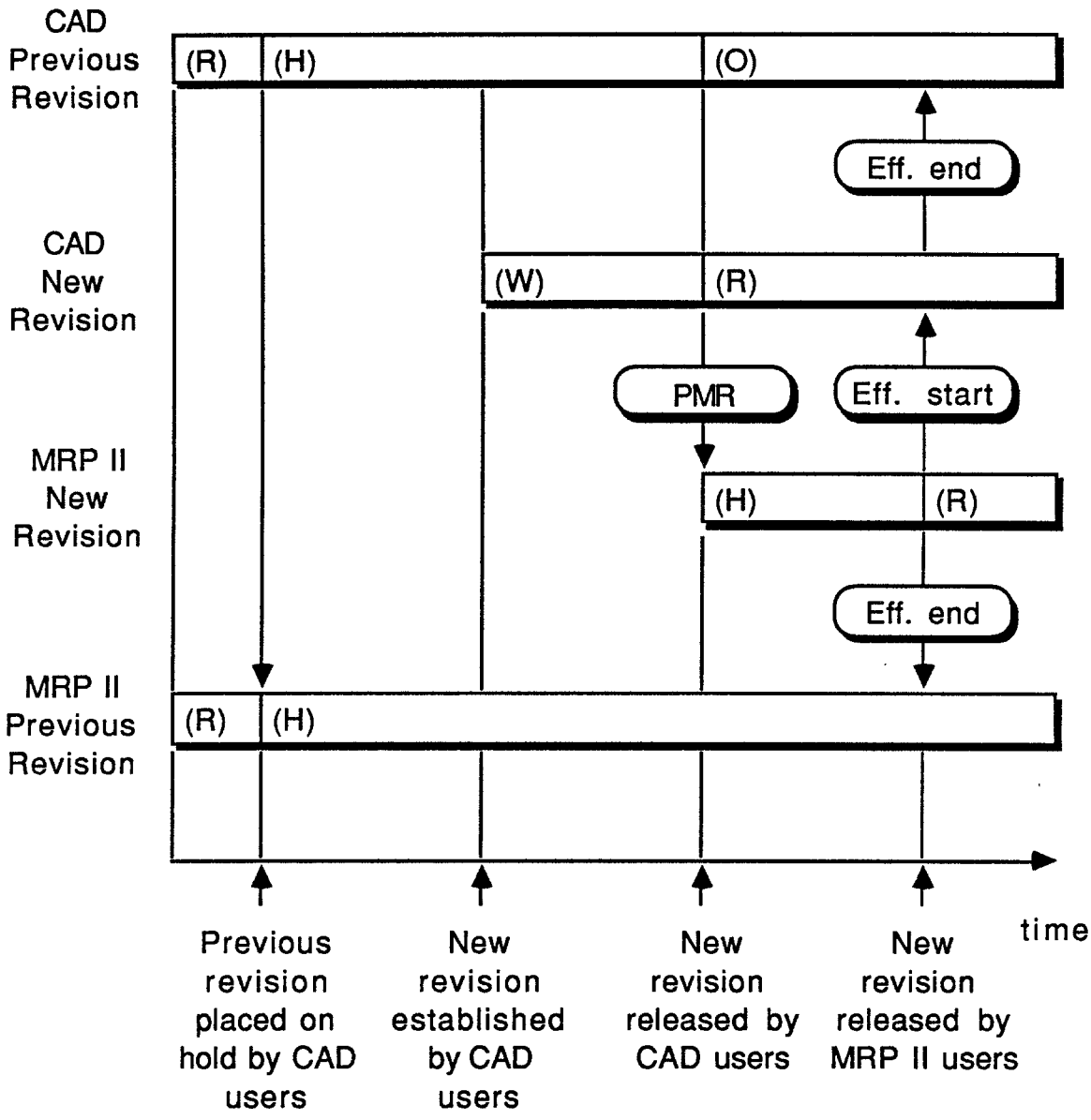


Figure 4.4: Status Diagram for Adding New Revisions Via CAD.

It may also come from Customer Service, if reliablity problems have been expressed by customers. Or the request may be part of a regular design evaluation aimed simply at bringing the product "up-to-date."

Regardless of the source of the request for an engineering change, there are a number of possible scenarios that may occur in response to it. The model of the integrated system is intended to be flexible enough to handle these situations.

Assuming that the current revision data for the part initially has a "released" status in both CAD and MRP II, either system may elect to place the data on hold while the design is being reviewed. If the revision is placed on hold in the CAD system, as shown in Figure 4.4, it is usually because there may be serious safety or performance problems with the current design, and all use of the part should be suspended. Accordingly, if this occurs, a hold on the same data in MRP II is automatically triggered. If only MRP II users place the revision data on hold, the CAD status is not affected. This may occcur if problems that become apparent during manufacturing activities are thought to be correctable by design changes. MRP II users may in such cases place a hold on the revision data while requesting a design review.

If the revision is on hold in CAD, and it is subsequently determined that the current part design is adequate, then the part can be rereleased in CAD. The data is not automatically rereleased in MRP II, but instead a message is sent to MRP II users indicating that no design problems are apparent, and that the part may be rereleased, subject to any further investigation of MRP II related problems (manufacturing, vendor quality) that may have been the cause for the design review.

If the part was not on hold in CAD or MRP II and the current design is found to be satisfactory, no specific action is required. If the part was on hold in MRP II but not in CAD, the only CAD action necessary is to notify MRP II users that the

design is not being revised, allowing MRP II users to concentrate on determining the nature of the problems that precipitated the hold.

If it is determined that a revision to the design is necessary, the revised design is created in CAD; its revision record is established by the designer, who must provide:

- Part Number
- New Revision Level
- Drawing File Name

If known to the designer, the effectivity start date may also be entered, though it is not required.

The status of the new revision record is set to "working," as indicated in Figure 4.4. If not specified, the effectivity start and end dates are left as unknown quantities. Any number of revisions with "working" status may exist for a given part at any time, allowing for the study of various alternatives. As with a new part, the revision data is maintained locally by CAD prior to its release.

Upon its release from CAD, the subsequent chain of events involving both CAD and MRP II is similar to that occuring when a new part is released, since that, likewise, is regulated by the status of the revision record. First, a record containing the new revision information (part number, revision level, and effectivity start date if assigned) is established in MRP II; the status is set to "hold," as shown in Figure 4.4, to give MRP II users time to prepare for the release of the new revision.

Second, in CAD, the status of the previously active revision is changed, from either "hold" or "released" to "obsolete," also shown in Figure 4.4. Since the model is designed to operate in a discrete parts, make-to-stock environment, only one revision of each part can be active at a given time; new revisions automatically supersede all previous revisions of the same part.

Finally, the status of the new CAD revision is changed from "working" to "released," as depicted in Figure 4.4, making the part active within CAD. For the new revision to become active in MRP II, the effectivity start date must be determined, (assuming it was not specified by the designer), and any other changes to the routings or vendor information required by the revision should be made. If the revision corrects a serious design deficiency, the new revision should be made effective as early as possible; otherwise the decision is based on current levels of inventory and pending orders for the old revision, and the set-up time required to begin producting or purchasing the new revision.

Once the revision data is finalized and complete, the new revision can be released within MRP II. When this occurs, the MRP II status of the new revsion is changed from "hold" to "released," as shown in Figure 4.4. In addition, the effectivity start date of the new revision is placed in the effectivity end date field of the previous revision level in both databases, and is also used to update the effectivity start date for the CAD revision record of the new revision. Because MRP II uses effectivity dates to determine which revision is active, there is no need to explicitly make the prior revision obsolete in MRP II.

4.2.5 Adding New Revisions Via MRP II

As with the insersion of new parts, it is assumed that design revisions entered into MRP II relate to parts without design drawings and therefore primarily under the control of MRP II users. New revisions may be entered into MRP II with either a "hold" or "released" status. Figure 4.5 depicts the status diagram for the insersion of a revision with a "released" status, while Figure 4.6 shows the status diagram for the insersion of a revision with a "hold" status. Each diagram shows both the revision being added as well as the previously active revision.

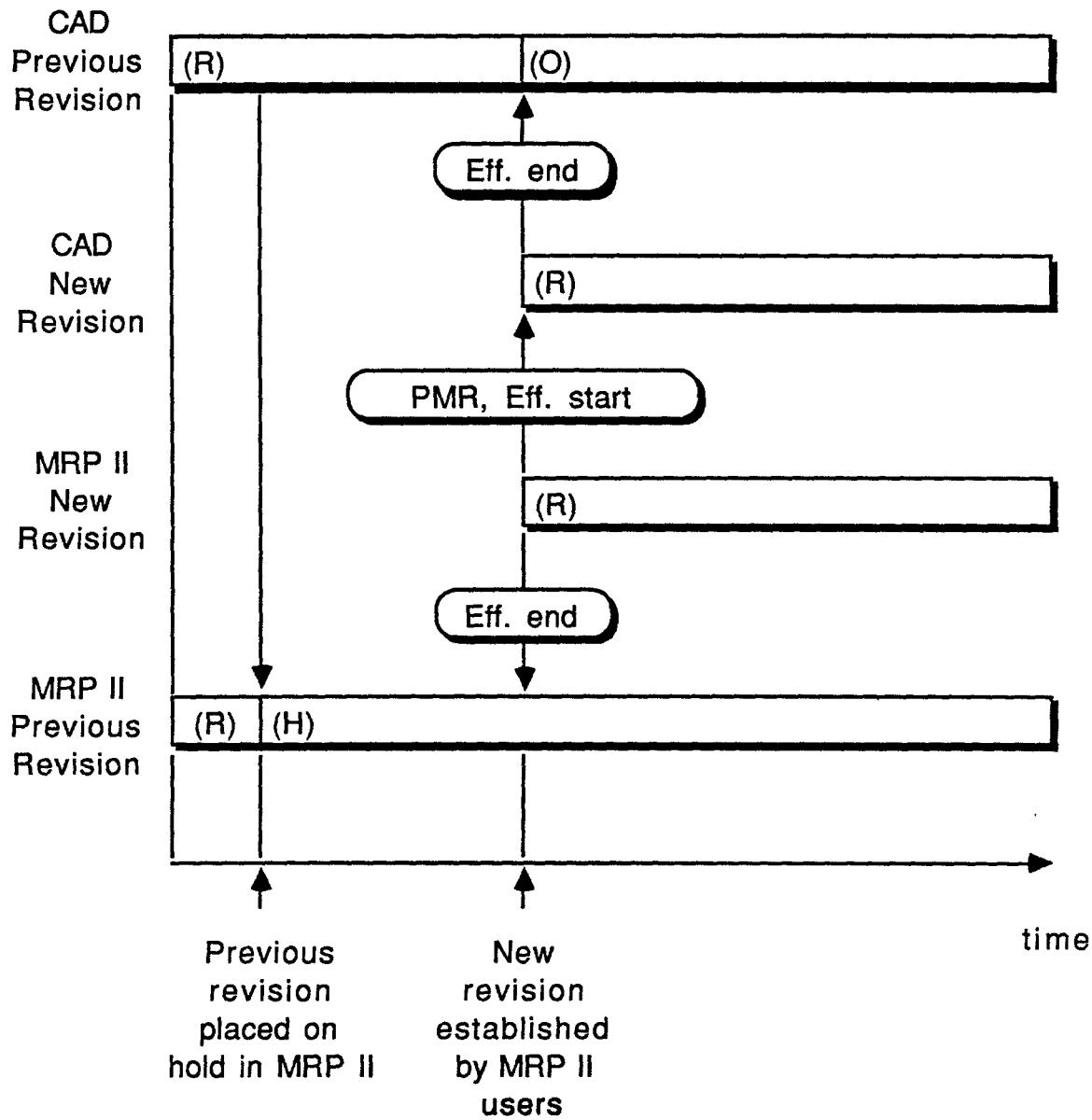


Figure 4.5: Status Diagram for Adding New Revisions with Released Status Via MRP II.

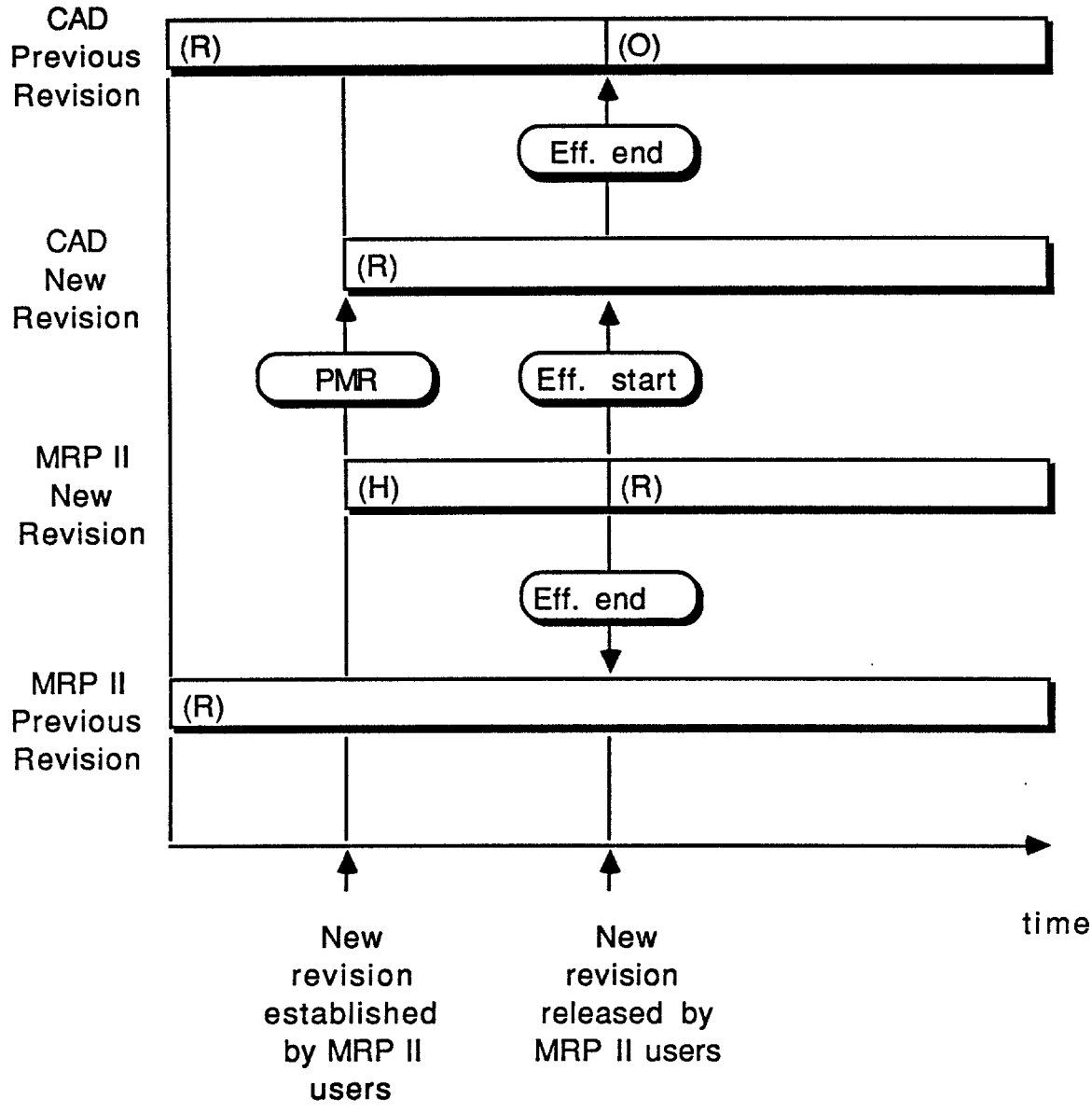


Figure 4.6: Status Diagram for Adding New Revisions with Hold Status Via MRP II.

During the changeover to the new revision, MRP II users may choose to change the status of the former revision to "hold," as shown in Figure 4.5, or may leave it "released," which is shown in Figure 4.6. If the previous revision is placed on hold and it is later decided that a revision to the design is not necessary, the old revision can simply be rereleased. In either case, the status of the previous revision data in CAD remains unchanged prior to the creation of the new revision.

To be added to MRP II with a "released" status, the revision must be identified by:

- Part Number
- Revision Level
- Effectivity Start Date

Before the revision is added, the system first verifies that there is a part record in MRP II with the same part number and that a revision with the same revision level does not exist for the part. Second, a new revision record containing the same information is generated in CAD, with the value "inapplicable" placed in the drawing name field to indicate that there is no drawing for the part. The status of the CAD revision is set to "released," as indicated in Figure 4.6, and the status of the previously active revision is changed to "obsolete." When these actions have been successfully performed, the new revision is added to the MRP II database, and the effectivity start date of the new revision is recorded as the effectivity end date of the previous revision, both in CAD and MRP II, as shown in Figure 4.6.

If the revision to the part is entered with a "hold" status, the effectivity start date is not required. In this case, the activities preceding the actual addition of the new revision to MRP II are similar to those just described: the system checks for consistency, adds a new CAD revision record with "released" status, gives the

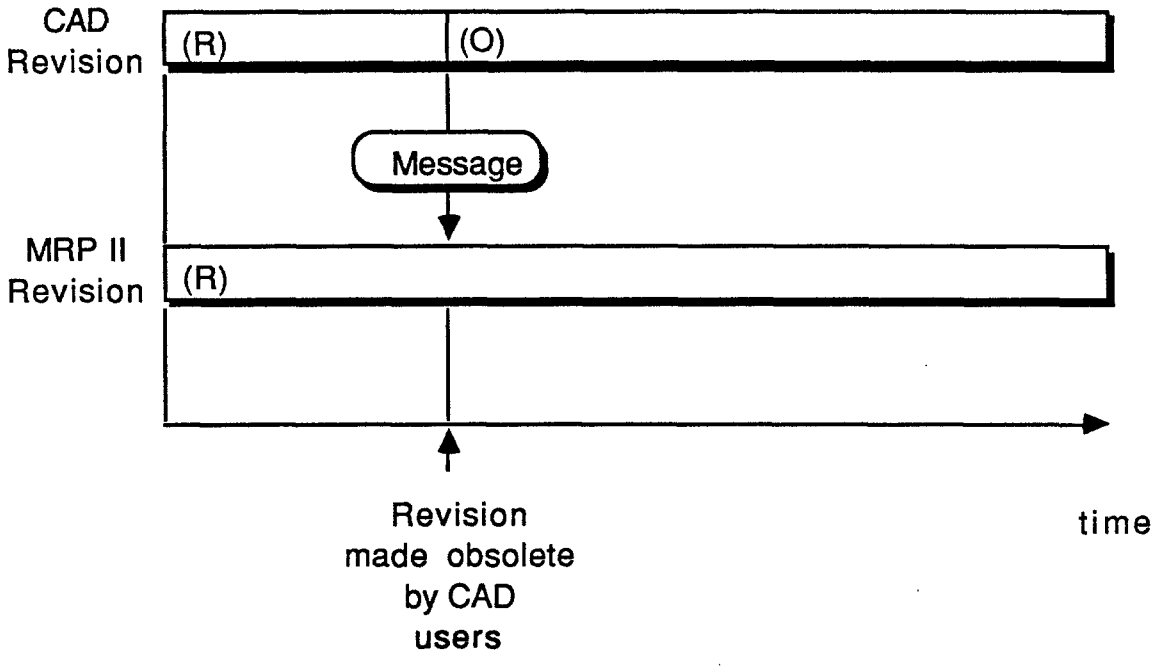


Figure 4.7: Status Diagram for Making Parts Obsolete Via CAD from Released Status.

previous CAD revision an "obsolete" status, and then adds the new revision to MRP II. These activities are shown in Figure 4.5.

The revision can then be released in MRP II at a later date by adding its effectivity start date. The status of the new revision is then changed to "released," and the effectivity start date is used to update the CAD revision record for the new revision. Also, the effectivity end date of the old revision is set to the effectivity start date of the new revision in both MRP II and CAD, as depicted in Figure 4.5.

4.2.6 Making Parts Obsolete

In addition to the implicit supersession process for making parts obsolete, the model allows users to explicitly make parts obsolete in CAD. The status diagrams for this are shown in Figures 4.7 and 4.8.

Revisions may be assigned an "obsolete" status in CAD from either a "released"

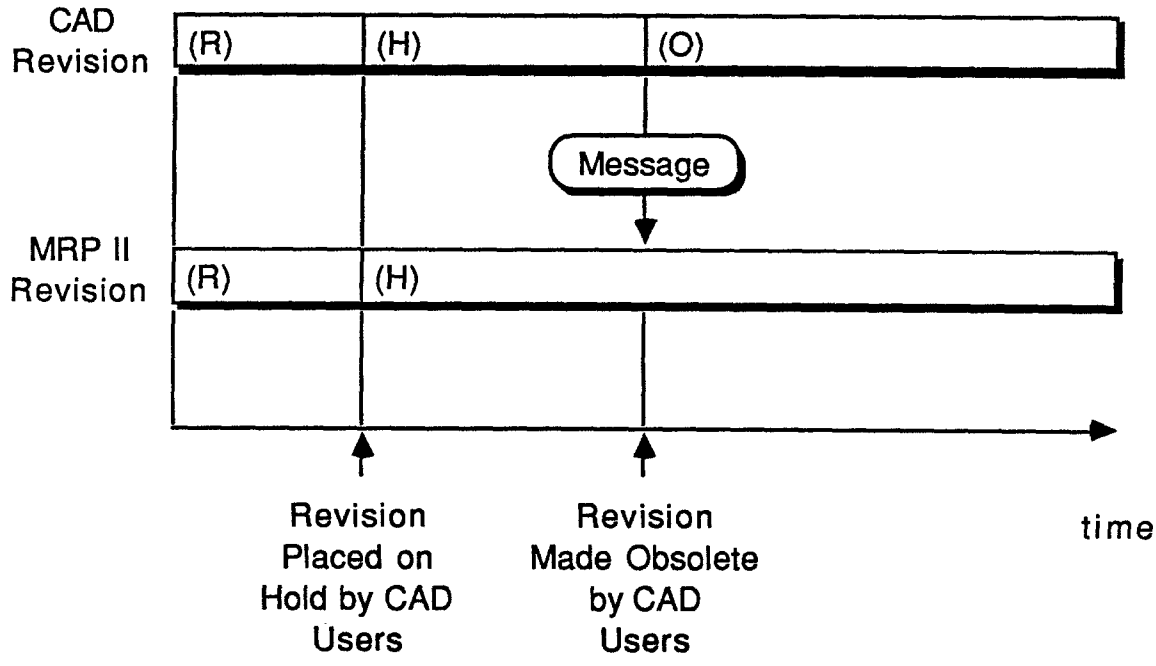


Figure 4.8: Status Diagram for Making Parts Obsolete Via CAD from Hold Status.

or "hold" CAD status, since the obsolescence may be due to a routine phasing out of a part or due to safety or performance problems. For routine phasing out, shown in Figure 4.7, the change may be made directly from "released" status; the status of the revision in CAD is changed to "obsolete," and a message is sent to MRP II users indicating the change in status. No change to the MRP II status occurs however, allowing the use of existing inventory and pending orders.

If the revision is being made obsolete for safety or performance reasons, as shown in Figure 4.8, it should be placed on hold in CAD prior to being made obsolete, so that it will not be used by either system. As previously described, a hold in CAD automatically invokes a similar hold on the same data in MRP II, to prevent MRP II related activities involving the part. Also, a message is sent to MRP II users to alert them to the obsolescence. This may eventually lead them to delete the part from MRP II.

4.2.7 Deleting Parts

Part deletion may be initiated by either MRP II or CAD users. It is assumed that most parts, and in particular, all parts with drawings associated with them, will be controlled by the CAD system. Hence the deletion of part data will generally be initiated by CAD users. Parts without associated drawings, however, may be controlled by MRP II users; the ability to delete such part data via MRP II is thus provided. For further flexibility, design data with drawings can be deleted by MRP II users while the same design data remains in CAD, allowing MRP II users to eliminate obsolete data that CAD users wish to maintain for historical purposes or for use in future designs.

Deletions are generally processed by part number; when a part is deleted, the data describing all versions of the part, as well as the part record itself, are deleted. The exception to this is that any revision of a part with a "working" status in CAD may be individually deleted, providing for the deletion of designs that never reach "released" status. Because revisions in this category are local to CAD, no consistency checks in MRP II are required.

Before a part can be deleted, certain constraints must be satisfied. CAD users may initiate the deletion of CAD-controlled parts only; those originally generated by MRP II must be deleted via that application system. The process of deleting a CAD-generated part via CAD is shown in Figure 4.9. To delete such a part via CAD, all revisions of the part must have a CAD status of either "working" or "obsolete;" if the part still has an active (i.e., "released" or "hold" status) revision, it cannot be deleted. This is to prevent the accidental deletion of an active part. Further, the part cannot be used in a product structure in the CAD database. If these constraints are satisfied, the system checks MRP II to see if the part is

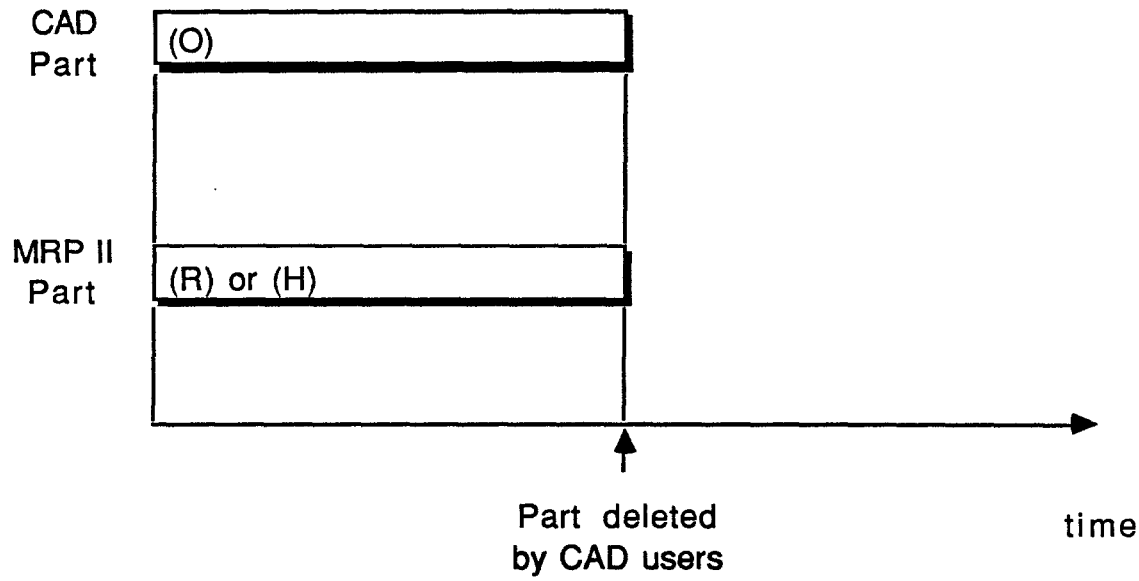


Figure 4.9: Status Diagram for Deleting CAD-Generated Parts Via CAD.

used in any product structures in that database, has non-zero inventory, or has any outstanding orders. If any of these are found to be true, the part cannot be deleted in MRP II, and therefore cannot be deleted from CAD. If the part is not used in any product structures, has zero inventory, and no outstanding orders, then all revision records for the part in both MRP II and CAD are deleted, as are the CAD part and MRP II part master records. There is no restriction on the status of the revisions in MRP II; they may be either "hold" or "released."

CAD-generated parts may also be deleted via MRP II, though the results of this operation are somewhat different, as shown in Figure 4.10. The part is first subjected to the checks on the MRP II database described above; it must not be used in any product structures, must have zero inventory, and must have a zero quantity on order. If the part fails any of these checks, indicating that it is still active in MRP II, then nothing is deleted, and the operation fails. If the part passes all of these checks, then the system checks the CAD database. Once again, the part

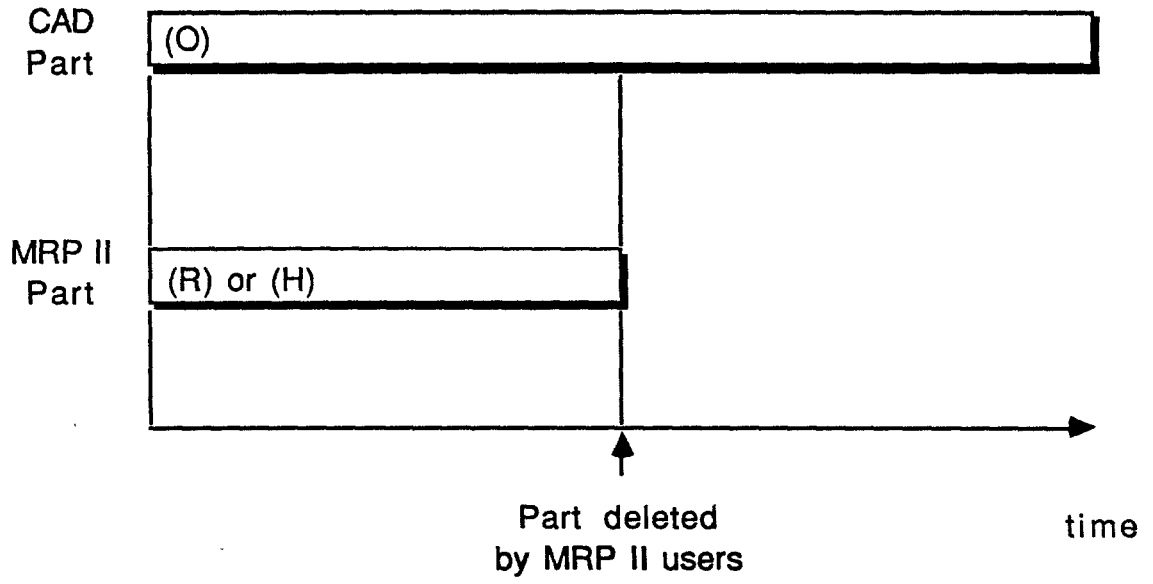


Figure 4.10: Status Diagram for Deleting CAD-Generated Parts Via MRP II.

must not have any revisions with "released" or "hold" status, and must not be used in a product structure. If these checks are successful, then the part master record and all revision records associated with the part are deleted from MRP II. All part and revision data remains in CAD however, since it is CAD that has control of the final disposition of this information. CAD users may later decide to delete the part, or it may be maintained for historical purposes or use in future designs.

Unlike CAD-generated parts, MRP II-generated parts may only be deleted via MRP II, since these are assumed to have no design drawings, and thus be under the control of MRP II. If a CAD user attemps to delete such a part, an appropriate message is printed out and the operation fails. The status diagram for deleting an MRP II-generated part via MRP II is shown in Figure 4.11.

Before processing the operation, the system imposes the same constraints as those for CAD-generated parts: the part must not be used in a product structure, it must not have any inventory, and it must not have an outstanding quantity on

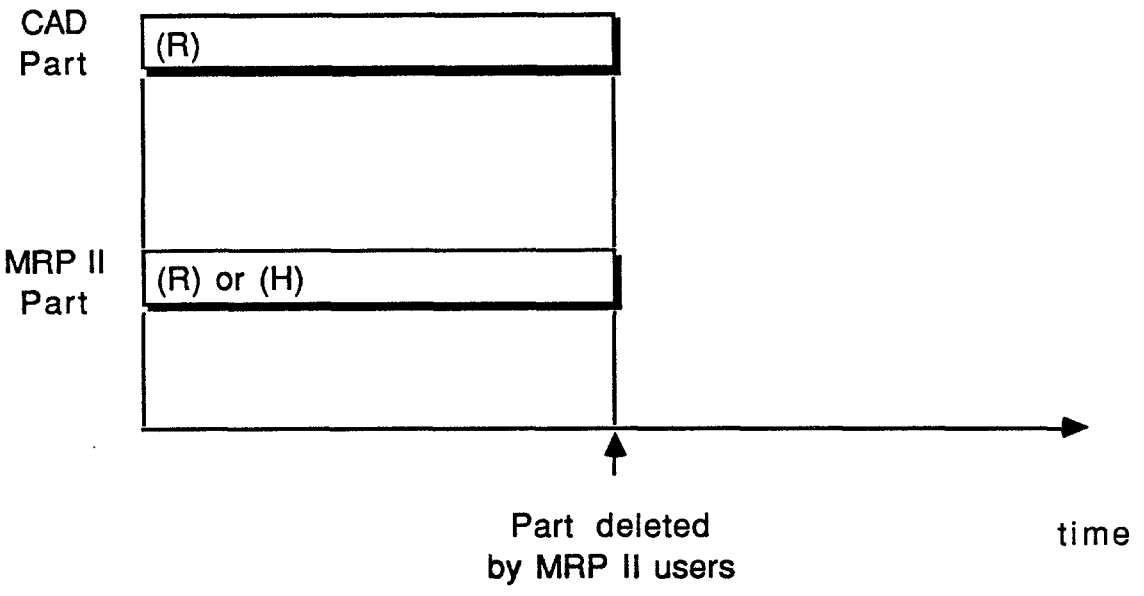


Figure 4.11: Status Diagram for the Deletion of MRP II-Generated Parts Via MRP II.

order. If this is the case, then all information related to the part, the part, revision, and latest revision records are all deleted from MRP II, and from CAD as well, without any additional checks in the CAD database.

4.3 Product Structures

The MRP II/CAD integrated system provides for the maintenance of single level product structures, or bills of material, in both application systems. Bills of Material identify the components of an assembly, and can be viewed as an extension of the part master data of the assembly, providing further static information about the item. As in the creation of new parts, it is CAD that typically initiates new product structures as part of the design process, in order to describe assemblies. It is also generally CAD that modifies the structures as necessary to reflect engineering changes to the design. Changes to bills of material are occasionally made by MRP II users as well, in order to make the structures better reflect the manufacture of a

given product.

The handling of product structure maintenance by the system is similar to the handling of part master data. Product structures are represented and recorded by the individual parent-component relationships comprising them. For each relationship, the following information is recorded in a relationship record:

- **Parent Part Number.** This is the part number of the parent assembly. Each assembly will have as many relationship records as it has components (greater than or equal to one).
- **Parent Revision Level.** This is the revision level of the assembly. Along with the parent part number, it uniquely defines the particular assembly involved in the relationship.
- **Item Number.** The item number is a sequence number used simply to uniquely identify each component within an assembly.
- **Component Part Number.** This is the part number of the component in the relationship.
- **Quantity.** This is the quantity of the component used per unit of measure of the assembly in question.

It is the total set of relationships for a part that defines its single-level bill of material. Through the concatenation of single level bills of material, a multi-level bill of material for an assembly can be constructed, indicating all subassemblies and components necessary to make that assembly. The same relationship information is recorded in both the MRP II and CAD systems, as shown in Table 4.3. No component revision level information is used in the relationship record. Because the system is designed for a make-to-stock environment, in which there is only one active

MRP II	CAD
Parent Part Number	Parent Part Number
Parent Revision Level	Parent Revision Level
Item Number	Item Number
Component Part Number	Component Part Number
Quantity Per Assembly	Quantity Per Assembly

Table 4.3: Product Structure Data Maintained by MRP II and CAD in Relationship Records.

revision of each part at any given time, it is assumed that assemblies will always use the most recent revisions of their components. The specific revision of a component that is active in all product structures requiring the part is determined by the effectivity dates of the various revisions of the component part. The revision level of the assembly is maintained in the component relationship record so that the bills of material of several revision levels of the assembly can be maintained simultaneously, which is essential during the transition between different revisions. The combination of parent part number and revision level refer back to the revision record of the parent part, where the effectivity dates of the assembly are recorded. MRP II uses these effectivity dates to determine which of the versions of the assembly is active, and therefore which set of relationships is valid for planning.

There are two levels of changes that may occur in product structures. The first is at the level of the components themselves, as reflected by revision changes to these parts, while the second is at the level of the assembly itself, involving a change to one or more of its relationship records. A difficulty arises in either of these cases, namely the determination of the effects of these changes on the assembly. When a part is changed, the impact of the change on all related items, including both the parent assemblies requiring the part as well as other parts at the same level which interact with the part, must be determined. Typically, if a component of

an assembly experiences a revision level change, it is not necessary to change the revision of the assembly. If however, a component relationship is changed, (e.g., a particular component of an assembly is replaced by another part, or the quantity of a given component in the bill of material is altered), a revision change at the assembly level is usually appropriate. Alternatively, it may be necessary to change the part number of the resulting assembly altogether, creating a new assembly. The specific guidelines for such changes are documented in detail by Harhalakis [11]. Moreover, the impact of a revision change to a part on neighboring parts in bills of material is virtually impossible to predict, as it always depends on the specific design, i.e., whether or not physical changes to the neighboring part are required.

The integrated system is designed to facilitate changes at both levels, though the crucial decision making is left to the user. When a new version of a part is created, for example, the assemblies requiring the part should be checked to see if there is any need for changes at the assembly level (e.g., a new revision level for the assembly, a different quantity of the component, and so on). No change at the assembly level is made automatically however; the decision making process and the appropriate actions are left to the user. At all times, however, consistency between MRP II and CAD is preserved.

Changes that can be made at the assembly level include:

- Adding components
- Deleting components
- Replacing a component with another component
- Changing the quantity of a component used in an assembly

Each of these will likely require at least a new revision level for the assembly. If

a new assembly revision level is necessary, then prior to making such changes to the relationship records, the user should create a new revision record for the resulting assembly following the procedures described in sections 4.2.4 or 4.2.5, depending on whether the change is originating in CAD or MRP II, respectively. If a new assembly part number is required, the user should create a part master record for the new assembly following the steps in sections 4.2.2 or 4.2.3, again depending on whether the change is occuring via CAD or MRP II, respectively.

Whenever new revisions are created, the system copies all component relationships for the previous revision to the new revision. These can then be modified by making the additions, deletions, and so on, that differentiate the old and new revision of the assembly. The effectivity dates of the new assembly revision hence determine when the new structure is to be active for planning. When a new assembly is created to replace another due to an engineering change, the relationships from the old assembly are not automatically copied, since there may be significant differences between the old and new structures. The system does provide a copying feature for this case, in the event that the structures are similar.

To facilitate this procedure in the event that the user has not already created a new revision or part, the system asks the user prior to making each assembly level change whether the operation he or she is invoking will necessitate either a new revision level or part number for the parent assembly. If the user answers "yes" to one of these questions, he or she is prompted for the details of the new part or revision, and it is added following the procedures outlined in sections 4.2.2 – 4.2.5, depending on the source of the operation. If the assembly level change is initiated by a CAD user, the procedures are the same as adding a new revision or part via CAD (sections 4.2.2 and 4.2.4, where the new part/revision data is initially created locally within CAD with a "working" status, pending its formal

release.) Likewise the new product structure, and any changes to it, are maintained locally until the new part/revision is released. If performed via MRP II, the system will follow the procedures of sections 4.2.3 and 4.2.5 in creating the new part or revision, respectively, which involves the immediate updating of the CAD database, with respect to both the new part/revision data as well as the product structure information.

The system does not ask the user about revision level or part number changes however, if the assembly has a "working" status in CAD. Since parts and revisions in this category are not yet finalized, the change is processed without any questions to the user.

In addition to allowing users to create new parts and revisions within the assembly level operation, all relationships defined for the current part/revision in the database of the application initiating the operation are copied to the new one so that the change being performed will be made on the proper bill of material.

Though this is the desired procedure, the system does not enforce it; additions, deletions, and modifications can be performed on product structures without creating a new revision or part. This provides flexibility in the event that a new revision or part number is not required by the change in question (such as correcting data entry errors or processing multiple operations as part of a single engineering change). It is the user's responsibility to follow established procedures in determining whether a new revision level or part number is necessary.

In the following sections, the basic assembly level operations are described. The updating of the CAD and MRP II databases is performed in a manner parallel to that used in the creation and maintenance of part master data, based on the status of the assembly and its components in the two applications.

4.3.1 Adding Component Relationships Via CAD

Component relationships may be added via CAD or MRP II. Typically CAD will be the source of the first product structure for a given assembly, as well as the origin of engineering changes requiring the modification of components in an assembly. However, it may also happen that MRP II users will add components. This may occur if MRP II users find it necessary to regroup the components of an assembly to form one or more intermediate assembly stages to improve the representation of the manufacturing operations. It may also occur if MRP II users decide to manufacture a part that CAD users assumed would be purchased, and hence created no product structure. MRP II users could in such cases create the product structure using either existing or new parts.

If the addition of the component calls for either a revision change or new part number at the assembly level, the new revision or part should be created first, and the addition made to the new bill of material, regardless of whether the addition is being made via CAD or MRP II. If the new revision or part is not created prior to the operation, the system allows the user to create it as a part of the operation, as described above.

To add a component relationship via CAD, all fields in the record must be specified:

- Parent Part Number
- Parent Revsion Level
- Item Number
- Component Part Number
- Quantity Per Assembly

The indicated revision of the parent part, or assembly, may have a CAD status of either "working," "hold," or "released," it cannot have an "obsolete" status. Typically, designers will create the product structure of an assembly before it is finalized, i.e., while it has "working" status. If the assembly revision has either a "hold" or "released" status in CAD, the system also requires that the part/revision combination exists in MRP II. These requirements allow the product structure of an assembly to be compiled prior to its release from CAD, but prevent the creation of a product structure for an assembly that has been deleted from MRP II but not CAD.

The component part must have a revision level with a status of either "hold" or "released" in both CAD and MRP II. Thus parts cannot be used as components in product structures prior to the release of their first revision by CAD users and the subsequent creation of a part master record in MRP II. Again, the system does not allow the use of parts deleted from MRP II but not CAD to be used as components in product structures.

In addition, the system checks to make sure the item number assigned to the component part has not already been used by another relation in the same structure. The system does, however, allow, the same component to be recorded in two different relationship records, allowing users to break up the quantity of a component into two or more items to better represent the manufacturing operations of the assembly.

Finally, the system ensures that the resulting relationship does not cause a "loop," i.e., a use of the assembly as a component of itself. To do so, the component part number must not be the same as the assembly part number; in addition, the system searches through all levels of the product structure of the component part (if applicable) for occurences of the parent part number. If any are found, the

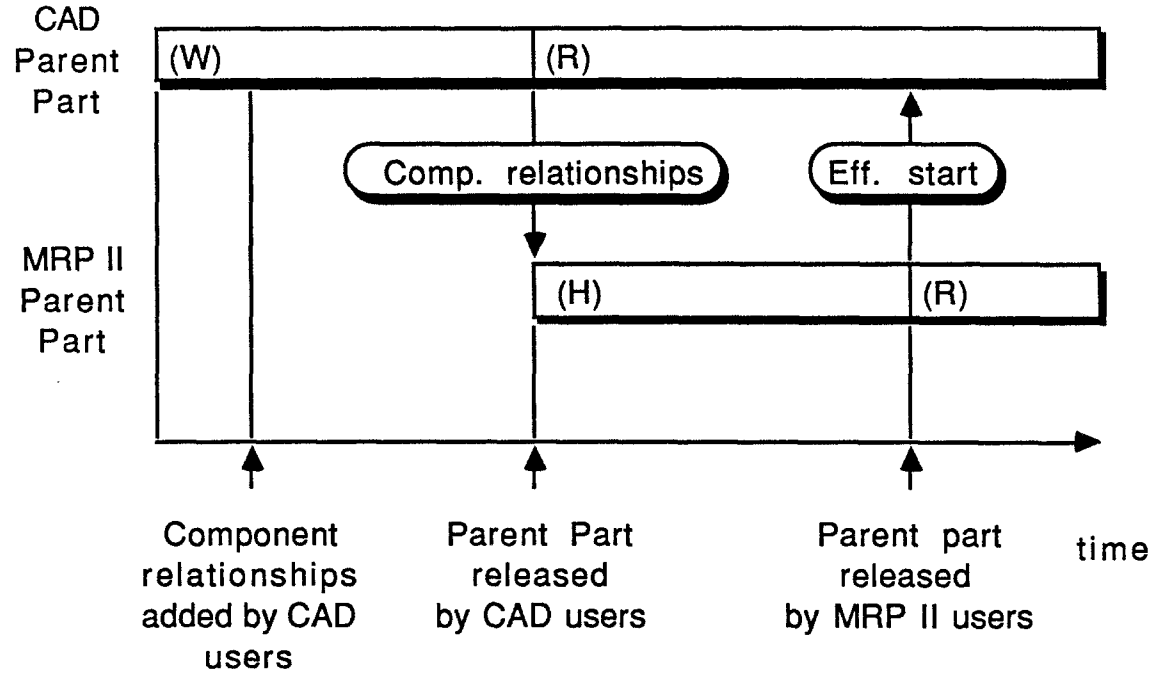


Figure 4.12: Transfer of Component Relationships from CAD to MRP II beginning with Assemblies having Working Status.

relationship cannot be added.

If the database checks are successful, the relationship record is added to the CAD system. If any one of the database checks fails, a message describing the failure is printed, and the operation fails.

The timing of the transfer of the component relationships from CAD to MRP II is dependent on the CAD status of the parent part/revision combination. The two possible cases are shown in Figures 4.12 – 4.13.

In the first case, shown in Figure 4.12, the revision of the assembly being constructed begins with a "working" status. This is perhaps the most common situation, as designers typically construct the bill of material for an assembly before it is released. As is the case with the data associated with the revision itself (and the part data as well, if the revision in question is the first one for the assembly),

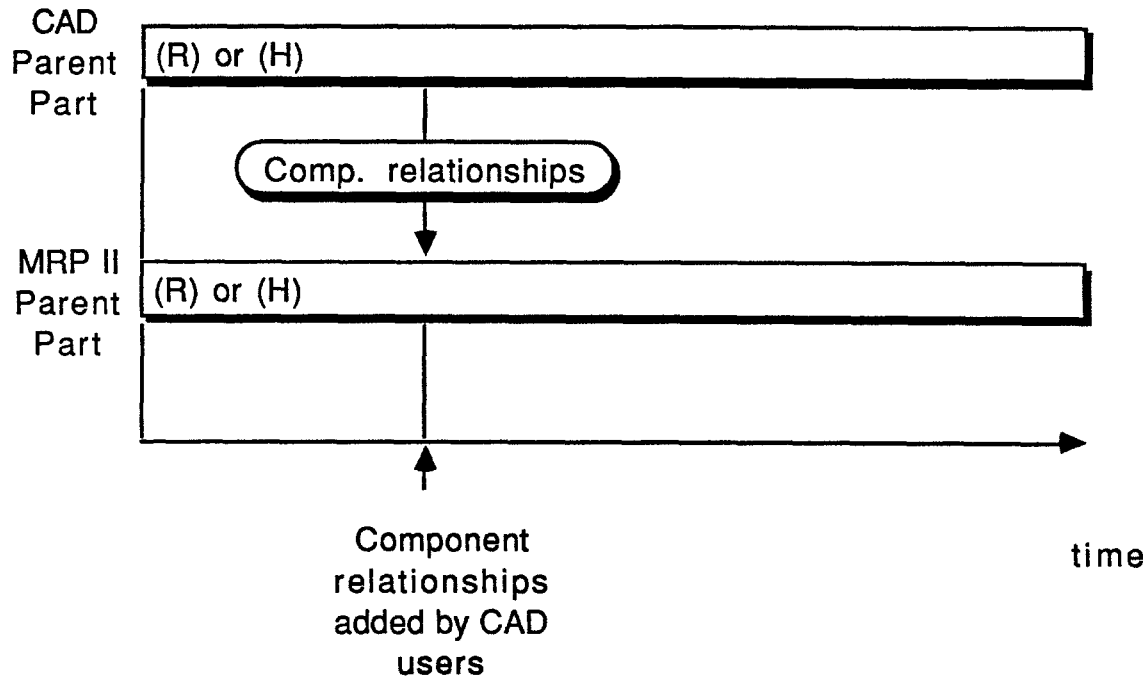


Figure 4.13: Transfer of Component Relationships from CAD to MRP II beginning with Assemblies having Hold or Released Status.

the component relationship data remains local to CAD until the part is finalized and released. When it is released, the component relationship data is added to the MRP II database at the same time as the revision data, which is assigned an MRP II status of "hold." When MRP II users have determined and entered any information not provided by the designers of the assembly, such as the effectivity start date, the revision is released to MRP II (see section 4.2.2). It is then the effectivity dates of the new revision which determine when the new relationship becomes effective.

In the second case, depicted in Figure 4.13, the revision of the assembly being formed has already been finalized and has either a "hold" or "released" status in CAD. Since parent parts such as these exist in MRP II as well as CAD, the component relationships are immediately tranferred to MRP II as well. The system allows the construction of product structures for parent parts with a "hold" status

in CAD since they may again be made active at some later date.

As mentioned in the previous case, the parent part/revision combination must exist in both CAD and MRP II for a relationship to be defined. The assembly revision may have an MRP II status of either "hold" or "released." Because a "hold" status on a part in CAD automatically implies a "hold" status in MRP II, the MRP II status of the parent part will be "hold" if the CAD status of the same part is also "hold."

If the parent part has a "released" status in MRP II, then the effectivity dates of the revision should have already been defined. If it has a "hold" status, then the effectivity starting date may or may not have been defined, depending on whether the part had a "released" status prior to being put on hold, or was just transferred from CAD. In either case, the effectivity start date will be defined prior to the release or rerelease of the revision.

4.3.2 Adding Component Relationships Via MRP II

Adding component relationships via MRP II again requires the complete specification of the relationship, including:

- Parent Part Number
- Parent Revision Number
- Item Number
- Component Part Number
- Quantity per assembly

If a new revision level or part number at the assembly level is called for due to the change, then the user may elect to create the new part master data either prior to the addition operation, or as a part of it.

Before adding the relationship to either database, the system verifies that the parent part and revision level exist in both CAD and MRP II. Further, the CAD status of the parent revision must be either "released" or "hold," but not "obsolete," since obsolete parts cannot be used in product structures. There is no restriction on the MRP II status of the parent revision. As before, the component part must have a revision level with a status of either "hold" or "released" in both CAD and MRP II.

Once again, in addition to these requirements, the system performs the same checks described above to ensure that the item number is unique within the assembly and that there are no "loops" formed by the addition of the relation. If all of the checks are successful, the operation is completed by the system.

As in the creation of new parts and revisions via MRP II, relationship records are transferred to CAD immediately, regardless of the status of the parent part/revision in MRP II or CAD, since all parts involved are finalized. This is shown in Figure 4.14. Depending on the timing of the insersion and the status of the parent part in MRP II, the parent part revision may or may not have an effectivity start date assigned. If the part has a "hold" status and the effectivity start date is not defined, it will be prior to the release of the part.

4.3.3 Deleting Component Relationships

Component relationships may be deleted via either MRP II or CAD, although it is assumed that CAD users will be the primary initiators of deletions. As with the addition of relationships, when the deletion of a relationship would result in a new revision level or part number for the assembly involved, the new revision or part should be created, and the deletion made on the new bill of material. This may be done directly, before the operation is performed, or as part of the operation, as

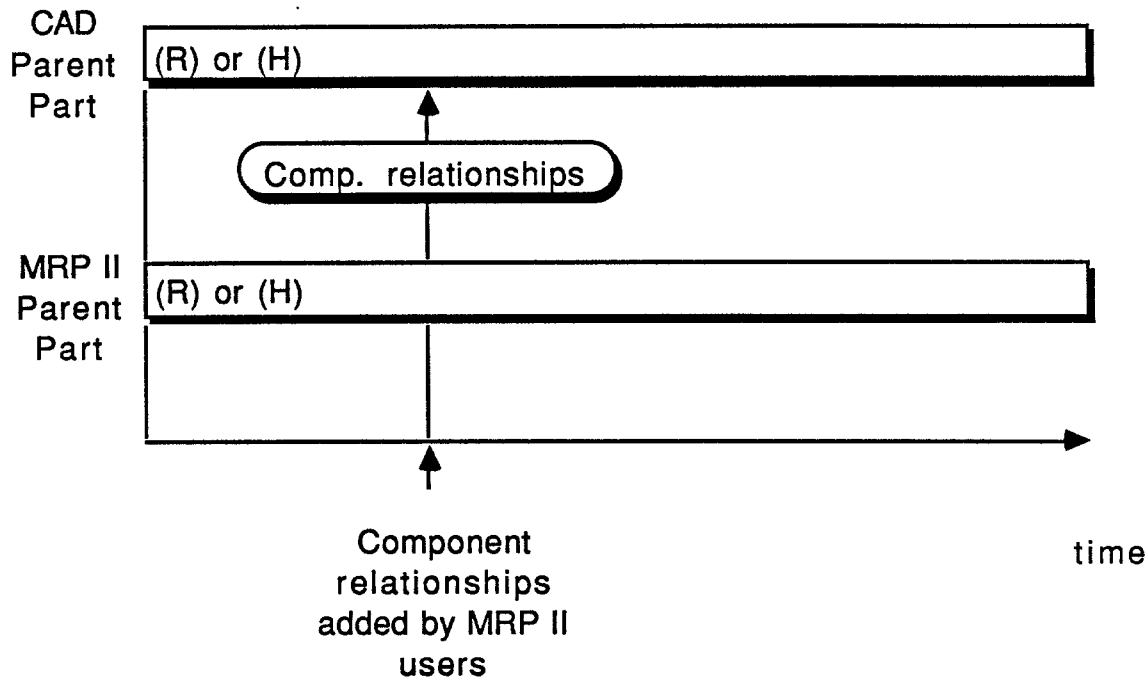


Figure 4.14: Transfer of Component Relationships from MRP II to CAD.

described above.

To be deleted via either system, a relationship must be identified by:

- Parent Part Number
- Parent Revision Level
- Item Number
- Component Part Number

The only check necessary for this operation is for the existence of the relationship record. The specific details of the operation depend on whether the deletion is performed via CAD or MRP II. If the operation is performed via CAD, the process further depends on the status of the assembly part/revision in CAD. If the relationship exists in CAD and the parent part/revision has a "working" status, then

the relationship record is local to CAD; the record is deleted from CAD with no interaction with MRP II. If the relationship exists and the parent part/revision has either a "released" or "hold" status in CAD, then it should also exist in MRP II; accordingly, the relationship record is deleted from both CAD and MRP II at the same time.

If the operation is being performed via MRP II, then the system makes sure the relationship exists both in CAD and MRP II, and if so, deletes it from both application systems.

4.3.4 Substituting Components in Relationships

The combination of deleting a particular relationship and adding another can be used to effectively substitute a given component part in a particular bill of material with a different component part. The system also provides for "mass" substitutions, i.e., the substitution of all occurences of a part as a component in a bill of material with another part. This occurs, for example, when a new part is created that supersedes a previous part. It may also occur when it is decided to replace a currently active part by another part that is also already active. This is a powerful transaction that should only be used with caution and perfomed by someone with the proper authority.

Once again, if appropriate, a new revision level or new part number for all assemblies involved should be created prior to this operation, and the operation performed on the new structures. Obviously this is not always practical; again the system lets any new revision levels and/or part numbers be created as part of the operation. As it finds each occurence of the old part as a component in a relationship record, the system asks the user if the substitution necessitates a change in the revision level or part number of the particular assembly, unless the

assembly revision has a CAD "working" status. Each time the user answers "yes" to one of these questions, the new part or revision is added as previously described, the product structure of the current revision or part is copied to the new one, and the operation is performed on the new structure. By sequencing through all of the occurrances of a part as a component, the system can be used to make all the necessary part number/revision level changes for this operation.

Users from either MRP II or CAD may originate this operation. To do so, the user must enter:

- Current Part Number
- New Part Number

As long as each relationship involving the current part number meets the requirements for deletion, and each relationship involving the new part number meets the requirements for addition, the operation is processed. For assembly revisions with either a "released" or "hold" status in CAD, both the MRP II and CAD databases are immediately updated to reflect the change; for assemblies with a CAD "working" status, which are local to CAD, the change is made only in the CAD database, and propagated to MRP II only after the new part or revision is released by CAD users. This is likely to be the most common case, since CAD users are primarily responsible for engineering changes; each new revision level or part created by CAD users to accomodate the changes will have a "working" status.

This operation is automatically invoked when a new part that supersedes another is released by CAD or MRP II users. In fact, this is its primary purpose, to replace all occurences of a part as a component with the part superseding it, at the time the new part becomes active. In this case, the system will again prompt the user about new revision levels and part numbers as it finds each occurence of the

superseded part as a component. As in the previous case, the timing of the transfer of information between CAD and MRP II depends on the status of each assembly revision being modified in the system releasing the new part.

4.3.5 Modifying Component Quantities

The final assembly level operation considered in the model is that of changing the quantity of a given component in a bill of material. This generally requires a revision change or new part number for the assembly, so a new revision or part should be established prior to this operation, and the modification made to the new product structure; alternatively, the user can create the new revision or part as part of the operation. Changing a component quantity requires:

- Parent Part Number
- Parent Revision Level
- Item Number
- Component Part Number
- New Quantity Per Assembly

If the indicated parent-component relationship exists, the modification is made. If it does not, the operation fails. If performed by CAD users on an assembly revision with a "working" status, the change is made in the CAD database only, since the relationship record is local to CAD. When the revision is later released by CAD users, all relationship records for the revision are transferred to MRP II. If performed by CAD users on an assembly revision with "released" or "hold" status, or by MRP II users, then both the CAD and MRP II databases are immediately updated.

4.3.6 Copying Component Relationships from one Assembly to Another

In addition to providing an automatic copying feature when new parts or revision levels are established as part of an assembly level change to a product structure, the system provides a direct copying function to allow users who create new parts prior to invoking an operation to copy the product structure from the old part to the new part. This operation can be performed by either CAD or MRP II users, and requires the following information:

- Copy from Part Number
- Copy from Revision Level
- Copy to Part Number
- Copy to Revision Level

When this operation is invoked, all of the relationship records for the "copy from" part number and revision level defined in the database in which the operation is originating are copied to the "copy to" part number and revsion level. As discussed in the above sections, the transfer of the newly created relationship records between the two applications depends on the application in which the operation originated as well as the status of the "copy to" assembly revision. If the operation is performed by CAD users, and the "copy to" assembly revision has a "working" status in CAD, then only the CAD database is immediately updated. The transfer of information to MRP II occurs when the new assembly is released by CAD users. If performed by CAD users and the "copy to" assembly revision has a status of either "hold" or "released," then both the CAD and MRP II databases are immediately updated. Similarly, if the operation originates from MRP II, regardless of the MRP II status of

the "copy to" assembly revision, both the CAD and MRP II databases are updated as part of the operation.

Chapter 5

Database Interoperability through Update Dependencies

To implement and demonstrate the integrated MRP II/CAD system described in chapter 4, the concept of database interoperability is being utilized. This chapter presents this concept, which is based on the Update Dependency Language, currently under development in the department of Computer Science at the University of Maryland [17] as a means for achieving interoperability. The language is being applied to the current problem both as a means to critically analyze the design of the integrated system, and to analyze the effectiveness of the language in specifying such a system.

5.1 Database Interoperability

Database interoperability is a fairly new approcach to systems integration that differs from the more traditional approach of database integration. Database integration involves the notion of a global schema, integrating the schemata of the existing databases. There are basically two ways to integrate existing databases using a global schema. The global schema can be placed between the databases and the system. In this case, the schemata of the existing databases become external schemata and the application software can be preserved, but the data must be

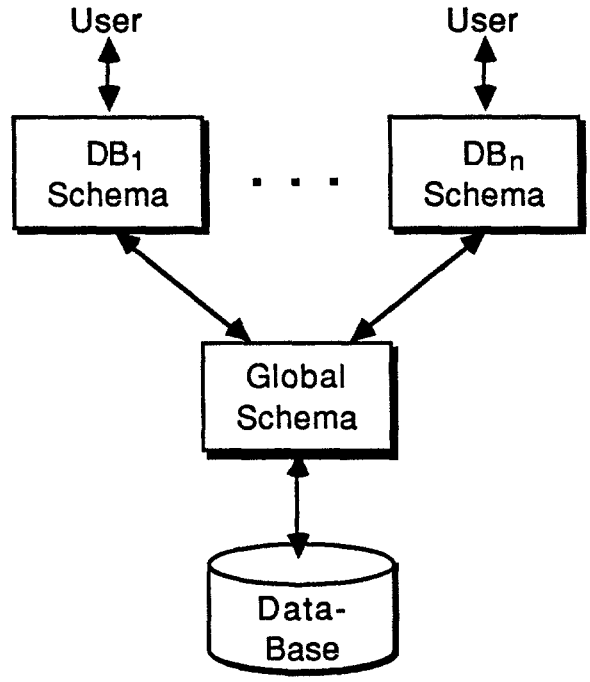


Figure 5.1: A Global Schema between the Databases and the System.

reorganized and stored under the global schema. This is illustrated in Figure 5.1. Alternatively, the global schema can be placed between the users and the databases. In this case, the application software must be rewritten, but the data need not be reorganized. This is illustrated in Figure 5.2.

The basic problem in database integration is the required initial design of a global schema which is the *union* of the schemata of the databases to be integrated.

If these databases are homogeneous, i.e., their schemata are all defined in terms of the same data model, then one encounters the following:

- Easy problems:
  - Domains may have different physical representation, e.g., integer or real
  - Domains may have different units of measure , e.g., inches and centimeters

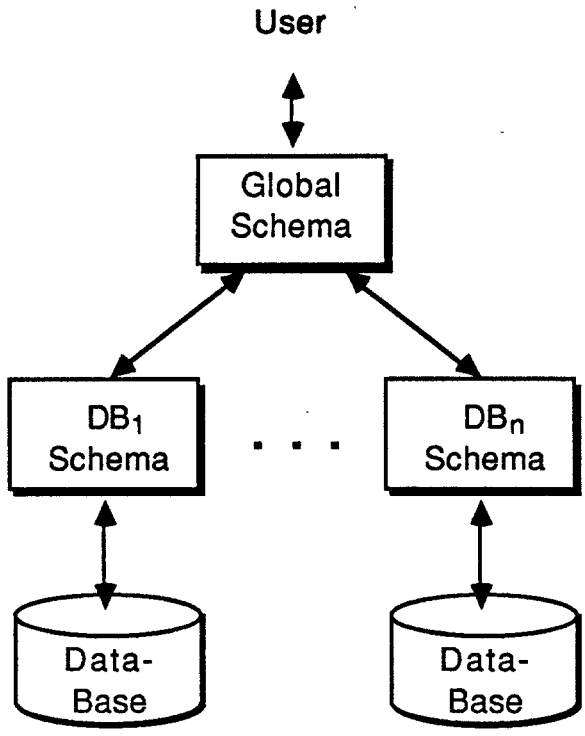


Figure 5.2: A Global Schema between the Users and the Databases.

- Domains may have different structures, e.g., date may be mmddyy or ddmmyy
- Domains that represent the same data may have different names
- Records that represent the same data may have different names

- Difficult problems:
  - The same fact is modeled by different record structures
  - Different constraints may apply to the same fact

- Very difficult problems:
  - Conflicting models of similar facts
  - Conflicting constraints applied to similar facts

If the databases to be integrated are heterogeneous, i.e., their schemata are defined in terms of different data models, then one encounters:

- Very difficult problems:
  - Defining mappings between data structures in different data models
  - Defining mappings between Data Manipulation Languages of different data models

It is noted that it is not the notion of a global schema as such that creates problems in database integration. The problems stem from the requirement that the global schema be designed from the very outset of the integration; and worse, that the global schema is thought of as the *union*–without redundancy and internal

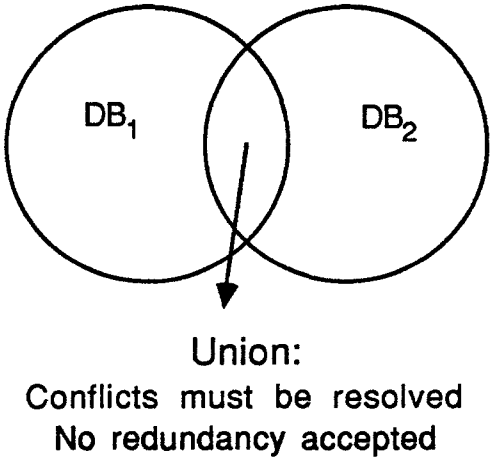


Figure 5.3: Database Integration.

conflicts–of the schemata of the existing databases. This situation is illustrated in Figure 5.3.

The notion of database interoperability can be distinguished from database integration by considering the illustration in Figure 5.4.

The basic idea is to let the initial global schema be the *concatenation* of the schemata of the existing databases. In other words, The global schema still consists initially of all the schemata of the existing databases with all the redundancy and all the conflicts this implies.

In addition, a rule set is constructed for each separate database called update and retrieval dependencies, which controls inter-database consistancy through inter-database operation calls. This rule set is used here to enforce the functionality of the integrated MRP II/CAD system.

5.2 Update Dependencies

A relation R is said to be update dependent on relation S if there exists an update on relation R that succeeds only if one or more implied updates on relation S succeed.

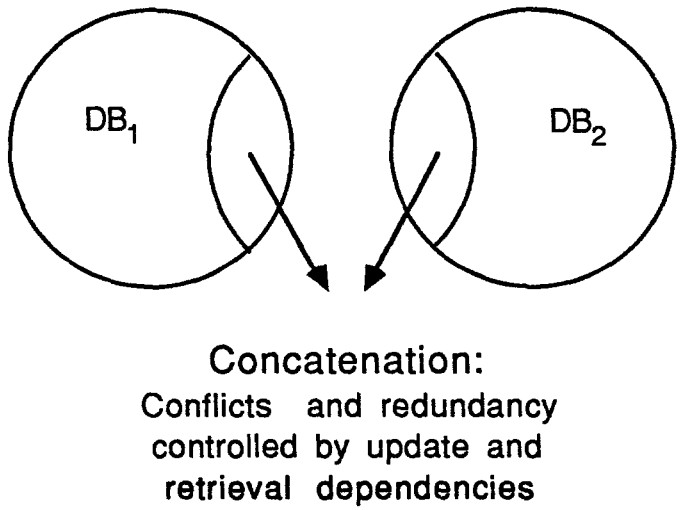


Figure 5.4: Database Interoperability.

Similarly, a relation R is retrieval dependent on relation S if there exists a retrieval from relation R that succeeds only if one or more implied retrievals from relation S succeed.

The structure of update and retrieval dependencies is demonstrated in the following example:

$op_1(R)$
$\rightarrow$ $cond_1$,
$op_2(S)$,
$op_3(T)$,
$op_4(R)$.
$\rightarrow$ $cond_2$,
$op_5(S)$,
$op_6(U)$,
$op_7(R)$.

The meaning of this is as follows: operation op_1 on relation R is said to succeed if and only if for at least one of the alternatives in its update dependency, the condition evaluates to true and all the implied operations succeed. It fails otherwise. For the first alternative in the above example to succeed, for instance, the condition, "$cond_1$," must evaluate to true, and the operation op_2 on S, the operation op_3 on

T, and the opertation op_4 on R must all succeed. The operations on the right hand side may be primitive operations or they may themselves be specified as above.

The relations R, S, T, and U may reside in the same database or in different databases. If all the relations reside in the same database, then the update dependencies merely give an operational specification of a set of constraints in that database. If, on the other hand, the relations reside in different databases, then the update dependencies give an operational specification of a set of inter-database constraints.

Communication is established through inter-database operation calls. The only data passed between databases are the actual parameters of the operation calls. Finally, messages about the success or failure of an operation are passed from the site where the operation was executed to the site where the call of it originated.

In the following sections, the syntax and semantics of update dependencies are more formally presented.

5.2.1 Syntax

A compound update operation is defined by an update dependency with the following form:

$$
\begin{array}{ll}
 & < op > \\
\rightarrow & < c_1 >, \\
 & < op_{1,1} >, \\
 & < op_{1,2} >, \\
 & \vdots \\
 & < op_{1,n1} >. \\
\rightarrow & < c_2 >, \\
 & < op_{2,1} >, \\
 & < op_{2,2} >, \\
 & \vdots \\
 & < op_{2,n2} >. \\
\rightarrow & \vdots
\end{array}
$$

where $< op >$ is the compound update operation being defined, $< op_{i,j} >$ is either an implied compound update operation or an implied primitive operation, and $< c_i >$ is a condition on the database state.

A compound update operation $< op_i >$ has the following form:

- <operation name>(<relation name>(<tuple spec>))

where the <tuple spec> is a tuple variable for the relation with the name <relation name> and consists of a list of <domain variable>s. The <tuple spec> in <op> is the formal parameter for <op>. All the <domain variable>s in the <tuple spec> of <op> are assumed to be universally quantified. All <domain variable>s in the <tuple spec>s of $< op_{i,j} >$, that are not bound to a universally quantified <domain variable> in <op>, are assumed to be existentially quantified. All <domain variable>s are in upper case; nothing else is.

The implied primitive operators are: 'add' for adding a new tuple in a relation, 'remove' for eliminating one, 'write' and 'read' for retrieving data by and from the user, 'new' for creating a unique new surrogate, and 'break' for temporarily stopping the system to do some retrieval before giving the control back to the system. The implied primitive operations $< op_{i,j} >$ have the following forms:

- add(<relation name>(<tuple spec>))
- remove(<relation name>(<tuple spec>))
- write('<any text>'), or write(<domain variable>)
- read(<domain variable>)
- new(<relation name>(<tuple spec>))

- break

The <relation name> used in the operation 'new' must be the name of a unary relation defined over a non-lexical domain. The conditions <cond> are expressions of predicates. The connectives used in forming the expressions are 'and' ($\wedge$) and 'not' ($\tilde{}$). The predicates are of the form <relation name>(<tuple spec>) to determine whether or not a given tuple is in a given relation; or of the form 'nonvar(X)' or 'var(X)' to decide whether or not a <domain variable>, X, has been instantiated; or of the form X <comp> Y, where <comp> is a comparison operator.

Conditions, or retrieval dependencies, can also be used to retrieve data from the system.

5.2.2 Semantics

A compound update operation succeeds if, for at least one of the alternatives in its update dependency, the condition evaluates to true and all the implied operations succeed; otherwise it fails.

When a compound update operation is invoked, its formal parameters are bound to the actual parameters. The scope of a variable is one update dependency. Existentially quantified variables are bound to values selected by the database system or to values supplied by the interacting user on request from the database system. Evaluation of conditions, replacement of implied compound update operations, and execution of implied primitive operations is left-to-right and depth-first for each invoked update dependency. For the evaluation of conditions we assume a closed world interpretation.

The non-deterministic choice of a replacement for an implied compound update operation is done by backtracking, selecting in order of appearance the update dependencies with matching left-hand sides. If no match is found, the operation

fails.

An implied compound update operation matches the left-hand side of an update dependency if:

- the operation names are the same, and
- the relation names are the same, and
- all the domain components match. Domain components match if they are the same constant or if one or both of them is a variable. If a variable matches a constant it is instantiated to that value. If two variables match they share value.

The semantics of the primitive operations are:

- add($r(t)$); its effect is $r := r \cup \{t\}$; it always succeeds; all components of 't' are constants.
- remove($r(t)$); its effect is $r := r \backslash \{t\}$ where all components of 't' are constants. It always succeeds.
- write('text'); it writes the 'text' on the user's screen. It always succeeds.
- write(X); writes the value of 'X' on the user's screen. It always succeeds.
- read(X); reads the value supplied by the user and binds it to 'X'. It always succeeds (if the user answers).
- new($r(D)$); produces a new unique surrogate, from the non-lexical domain over which 'r' is defined and binds the value of the variable 'D' to this surrogate. It always succeeds.

- break; suspends the current execution and makes a new copy of the interpreter available to the user, who can use it to retrieve the information he needs to answer a question from an operation.

The list of primitive operations is minimal for illustrating the concept. It can easily be extended. It is emphasized that primitive operations are not available to the user; he cannot directly invoke them.

The execution of 'add' and 'remove' operations done by the system in an attempt to make a compound update operation succeed, will be undone in reverse order during backtracking. This implies, that a (user invoked) compound update operation that fails will leave the database unchanged.

5.2.3 Implementation Strategy

The strategy for implementing the model integrated MRP II/CAD system has been planned to allow for the testing of the specification of the functional relationships between MRP II and CAD early in the project.

The first step has therefore been to define the Update Dependency Language, a formal language for specifying this functional relationship. The language allows for the specification of operations in and between the MRP II and CAD applications in the form of an Artificial Intelligence production system.

The second step of the implementation strategy has been to implement an interpreter for the Update Dependencies Language. A first version of the interpreter has been implemented in Prolog. Both the interpreter and the specification of the functional relationship between the MRP II and CAD aplications under one instance of the interpreter have been tested. Chapter 6 discusses the results of this testing.

The third step is to integrate a remote procedure call facility into the interpreter. This will allow for the running of functional copies of the MRP and the CAD system

under separate instances of the interpreter on the same machine.

The fourth step is to move the two interpreters to different machines by generalizing the remote procedure call facility to allow calls over a network.

An important aspect of this implementation strategy is that it allows early testing of the specification of the functional relationship between the MRP II and the CAD system. Furthermore, step three and four should not imply any changes in this specification, i.e. the distribution of information in the system should be transparent to the user.

Whereas this implementation strategy does allow us to test the specification of the functional dependencies within each system and between the two systems, it does not provide an integration of two actual systems.

Though it is clearly desirable to integrate existing pieces of software rather than developing new systems from scratch, it is not the purpose of this research to provide a "bridge-box" that will allow users to hook up any two particular MRP II and CAD systems. What is provided is a functional description of how such a "bridge-box" for two given systems could be specified. Update Dependencies can then be used to implement it.

To actually build a "bridge-box" for two given systems, one would proceed as follows. First, the set of operations available to the users in each system should be identified. These operations should continue to be available to the users and the user interface should be kept unchanged to the extent possible. Second, The software procedures that supports these operations should be identified. Third, through calls to the interpreter, one would insert a set of update dependencies between the operations available to the users and the software procedures supporting them. These update dependencies would capture calls of operations made by the users and would issue calls of the software procedures supporting them, i.e. the calls of the

software components would be implied operations in the update dependencies. This approach would allows the enforcement of consistency both within and between the two systems, by reusing the software components already available.

It is important to realize that the above approach keeps the user interfaces stable, reuses the software components that are already there, avoids redesign of existing databases, and solves the problem. However, it is also important to realize, that a certain amount of additional programming cannot be avoided.

Chapter 6

Demonstration and Discussion of Results

In Chapter 4, the functional description of the model integrated MRP II/CAD system was presented. To describe the model, individual operations were proposed to handle the tasks associated with particular part and product structure maintenance. With the set of operations thus defined, part master data and bills of material information can be exchanged between the two systems to maintain consistency between them during normal design and manufacturing activities.

As mentioned earlier, the goal of this research is to develop the generic functionality for an integrated MRP II/CAD system; accordingly, the model will not be demonstrated by the actual integration of any two particular MRP II and CAD systems. Instead, using the Update Dependency Language presented in Chapter 5, it is possible to both demonstrate and test the specification of the functional relationships between MRP II and CAD in the model.

6.1 Implementing the Model System Using Update Dependencies

The Update Dependencies Language provides a convenient formalism for organizing, expressing, and communicating algorithms to implement an integrated system such

as the one described in this work. Because it uses a declarative representation, algorithms can be expressed in a natural manner, as a concise statement of the designer's notions. Hence the translation from functional design to program code is fairly straightforward and direct, reducing the tedious and error-prone phase of translating algorithms into code.

Further, the Update Dependency Language allows for the definition of a system as a modular series of operations, which can be easily modified or expanded to improve the capabilities of the system. Each of these operations consists of a series of rules to be applied in particular circumstances to achieve a particular goal. This representation is particularly well suited for the MRP II/CAD integrated system, which has likewise been presented in this manner.

Using the Update Dependency Language, the operations described in Chapter 4 have been coded; the resulting code is presented in Appendix A. As described in section 5.2.3, these operations act within one instance of the Update Dependency interpreter. This does not provide true "interoperability" between two systems; instead, two sets of data, one for MRP II and one for CAD, are maintained within the single database under the control of the interpreter. Interoperability is simulated by treating the operations and data sets as if they were in separate databases. As the Update Dependency interpreter evolves to include remote procedure calls, true interoperability will be possible with little or no change to the current operation definitions.

6.1.1 Relations Used by the System

All of the data maintained in the interoperability system are recorded in sets of relations, one set representing the CAD database, and another representing the MRP II database. In the CAD data set, the following relations are defined, following

the data formats specified in chapter 4:

cadpart. Records part master data by part number. Each part in the CAD database has a cadpart record. Fields include:

- Part Number
- Drawing Number
- Drawing Size
- Description
- BOM Unit of Measure
- Supersedes Part Number
- Superseded by Part Number

cadrev. Records information about the various revisions of a part. Each revision of each part in the CAD database has a cadrev record. Included in this record are:

- Part Number
- Revision Level
- Effectivity Start Date
- Effectivity End Date
- CAD Status Code
- Drawing File Name

latestcadrev. Itentifies the most recent version of each part in the CAD database. There is one record for each part, which is updated as each new revision is released in CAD. This relation contains:

- Part Number
- Revision Level of most recent version

cadcomponent. Records component relationship information for the parts in the CAD database. There is one cadcomponent record for each relationship; an assembly part will therefore have as many cadcomponent records as it has components. The information in this record consists of:

- Parent Part Number
- Parent Revision Number
- Item Number
- Component Part Number
- Quantity per Assembly

Similar records are defined for the MRP II data set; these include:

mrppmr. Contains the part master record for a part. Each part in the MRP II database has one of these records. Its fields include:

- Part Number
- Drawing Number
- Drawing Size
- Description
- BOM Unit of Measure
- Purchasing/Inventory Unit of Measure
- Unit of Measure Conversion Factor

- Source Code
- Standard Cost
- Leadtime
- Supersedes Part Number
- Superseded by Part Number

mrprev. Contains revision-specific information. An mrprev record is produced for each version of each part in the MRP II database, containing:

- Part Number
- Revision Level
- Effectivity Start Date
- Effectivity End Date
- MRP II Status Code

latestmrprev. Identifies the most recent version of each part in MRP II. A single relation is maintained for each part, and is updated as new revision levels are released in MRP II. This relation contains:

- Part Number
- Revision Level of most recent version

mrpcomponent. Records relationship information for assemblies. One mrpcomponent record is maintained for each relationship, such that each assembly in the MRP II database will have as many relationship records as it has components. The fields in this record include:

- Parent Part Number

- Parent Revision Level
- Item Number
- Component Part Number
- Quantity per Assembly

In addition to these primary records, there is a series of secondary records used on a temporary basis to alert MRP II users of activities requiring their attention, such as the release of a new part or revision. These include:

newpmr. Notifies MRP II users that a new part master record has been established by the release of a part from CAD, allowing the MRP II users to begin planning for its procurement. The record is deleted when the first revision of the new part is released in MRP II. Its only field is:

- Part Number

newrev. Signifies that a new version of a part has been released by CAD, allowing MRP II users to determine an effectivity starting date and to plan for any changes required by the transition to the new revision. The record is deleted when the new version is released to MRP II. The structure of the record includes the following fields:

- Part Number
- Revision Level

rereleased. Used to notify MRP II users that a part/revision formerly placed on hold by CAD users (which would have invoked an MRP II hold as well) has

been rereleased to CAD without a design change. The revision is not automatically rereleased to MRP II, as there may still be manufacturing or vendor problems that MRP II users would want to explore before rereleasing the part/revision. This relation is deleted when the part/revision is rereleased to MRP II. The relation contains:

- Part Number
- Revision Level

obsolete. This record serves to alert MRP II users that a particular part and revision level has been given an "obsolete" status in CAD, and that the phase-out of the part and revision level should be planned. The record is maintained until the part is deleted from MRP II. It consists of:

- Part Number
- Revision Level

Finally, there are two relations used solely to represent MRP II related activities involving parts. The system provides for the insersion and deletion of these records with minimal consistency checks, and uses their presence or absence for determining whether parts may be deleted from the system. These records include:

inventory. Used to represent the quantity of a part in inventory. Following the approach used in most commercial MRP II systems, inventory records are maintained by part number only, not by the specific revision level. The system thus assumes that the quantity relates to the active revision. The relation consists of:

- Part Number

- Quantity in Inventory

onorder. This relation indicates an on-order quantity of a specific part. Again, the record uses only the part number; the active revision level is implied. The fields in the relation are:

- Part Number
- Quantity on order
- Order ID number

Due to the typical time lag between CAD and MRP II activities, the data in equivalent CAD and MRP II relations may be different at times. For example, new revisions created via CAD are not transferred to MRP II until they are released by CAD users. Similarly, a part's latestcadrev record is updated as soon as a new revision is releasd from CAD; its latestmrprev record is not updated until the revision is released from MRP II. In these examples and in many others in the model, the database conflicts are temporary, being resolved over the natural course of design and manufacturing activities.

6.1.2 Programing the Interoperability System

The integrated system is comprised of operations for inserting, deleting, and modifying the relations. Each operation consists of a list of alternative actions, each appropriate for a particular database state. Each alternative begins with a list of conditions which must be satisfied prior to performing any of the actions. These conditions are thus used to channel control into the proper set of actions under a given set of circumstances.

The list of actions to be performed as a result of calling an operation and satisfying a particular set of conditions often includes calling other operations on other

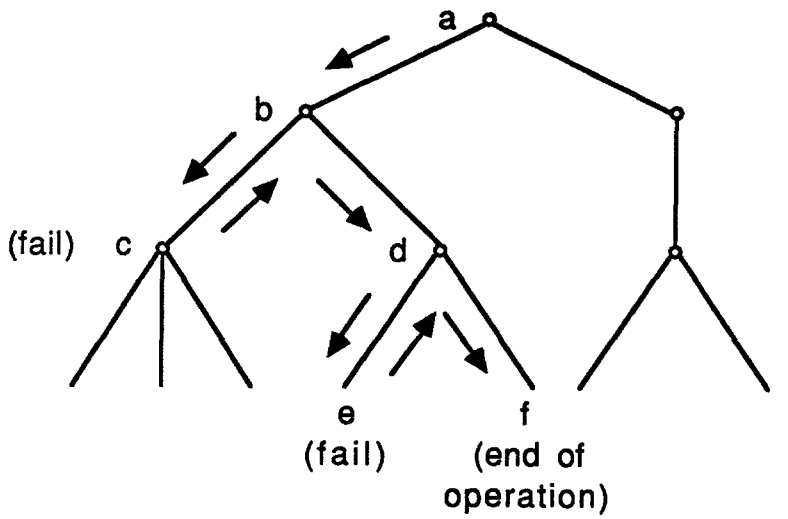


Figure 6.1: A Demonstration of Backtracking.

relations, either in the same database, or in another database. These operations may again involve calling other operations, and so on. In order for the original operation to succeed, all of the operations in the resulting "chain" must also succeed. If any one of them fails, and no alternatives are found by backtracking, then the original operation fails, and no change to either database occurs. Thus with a well specified system, it is possible to protect the integrity of the databases.

6.1.2.1 Backtracking

Backtracking is used by the Update Dependency Language to search out and attempt to satisfy as many of the possible solutions to an operation as necessary to either obtain a successful solution or exhaust all possible solutions. The concept of backtracking is demonstrated with the tree structure in Figure 6.1. Each node in the tree represents an operation, which is comprised of several alternatives, represented by the branches descending from it. Many of these alternatives, in turn, call other operations. Each level of the tree then can be viewed as a depth in a chain of operations. Starting at the top of the tree, the system begins down a particular

path. If any branch in the path fails, control is returned to the node above the failed branch, and another branch is found. Again progress is made in the descending (or forward) direction until either the bottom of the tree is successfully reached, or another failure occurs. If no new branches from a particular node are available, the system returns to the node one level higher on the same path, and again looks for an alternate path. The operation is completed when either the endpoint of a path is successfully reached, or the original (top) node is reached after backtracking and all of its descending branches have been attempted. In the first case, the operation succeeds, while in the second, it fails.

Several of the nodes in Figure 6.1 have been labeled to demonstrate a backtracking sequence. The complete path taken is indicated by the letters $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow f$. The search begins at node a, the top-level operation. The first path searched is toward node b, meaning that if operation b succeeds, operation a will succeed as well. Operation b will succeed if either operation c or d succeed. Operation c, however, fails in this example, without trying its implied operations. Thus backtracking occurs to operation b, and the search continues to operation d. Operation d can succeed only if either operation e or operation f succeeds. The first path found, operation e, fails, initiates backtracking to d, and then operation f is attempted. Because operation f is at the bottom of the tree and it succeeds, the original operation, a, as well as operations b and d succeed as well.

Backtracking is an important feature of the Update Dependency Language, as it ensures that all possible solutions to an operation will be attempted before the operation is said to fail. However, backtracking occasionally has undesirable effects, as discussed in the following sections.

6.1.2.2 "Failure Alternatives"

To detect particular actions that preclude the success of an operation, such as an attempt to insert a part that already exists in the database, it is necessary to specify "failure alternatives," i.e., alternatives whose conditions are met only if there is an inconsistency in the user's attempted operation. In such cases, a message is printed out and the operation is made to fail, preventing changes to either database.

The need for "failure alternatives" elicits several comments on the Update Dependency Language. First, if it were not for the desire to alert the user to errors by describing the inconsistency, there would be no need for such alternatives. Only sucessful alternatives would need to be specified, and operations would automatically fail whenever none of the alternatives could succeed. The amount of code required to implement an operation would be greatly reduced, at the expense of making it much more difficult for the user to catch errors and diagnose failures.

The second comment relates to the need to make the "failure alternatives" fail. It would seem logical that if all of the conditions of one of these alternatives were satisfied, and if its only action were to print out a message, then it should succeed and end the operation. This approach would work if the user were directly invoking the operation in which the "failure alternative" was located; the operation would "succeed," but no changes to any database would be made. If, however, the operation in which the "failure alternative" was located were called as an implied operation of another operation, a "successful" "failure alternative" would indicate the success of the implied operation to the calling operation, which would proceed to satisfy any further operations without recognizing the constraint violation found in the implied operation. By making the "failure alternative" fail, however, the system is forced to search for alternate solutions, and if none are found, the operation

with the "failure alternative" will fail, forcing the operation that called it to seek alternative solutions, and so on. The entire chain of operations will then fail if a successful solution is not found, and the database will remain unchanged.

6.1.2.3 System Performance

The next several comments involve the effect of the "failure alternatives" on the performance of the system. Simply because they greatly increase the number of alternatives defining an operation, the performance of the system is hindered somewhat. A more significant effect however, is the increase in backtracking they cause. As mentioned above, each "failure alternative" is made to fail after all of its conditions have been satisfied and it has printed an appropriate message. Because all of its conditions have been satisfied at this point, control of the system has been channelled into the proper set of actions for the specific set of circumstances. Functionally, there is no need to seek alternatives. However, when it is forced to fail, the interpreter responds as if one of the conditions or implied operations had failed, and it seeks other alternatives, as explained in section 6.1.2.1.

Though in general backtracking is desirable, it produces a number of undesirable side effects when combined with "failure alternatives." The first is simply the extra time involved. In the design of this particular system, the alternatives are specified as mutually exclusive cases; it is impossible for an operation to succeed if it satisfies the conditions of a "failure alternative." In spite of this, the interpreter searches for a successful solution, which it cannot find, delaying the eventual failure of the operation.

The second side effect is more troublesome. When a new branch is initiated by attempting to resatisfy an implied operation after the conditions of a "failure alternative" have been satisfied, the resulting backtracking is often both confusing

and unnecessary. For example, when a part is being deleted from the system, all of its revision records must be deleted. If any one of the revision records has a "released" status in CAD, the deletion fails. The system may find the revision with "released" status after marking several of the other revisions for deletion. When the "released" revision is found, a "failure alternative" causes a failure and initiates backtracking. During backtracking, the order in which the system finds the revisions becomes significant; each different sequence represents a different possible solution, or branch, for the operation to follow. If a part has two revisions, for instance, one of which has a "released" status, and the other an "obsolete" status, the system can either successfully mark the obsolete revision for deletion first, and then find the released revision and fail, or it can find the released revision first and fail without finding the obsolete revision. The effect of the backtracking is that both of these sequences are attempted, despite the fact that neither can succeed. As the number of revisions increases, so does the number of possible sequences, and the time consumed by backtracking through each of these. Also, since messages are printed out during such operations to indicate that particular actions have been performed and describing errors that are detected, it is very confusing for the user to see the same error message appear several times prior to the final failure of an operation.

Backtracking after a "failure alternative" may also cause a similar phenomenon if an operation is specified such that the conditions of more than one alternative are satisfied during a particular call to the operation. Again, the sequence in which the interpreter attempts to satisfy alternatives is significant; during backtracking, each possible sequence is attempted, whether it provides a true alternative path or just a permutation of the initial path. Eliminating this behavior, however, is fairly simple; operations should be specified such that the conditions of only one

alternative should be satisfied during each call to the operation, unless more than one true alternative is possible.

6.1.3 Problems with the Current Interoperability Model

6.1.3.1 Duplication of Application Functions

Due to the early stage of the development of the Update Dependency interpreter, and the desire to remain as general as possible in the solution to MRP II/CAD integration, the model is coded to function only as an isolated system. Instead of interacting with commercial MRP II and CAD interfaces, the user must interact directly with the interoperability system. Though this is sufficient to demonstrate the specification of the functional relationships between MRP II and CAD applications, it complicates the development and use of the system. In addition to providing functions for physically transferring information between MRP II and CAD, programming the model as an independent system means incorporating functions that should be inherent in the applications themselves, particularly in MRP II, with no need to reproduce them. Functions such as mass substitution of components in bills of material and copying bills of material from one assembly to another, for example, are provided in typical MRP II systems, but have also been programmed into the model to provide the user with the ability to perform transactions which exercise particular portions of the interoperability system.

6.1.3.2 Operation "Chains"

A problem with including such activities as interoperability functions is that invoking them often results in very long chains of operations that must succeed before the original operation can succeed; in some cases, the chains involve combinations of activities that the interpreter cannot satisfy. In contrast, if actual application sys-

tems were interfaced with the interoperability system, the chain of activities could often be broken down into smaller chains that could be performed independently by separate calls from the applications to the lower level interoperability functions.

For example, the current implementation of the product structure operations is intended to allow a user to make a part number or revision level change as part of an operation. If he or she chooses to do so, the resulting chain of activities includes the insersion of the new part or revision level, the copying of the bill of material from the previous part and revision level, and then performing the desired operation on the new bill of material. If augmented by the application systems, this process could be broken down into a separate chain for each of these activities.

With the current version of the Update Dependency interpreter and the current implementation of the model, the full chain can only succeed if the operation on the new bill of material is a component addition. This is due to the fact that the interpreter does not allow changes to or deletions of relations inserted in the same operation. If the operations could be broken down as described above, this problem would not occur, since the insersion of new relations would be performed as a separate operation prior to the changes or deletions to be made to them. In the meantime, the operations are being left in their current state, with the recognition that exercising the feature of new part or revision creation during a product structure operation will not be fully successful.

For operations involving only one assembly, such as adding or deleting a component, the user should insert a new part or revision if required, prior to performing the assembly operation, which will eliminate the problem. For operations that involve several assemblies, notably the mass substitution of a component by another, this is more complicated, since the particular assemblies involved may not be known by the user, making it difficult to make changes to them before the operation is

invoked.

6.1.3.3 User Interface

Another result of the current implementation of the model is its poor user interface. To invoke any of the operations in the system, the user must, as a minimum, type the operation name as well as the relation it operates on. The system will then request from the user the minimum information it requires to process the transaction. If the user wishes to enter any values beyond those required, he or she must enter the operation, the relation, and the optional values; the required values may also be specified, or may be left as variables. This arrangement is used as a compromise between flexiblity and ease of use, to avoid having the system ask the user for a long series of values that may be left unknown. However, it must be remembered that once the interpreter is capable of interacting between two actual systems, the interoperability system can become completely transparent to the user; as described in section 5.2.3, the calls to the Update Dependency interpreter will be positioned between the operations available to the users and the software procedures supporting them. The users will interact with the applications' user interfaces, which will, in turn, be internally interfaced with the interoperability system. Hence the interoperability system need not be designed to provide a commercial-quality user interface directly.

Similarly, the system does not perform validity checks on most of the data fields entered by users. For example, units of measure are not checked against a table of acceptable values; any value is accepted. Though such capabilities could be programmed into the system, it is assumed that the application systems would perform any such checks prior to calling the interoperability functions.

6.2 A Sample Session with the Model System

In the following sections, the basic part master data and product structure maintenance operations of the model MRP II/CAD system are demonstrated. At the beginning of the demonstration, both the CAD and MRP II databases are empty.

6.2.1 Part Master Data

6.2.1.1 Adding New Parts Via CAD

The first demonstration of part master data operations follows the insersion of a part via CAD and its subsequent release from CAD to MRP II.

The first operation, **insert(cadpart)**, is used to enter both cadpart and cadrev records for the first version of the new part into the CAD database. The subsequent listing of part records in both the CAD and MRP II databases verifies that the information is stored in CAD only at this stage, and that the first version of the part has a working status. Accordingly, there is no latestcadrev record. Because no optional fields were given values in the call to the operation, the system asks only for those fields required, and the remaining fields are assigned the value "unknown."

```
ud> insert(cadpart).

Part Number? 123.
Description? cad_part_1.
Unit of Measure? each.
New Revision Level? 1.
Drawing File Name? file_1.
Revision has been added to CAD
Part has been added to CAD


ud> listing(cadpart).

cadpart(123,unknown,unknown,cad_part_1,each,unknown,unknown).
```

```
ud> listing(cadrev).

cadrev(123,1,unknown,unknown,w,file_1).

ud> listing(latestcadrev).

ud> listing(mrppmr).

ud> listing(mrprev).
```

The next step is the release of the first version of the part by CAD users using **releasework(cadrev)**. Again, the listings following the call to the operation show the results of its actions. The first version of the new part is given a released status in CAD, a latestcadrev record is inserted in the CAD database, and the part information is transferred to MRP II where skeletal mrppmr and mrprev records are created to describe the first version of the new part to MRP II users. To notify MRP II users that the new part and revision records have been inserted, newpmr and newrev records are also inserted into the MRP II database. Fields not maintained by CAD in the cadpart record are assigned the value "unknown" in the mrppmr record. The status of this version is set to "hold" to allow MRP II users to complete the fields required for the release of the part in MRP II. A latestmrprev record is not established until the part is released in MRP II.

```
ud> releasework(cadrev).

Part Number? 123.
Revision Level? 1.
Part has been added to MRP II
Revision has been added to MRP II
```

```
Revision has been released in CAD

ud> listing(cadpart).

cadpart(123,unknown,unknown,cad_part_1,each,unknown,unknown).

ud> listing(cadrev).

cadrev(123,1,unknown,unknown,r,file_1).

ud> listing(latestcadrev).

latestcadrev(123,1).

ud> listing(mrppmr).

mrppmr(123,unknown,unknown,cad_part_1,each,unknown,unknown,
       unknown,unknown,unknown,unknown,unknown).

ud> listing(mrprev).

mrprev(123,1,unknown,unknown,h).

ud> listing(latestmrprev).

ud> listing(newpmr).

newpmr(123).

ud> listing(newrev).

newrev(123,1).
```

The additional part master record fields required by MRP II may be entered prior to the MRP II release of the version using the **modify(mrppmr)** operation.

Alternatively, the user can simply invoke the **releasehold(mrprev)** operation to release the part, which will check and ask the user for any required fields that still have the value "unknown." This approach is used in in the demonstration. The listings after this final operation show the first version of the part with a released status in both CAD and MRP II. It also shows the effectivity starting date assigned by MRP II users in the revision records of both databases. Note that some of the optional fields in both the part and revision records of both application systems remain unknown. Finally, it is shown that the newpmr and newrev records have been deleted from the MRP II database, and a latestmrprev record has been inserted for the new part.

```
ud> releasehold(mrprev).

Part Number? 123.
Revision Level? 1.
Effectivity Start Date? 7/01/87.
Purchasing/Inventory Unit of Measure? each.
Unit of Measure Conversion Factor? 1.
Source Code? m.
Lead Time? 4.
Revision has been released in MRP II


ud> listing(cadpart).

cadpart(123,unknown,unknown,cad_part_1,each,unknown,unknown).


ud> listing(cadrev).

cadrev(123,1,7/1/87,unknown,r,file_1).


ud> listing(latestcadrev).

latestcadrev(123,1).
```

```
ud> listing(mrppmr).

mrppmr(123,unknown,unknown,cad_part_1,each,each,1,m,
       unknown,4,unknown,unknown).


ud> listing(mrprev).

mrprev(123,1,7/1/87,unknown,r).

ud> listing(latestmrprev).

latestmrprev(123,1).


ud> listing(newpmr).


ud> listing(newrev).
```

6.2.1.2 Adding New Parts Via MRP II

The next scenario demonstrated is the insersion of a new part via MRP II. Using the **insert(mrppmr)** operation, the user is prompted for the fields required to specify the first version of the new part to MRP II. The user is given the option of establishing the part with either a hold or released MRP II status, as appropriate. The results of this operation are shown in the subsequent listings. The part master and revsion records, mrppmr and mrprev, have been created in the MRP II database, as has a latestmrprev record. Data from these records have also been used to establish cadpart, cadrev, and latestcadrev records in the CAD database. Because parts entered via MRP II are assumed to have no drawings and to be primarily under the control of MRP II users, the three fields in the cadpart and cadrev records related to the drawing are assigned the value "inapplicable." Further, the

status of the part version in CAD is set directly to released, regardless of the status entered in MRP II. This is to avoid the need for any CAD action to make the part active.

```
ud> insert(mrppmr).

Part Number? 234.
Description? mrp_part_1.
BOM Unit of Measure? each.
Purchasing Unit of Measure? each.
Unit of Measure Conversion Factor? 1.
Source Code? b.
Lead Time? 7.
Part has been added to CAD
Revision Level? 1.
Effectivity Start Date? 7/15/87.
Status Code? r.
Revision has been added to CAD
Revision has been added to MRP II
Part has been added to MRP II


ud> listing(mrppmr).

mrppmr(123,unknown,unknown,cad_part_1,each,each,1,m,
       unknown,4,unknown,unknown).
mrppmr(234,inapp,inapp,mrp_part_1,each,each,1,b,
       unknown,7,unknown,unknown).


ud> listing(mrprev).

mrprev(123,1,7/1/87,unknown,r).
mrprev(234,1,7/15/87,unknown,r).


ud> listing(latestmrprev).

latestmrprev(123,1).
latestmrprev(234,1).
```

```
ud> listing(cadpart).

cadpart(123,unknown,unknown,cad_part_1,each,unknown,unknown).
cadpart(234,inapp,inapp,mrp_part_1,each,unknown,unknown).


ud> listing(cadrev).

cadrev(123,1,7/1/87,unknown,r,file_1).
cadrev(234,1,7/15/87,unknown,r,inapp).


ud> listing(latestcadrev).

latestcadrev(123,1).
latestcadrev(234,1).
```

6.2.1.3 Adding New Part Revisions Via CAD

The next set of operations demonstrated is the insersion of a new version of a part via CAD. The session begins with the **insert(cadrev)** operation, which the user uses to enter data about a revision to the part created in section 6.2.1.1. A new cadrev record for the revision is created with a working status; as shown in the subsequent listings, no information about the revision is transferred to MRP II at this stage, nor is the latestcadrev updated to reflect the new revision.

```
ud> insert(cadrev).

Part Number? 123.
New Revision Level? 2.
Drawing File Name? file_2.
Revision has been added to CAD


ud> listing(cadrev).

cadrev(123,1,7/1/87,unknown,r,file_1).
cadrev(234,1,7/15/87,unknown,r,inapp).
```

```
cadrev(123,2,unknown,unknown,w,file_2).
```

```
ud> listing(latestcadrev).

latestcadrev(123,1).
latestcadrev(234,1).
```

```
ud> listing(mrprev).

mrprev(123,1,7/1/87,unknown,r).
mrprev(234,1,7/15/87,unknown,r).
```

The next operation shown is the release of the new revision, using **releasework(cadrev)**. This operation makes the previous revision obsolete, releases the new revision, and updates the latestcadrev record for the part. It also triggers the establishment of an mrprev record for the new version in MRP II; this record is given a hold status, as shown in the listings following the operation. A newrev record is also created in MRP II to alert users to the new revision level. The latestmrprev record for the part is not yet updated, however.

```
ud> releasework(cadrev).

Part Number? 123.
Revision Level? 2.
Revision has been given obsolete status in CAD
Revision has been added to MRP II
Revision has been released in CAD
```

```
ud> listing(cadrev).

cadrev(234,1,7/15/87,unknown,r,inapp).
cadrev(123,2,unknown,unknown,r,file_2).
cadrev(123,1,7/1/87,unknown,o,file_1).
```

```
ud> listing(latestcadrev).

latestcadrev(234,1).
latestcadrev(123,2).


ud> listing(mrprev).

mrprev(123,1,7/1/87,unknown,r).
mrprev(234,1,7/15/87,unknown,r).
mrprev(123,2,unknown,unknown,h).


ud> listing(latestmrprev).

latestmrprev(123,1).
latestmrprev(234,1).


ud> listing(newrev).

newrev(123,2).
```

The final operation in this sequence is the release of the new revision in MRP II, using the operation **releasehold(mrprev)**. The effectivity start date for the new revision is entered, and the revision is released. The listings after the operation show the transfer of the effectivity starting date of the new revision to the effectivity end date of the previous revsion, both in MRP II and CAD. Also, the newrev record has been deleted, and the latestmrprev record has been updated.

```
ud> releasehold(mrprev).

Part Number? 123.
Revision Level? 2.
Effectivity Start Date? 9/01/87.
Revision has been released in MRP II
```

```
ud> listing(cadrev).

cadrev(234,1,7/15/87,unknown,r,inapp).
cadrev(123,2,9/1/87,unknown,r,file_2).
cadrev(123,1,7/1/87,9/1/87,o,file_1).


ud> listing(latestcadrev).

latestcadrev(234,1).
latestcadrev(123,2).


ud> listing(mrprev).

mrprev(234,1,7/15/87,unknown,r).
mrprev(123,2,9/1/87,unknown,r).
mrprev(123,1,7/1/87,9/1/87,r).


ud> listing(latestmrprev).

latestmrprev(234,1).
latestmrprev(123,2).


ud> listing(newrev).

listing(newrev)
```

6.2.1.4 Adding New Part Revisions Via MRP II

Revisions may also be inserted via MRP II, but only if they relate to an MRP II-generated part. In the following session, a second revision level for the part established in section 6.2.1.2.

Only the single operation, **insert(mrprev)**, is necessary to do so. The user is prompted by the system for the necessary input data, including the desired status of the revision; the remaining fields are left as "unknown," since they were not specified

in the call. As shown in the subsequent listings, the data are used to establish an mrprev record and update the latestmrprev record for the part in MRP II; in CAD, a cadrev record with a released status is created, and the latestcadrev record for the part is likewise updated to reflect the new revision. To minimize the involvement of CAD users with such revisions, the previous revision recored for the part is also left as "released." Finally,the effectivity start date of the new revision is used to update the effectivity end date of the previous revsion, in both databases.

```
ud> insert(mrprev).

Part Number? 234.
Revision Level? 2.
Status Code? r.
Effectivity Start Date? 10/1/87.
Revision has been added to CAD
Revision has been added to MRP II


ud> listing(mrprev).

mrprev(123,2,9/1/87,unknown,r).
mrprev(123,1,7/1/87,9/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
mrprev(234,1,7/15/87,10/1/87,r).


ud> listing(latestmrprev).

latestmrprev(123,2).
latestmrprev(234,2).


ud> listing(cadrev).

cadrev(123,2,9/1/87,unknown,r,file_2).
cadrev(123,1,7/1/87,9/1/87,o,file_1).
cadrev(234,2,10/1/87,unknown,r,inapp).
cadrev(234,1,7/15/87,10/1/87,o,inapp).
```

```
ud> listing(latestcadrev).

latestcadrev(123,2).
latestcadrev(234,2).
```

6.2.1.5 Part Supersession

The final part addition, via CAD, demonstrates the processes that result when a new part supersedes another. In this demonstration, the new part will supersede the part entered into CAD in section 6.2.1.1.

Because the "supersedes part number" field is not required by the interoperability system, the user must specify the value in the call to operation **insert(cadpart)**. In this example, a few of the required values are left blank in the call to show that the system will request all of the fields required, regardless of those supplied in the call.

The part and revision records, cadpart and cadrev, for the first version of the new part, are inserted in the same manner as in section 6.2.1.1, using the **insert(cadpart)** operation. The revision record is given a working status, and both records remain local to CAD.

```
ud> insert(cadpart(_,_,_,cad_part_2,_,123,_)).

Part Number? 124.
Unit of Measure? each.
New Revision Level? 1.
Drawing File Name? file_3.
Revision has been added to CAD
Part has been added to CAD


ud> listing(cadpart).
```

```
cadpart(123,unknown,unknown,cad_part_1,each,unknown,unknown).
cadpart(234,inapp,inapp,mrp_part_1,each,unknown,unknown).
cadpart(124,unknown,unknown,cad_part_2,each,123,unknown).


ud> listing(cadrev).

cadrev(123,2,9/1/87,unknown,r,file_2).
cadrev(123,1,7/1/87,9/1/87,o,file_1).
cadrev(234,1,7/15/87,10/1/87,r,inapp).
cadrev(234,2,10/1/87,unknown,r,inapp).
cadrev(124,1,unknown,unknown,w,file_3).


ud> listing(latestcadrev).

latestcadrev(123,2).
latestcadrev(234,2).


ud> listing(mrppmr).

mrppmr(123,unknown,unknown,cad_part_1,each,each,1,m,
       unknown,4,unknown,unknown).
mrppmr(234,inapp,inapp,mrp_part_1,each,each,1,b,
       unknown,7,unknown,unknown).


ud> listing(mrprev).

mrprev(123,2,9/1/87,unknown,r).
mrprev(123,1,7/1/87,9/1/87,r).
mrprev(234,1,7/15/87,10/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
```

The revision record is then released using operation **releasework(cadrev)**; the listings following this operation show the effects of this action. The "superseded by part number" field in the cadpart record of the superseded part is updated to show the supersession. Also, the latest revision of the superseded part is made obsolete in CAD, and the latestcadrev record for the superseded part is deleted. Next, a

latestcadrev record is established for the new part. Finally, the part and revision data are used to establish mrppmr and mrprev records for the new part in the MRP II database. As before, the first version of the new part is given a hold status in MRP II, and therefore no latestmrprev record is created for the part. Note that the part record of the superseded part in MRP II already reflects the supersession.

```
ud> releasework(cadrev).

Part Number? 124.
Revision Level? 1.
Part has been added to MRP II
Revision has been added to MRP II
Part substitution has been completed
Revision has been given obsolete status in CAD
Revision has been released in CAD


ud> listing(cadpart).

cadpart(234,inapp,inapp,mrp_part_1,each,unknown,unknown).
cadpart(124,unknown,unknown,cad_part_2,each,123,unknown).
cadpart(123,unknown,unknown,cad_part_1,each,unknown,124).


ud> listing(cadrev).

cadrev(123,1,7/1/87,9/1/87,o,file_1).
cadrev(234,1,7/15/87,10/1/87,r,inapp).
cadrev(234,2,10/1/87,unknown,r,inapp).
cadrev(123,2,9/1/87,unknown,o,file_2).
cadrev(124,1,unknown,unknown,r,file_3).


ud> listing(latestcadrev).

latestcadrev(234,2).
latestcadrev(124,1).


ud> listing(mrppmr).
```

```
mrppmr(234,inapp,inapp,mrp_part_1,each,each,1,b,
       unknown,7,unknown,unknown).
mrppmr(123,unknown,unknown,cad_part_1,each,each,1,m,
       unknown,4,unknown,124).
mrppmr(124,unknown,unknown,cad_part_2,each,unknown,
       unknown,unknown,unknown,unknown,123,unknown).


ud> listing(mrprev).

mrprev(123,2,9/1/87,unknown,r).
mrprev(123,1,7/1/87,9/1/87,r).
mrprev(234,1,7/15/87,10/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
mrprev(124,1,unknown,unknown,h).


ud> listing(latestmrprev).

latestmrprev(123,2).
latestmrprev(234,2).


ud> listing(newpmr).

newpmr(124).


ud> listing(newrev).

newrev(124,1).
```

The final operation in this scenario is the release of the new part by MRP II users, using operation **releasehold(mrprev)**. At this point, the revision record of the new part is given a released status, a latestmrprev record is created for the new part, and the effectivity start date of the first version is transferred back to the CAD revision record of the new part. The effectivity start date is also used to update the effectivity end date of the latest revision of the superseded part in both CAD and MRP II.

```
ud> releasehold(mrprev).

Part Number? 124.
Revision Level? 1.
Effectivity Start Date? 11/01/87.
Purchasing/Inventory Unit of Measure? each.
Unit of Measure Conversion Factor 1.
Source Code? m.
Lead Time? 5.
Revision has been released in MRP II


ud> listing(cadpart).

cadpart(234,inapp,inapp,mrp_part_1,each,unknown,unknown).
cadpart(124,unknown,unknown,cad_part_2,each,123,unknown).
cadpart(123,unknown,unknown,cad_part_1,each,unknown,124).

ud> listing(cadrev).

cadrev(123,1,7/1/87,9/1/87,o,file_1).
cadrev(234,1,7/15/87,10/1/87,r,inapp).
cadrev(234,2,10/1/87,unknown,r,inapp).
cadrev(123,2,9/1/87,unknown,o,file_2).
cadrev(124,1,11/1/87,unknown,r,file_3).


ud> listing(latestcadrev).

latestcadrev(234,2).
latestcadrev(124,1).


ud> listing(mrppmr).

mrppmr(234,inapp,inapp,mrp_part_1,each,each,1,b,
       unknown,7,unknown,unknown).
mrppmr(123,unknown,unknown,cad_part_1,each,each,1,m,
       unknown,4,unknown,124).
mrppmr(124,unknown,unknown,cad_part_2,each,each,1,m,
       unknown,5,123,unknown).
```

```
ud> listing(mrprev).

mrprev(123,2,9/1/87,unknown,r).
mrprev(123,1,7/1/87,9/1/87,r).
mrprev(234,1,7/15/87,10/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
mrprev(124,1,11/1/87,unknown,r).


ud> listing(latestmrprev).

latestmrprev(123,2).
latestmrprev(234,2).
latestmrprev(124,1).


ud> listing(newpmr).


ud> listing(newrev).
```

6.2.1.6 Hold Functions in CAD and MRP II

The next set of operations demonstrates the hold functions in CAD and MRP II. In the following session, a CAD user invokes the operation **hold(cadrev)** to place a hold on a particular version of a part. The subsequent listings show that both the cadrev record in the CAD database and the mrprev record in the MRP II database have both been given hold status.

```
ud> hold(cadrev).

Part Number? 124.
Revision Level? 1.
Revision has been given hold status in MRP II
Revision has been given hold status in CAD


ud> listing(cadrev).
```

```
cadrev(123,1,7/1/87,9/1/87,o,file_1).
cadrev(234,1,7/15/87,10/1/87,r,inapp).
cadrev(234,2,10/1/87,unknown,r,inapp).
cadrev(123,2,9/1/87,unknown,o,file_2).
cadrev(124,1,11/1/87,unknown,h,file_3).


ud> listing(mrprev).

mrprev(123,2,9/1/87,unknown,r).
mrprev(123,1,7/1/87,9/1/87,r).
mrprev(234,1,7/15/87,10/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
mrprev(124,1,11/1/87,unknown,h).
```

In the next operation, **releasehold(cadrev)**, a CAD user rereleases the part, which would occur if the current design were determined to be satisfactory. As seen in the listings following the operation, the status of the cadrev record is changed from hold to released, but the status of the mrprev record remains hold. To alert MRP II users that the part has been rereleased in CAD, a rereleased record is created.

```
ud> releasehold(mrprev).

Part Number? 124.
Revision Level? 1.
Revision Has Hold Status in CAD--Cannot Be Released
failure


ud> releasehold(cadrev).

Part Number? 124.
Revision Level? 1.
Revision has been rereleased in CAD
```

```
ud> listing(cadrev).

cadrev(123,1,7/1/87,9/1/87,o,file_1).
cadrev(234,1,7/15/87,10/1/87,r,inapp).
cadrev(234,2,10/1/87,unknown,r,inapp).
cadrev(123,2,9/1/87,unknown,o,file_2).
cadrev(124,1,11/1/87,unknown,r,file_3).


ud> listing(mrprev).

mrprev(123,2,9/1/87,unknown,r).
mrprev(123,1,7/1/87,9/1/87,r).
mrprev(234,1,7/15/87,10/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
mrprev(124,1,11/1/87,unknown,h).


ud> listing(rereleased).

rereleased(124,1).
```

After an MRP II user invokes the **releasehold(mrprev)** operation, the status of the mrprev record is changed back to released, and the rereleased record is deleted, as shown in the listings following the operation.

```
ud> releasehold(mrprev).

Part Number? 124.
Revision Level? 1.
Revision has been released in MRP II


ud> listing(mrprev).

mrprev(123,2,9/1/87,unknown,r).
mrprev(123,1,7/1/87,9/1/87,r).
mrprev(234,1,7/15/87,10/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
mrprev(124,1,11/1/87,unknown,r).
```

```
ud> listing(rereleased).
```

In the next demonstration, a hold on a part revision is invoked via MRP II instead of CAD.

This is done with the **hold(mrprev)** operation. As the subsequent listings show, the status of the mrprev record changes to hold, but the status of the cadrev record remains released, since an MRP II hold is a local function. Rereleasing the part revision would be done with the **releasehold(mrprev)** operation, in the same manner as in the previous section.

```
ud> hold(mrprev).

Part Number? 124.
Revision Level? 1.
Revision has been given hold status in MRP II


ud> listing(mrprev).

mrprev(123,2,9/1/87,unknown,r).
mrprev(123,1,7/1/87,9/1/87,r).
mrprev(234,1,7/15/87,10/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
mrprev(124,1,11/1/87,unknown,h).


ud> listing(cadrev).

cadrev(123,1,7/1/87,9/1/87,o,file_1).
cadrev(234,1,7/15/87,10/1/87,r,inapp).
cadrev(234,2,10/1/87,unknown,r,inapp).
cadrev(123,2,9/1/87,unknown,o,file_2).
cadrev(124,1,11/1/87,unknown,r,file_3).
```

6.2.1.7 Part Deletions

The following three demonstrations involve the deletion of parts. In the first of these, a CAD user deletes the part added in section 6.2.1.1. Because both versions of this part have already been made obsolete, and since no product structure relationship, inventory, or on-order records have been created, the part is successfully deleted with the **delete(cadpart)** operation. The listings following the operation verify that the part, all revision, and latest revision records have been deleted in both CAD and MRP II (cadpart, cadrev, and latestcadrev in CAD, and mrppmr, mrprev, and latestmrprev in MRP II, respectively).

```
ud> delete(cadpart).

Part Number? 123.
Revision information has been deleted from MRP II
Revision has been deleted from CAD
Revision information has been deleted from MRP II
Revision has been deleted from CAD
Part has been deleted from MRP II
Part has been deleted from CAD


ud> listing(cadpart).

cadpart(234,inapp,inapp,mrp_part_1,each,unknown,unknown).
cadpart(124,unknown,unknown,cad_part_2,each,123,unknown).


ud> listing(cadrev).

cadrev(234,1,7/15/87,10/1/87,r,inapp).
cadrev(234,2,10/1/87,unknown,r,inapp).
cadrev(124,1,11/1/87,unknown,r,file_3).


ud> listing(latestcadrev).

latestcadrev(234,2).
```

```
latestcadrev(124,1).


ud> listing(mrppmr).

mrppmr(234,inapp,inapp,mrp_part_1,each,each,1,b,
       unknown,7,unknown,unknown).
mrppmr(124,unknown,unknown,cad_part_2,each,each,1,m,
       unknown,5,123,unknown).


ud> listing(mrprev).

mrprev(234,1,7/15/87,10/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
mrprev(124,1,11/1/87,unknown,h).


ud> listing(latestmrprev).

latestmrprev(234,2).
latestmrprev(124,1).
```

After successfully deleting the part entered via CAD in section 6.2.1.1, an attempt is made to delete the part entered via CAD in section 6.2.1.5. However in this case, the part has a released revision, so it cannot be deleted; the system prints out a message alerting the user to this fact, and the operation fails.

```
ud> delete(cadpart).

Part Number? 124.
Revision has Released Status--Cannot be Deleted
failure
```

The second delete scenario is the deletion of a CAD-generated part by an MRP II user.

For convenience, the database is returned to its state prior to the first deletion, and the part created in the previous session will again be deleted. Using the **delete(mrppmr)** operation, the subsequent listings confirm that while the system deletes the part, revision, and latest revision records from MRP II, they remain in CAD.

```
ud> delete(mrppmr).

Part Number? 123.
Revision information has been deleted from MRP II
Revision information has been deleted from MRP II
Part has been deleted from MRP II


ud> listing(mrppmr).

mrppmr(234,inapp,inapp,mrp_part_1,each,each,1,b,
       unknown,7,unknown,unknown).
mrppmr(124,unknown,unknown,cad_part_2,each,each,1,m,
       unknown,5,123,unknown).


ud> listing(mrprev).

mrprev(234,1,7/15/87,10/1/87,r).
mrprev(234,2,10/1/87,unknown,r).
mrprev(124,1,11/1/87,unknown,h).


ud> listing(latestmrprev).

latestmrprev(234,2).
latestmrprev(124,1).

ud> listing(cadpart).

cadpart(234,inapp,inapp,mrp_part_1,each,unknown,unknown).
cadpart(124,unknown,unknown,cad_part_2,each,123,unknown).
cadpart(123,unknown,unknown,cad_part_1,each,unknown,124).
```

```
ud> listing(cadrev).

cadrev(123,1,7/1/87,9/1/87,o,file_1).
cadrev(234,1,7/15/87,10/1/87,r,inapp).
cadrev(234,2,10/1/87,unknown,r,inapp).
cadrev(123,2,9/1/87,unknown,o,file_2).
cadrev(124,1,11/1/87,unknown,r,file_3).

ud> listing(latestcadrev).

latestcadrev(234,2).
latestcadrev(124,1).
```

Though nothing is deleted from CAD, the CAD records must meet the same requirements as if they were actually being deleted. This is shown by the following attempt by an MRP II user to delete the other CAD-generated part. Just like the CAD user in the previous scenario, the MRP II user is not allowed to delete any information about this part because it has a released revision in CAD.

```
ud> delete(mrppmr).

Part Number? 124.
Revision has released status in CAD--cannot delete
failure
```

The third delete scenario is the deletion of the part generated by an MRP II user in section 6.2.1.2.

This session begins with a CAD user attempting to delete the part using the **delete(cadpart)** operation; because the part is controlled by MRP II, however, he or she is not permitted to do so.

```
ud> delete(cadpart).
```

```
Part Number? 234.
Revision can only be deleted via MRP II
Revision can only be deleted via MRP II
failure
```

Next, an MRP II user uses **delete(mrppmr)**, and the part is sucessfully deleted. As shown in the listings following the operation, the part, revision, and latest revision records are removed from both MRP II and CAD. Note that there are no restrictions on the status of the revsions in CAD in this case.

```
ud> delete(mrppmr).

Part Number? 234.
Revision information has been deleted form MRP II
Revision information has been deleted form MRP II
Part has been deleted from CAD
Part has been deleted from MRP II


ud> listing(mrppmr).

mrppmr(124,unknown,unknown,cad_part_2,each,each,1,m,
       unknown,5,123,unknown).


ud> listing(mrprev).

mrprev(124,1,11/1/87,unknown,h).


ud> listing(latestmrprev).

latestmrprev(124,1).


ud> listing(cadpart).

cadpart(124,unknown,unknown,cad_part_2,each,123,unknown).
cadpart(123,unknown,unknown,cad_part_1,each,unknown,124).
```

```
ud> listing(cadrev).

cadrev(123,1,7/1/87,9/1/87,o,file_1).
cadrev(123,2,9/1/87,unknown,o,file_2).
cadrev(124,1,11/1/87,unknown,r,file_3).


ud> listing(latestcadrev).

latestcadrev(124,1).
```

6.2.2 Product Structures

In this section, the basic product structure operations are demonstrated. For this portion of the demonstration, a different database, containing several parts, is used. These parts are shown in the following listing; note that one of these parts is local to CAD with its version having a "working" status.

```
ud> listing(cadpart).

cadpart(123,unknown,unknown,assembly1,each,unknown,unknown).
cadpart(124,unknown,unknown,assembly2,each,unknown,unknown).
cadpart(125,unknown,unknown,assembly3,each,unknown,unknown).
cadpart(234,unknown,unknown,component1,each,unknown,unknown).
cadpart(235,unknown,unknown,component2,ft,unknown,unknown).
cadpart(236,unknown,unknown,component3,each,unknown,unknown).
cadpart(237,unknown,unknown,component4,lb,unknown,unknown).
cadpart(238,unknown,unknown,component5,each,unknown,unknown).

ud> listing(cadrev).

cadrev(123,1,unknown,unknown,w,file_a1).
cadrev(124,1,7/1/87,unknown,r,file_a2).
cadrev(125,1,8/1/87,unknown,r,file_a3).
cadrev(234,1,6/1/87,unknown,r,file_c1).
cadrev(235,1,6/15/87,unknown,r,file_c2).
cadrev(236,1,7/15/87,unknown,r,file_c3).
cadrev(237,1,7/1/87,unknown,r,file_c4).
cadrev(238,1,6/30/87,unknown,r,file_c5).
```

```
ud> listing(mrppmr).

mrppmr(124,unknown,unknown,assembly2,each,each,1,m,
       unknown,3,unknown,unknown).
mrppmr(125,unknown,unknown,assembly3,each,each,1,m,
       unknown,1,unknown,unknown).
mrppmr(234,unknown,unknown,component1,each,box,12,b,
       unknown,9,unknown,unknown).
mrppmr(235,unknown,unknown,component2,ft,ft,1,b,
       unknown,6,unknown,unknown).
mrppmr(236,unknown,unknown,component3,each,each,1,m,
       unknown,2,unknown,unknown).
mrppmr(237,unknown,unknown,component4,lb,lb,1,b,
       unknown,14,unknown,unknown).
mrppmr(238,unknown,unknown,component5,each,each,1,m,
       unknown,4,unknown,unknown).

ud> listing(mrprev).

mrprev(124,1,7/1/87,unknown,r).
mrprev(125,1,8/1/87,unknown,r).
mrprev(234,1,6/1/87,unknown,r).
mrprev(235,1,6/15/87,unknown,r).
mrprev(236,1,7/15/87,unknown,r).
mrprev(237,1,7/1/87,unknown,r).
mrprev(238,1,6/30/87,unknown,r).
```

6.2.2.1 Adding Component Relationships Via CAD

In the first product structure scenario, a simple bill of material for the working CAD revision will be constructed via CAD with the operation **insert(cadcomponent)**. The user session for this procedure follows.

In the first call to the operation, the user enters the values of the parent part and revision. Because the status of the parent revision is "working," indicating the construction of a new product structure, the system does not ask the user if a new assembly part number or revision is necessary. The system then asks the

user for each of the three remaining fields, all of which are required. The listings following the operation show that the relationship has been added to CAD as a cadcomponent record, but not to MRP II, since the parent part revision still has a "working" status in CAD.

```
ud> insert(cadcomponent(123,1,_,_,_)).

Item Number? 1.
Component part number? 234.
Quantity per assembly? 2.
Component relationship has been added to CAD


ud> listing(cadcomponent).

cadcomponent(123,1,1,234,2).


ud> listing(mrpcomponent).
```

Before entering another relationship, the parent part's first version is released, so that the addition of a relationship to a released CAD revision can be demonstrated. This version is released as shown in the previous section, using operation **releasework(cadrev)**. In addition to the actions triggered in those examples, the component relationship is inserted into an mrpcomponent record.

```
ud> releasework(cadrev).

Part Number? 123.
Revision Level? 1.
Part has been added to MRP II
Revision has been added to MRP II
Component relationship has been added to MRP II
Revision has been released in CAD
```

```
ud> listing(cadcomponent).

cadcomponent(123,1,1,234,2).



ud> listing(mrpcomponent).

mrpcomponent(123,1,1,234,2).


ud> listing(mrppmr).

mrppmr(124,unknown,unknown,assembly2,each,each,1,m,
       unknown,3,unknown,unknown).
mrppmr(125,unknown,unknown,assembly3,each,each,1,m,
       unknown,1,unknown,unknown).
mrppmr(234,unknown,unknown,component1,each,box,12,b,
       unknown,9,unknown,unknown).
mrppmr(235,unknown,unknown,component2,ft,ft,1,b,
       unknown,6,unknown,unknown).
mrppmr(236,unknown,unknown,component3,each,each,1,m,
       unknown,2,unknown,unknown).
mrppmr(237,unknown,unknown,component4,lb,lb,1,b,
       unknown,14,unknown,unknown).
mrppmr(238,unknown,unknown,component5,each,each,1,m,
       unknown,4,unknown,unknown).
mrppmr(123,unknown,unknown,assembly1,each,
       unknown,unknown,unknown,unknown,unknown,unknown,unknown).


ud> listing(mrprev).

mrprev(124,1,7/1/87,unknown,r).
mrprev(125,1,8/1/87,unknown,r).
mrprev(234,1,6/1/87,unknown,r).
mrprev(235,1,6/15/87,unknown,r).
mrprev(236,1,7/15/87,unknown,r).
mrprev(237,1,7/1/87,unknown,r).
mrprev(238,1,6/30/87,unknown,r).
mrprev(123,1,unknown,unknown,h).
```

With the parent part's first revision released, a second component relationship

is added with **insert(cadcomponent)**. Again, no part number or revision level change at the assembly level is required, since the operation is part of the construction of the same bill of material. First, the user tries to insert the same item number as in the previous relationship, but a different component part number. This violates the system constraints, however, and the operation fails.

```
ud> insert(cadcomponent).

Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Parent part number? 123.
Parent revision level? 1.
Item Number? 1.
Component part number? 235.
Quantity per assembly? 3.
Item number already exists in CAD
failure
```

The user next corrects the mistake and enters the relationship with a new item number. As shown in the subsequent listings, the operation is successful, and both cadcomponent and mrpcomponent records are created to represent the relationship.

```
ud> insert(cadcomponent).

Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Parent part number? 123.
Parent revision level? 1.
Item Number? 2.
Component part number? 235.
Quantity per assembly? 3.
Component relationship has been added to MRP II
Component relationship has been added to CAD


ud> listing(cadcomponent).
```

```
cadcomponent(123,1,1,234,2).
cadcomponent(123,1,2,235,3).


ud> listing(mrpcomponent).

mrpcomponent(123,1,1,234,2).
mrpcomponent(123,1,2,235,3).
```

6.2.2.2 Adding Component Relationships Via MRP II

In the next scenario, the insersion of a relationship component via MRP II is demonstrated. For convenience, the database is returned to its state just after the release of the first version of the parent part from CAD, and the second relationship is added again. The MRP II user invokes operation **insert(mrpcomponent)**, and this time the parent part number and revision level are entered as part of the call. No assembly level part number or revision level change is required, and after entering the remaining values, the relationship is created in both databases, as represented by the cadcomponent and mrpcomponent records. These are shown in the listings following the operation.

```
ud> insert(mrpcomponent(123,1,_,_,_)).

Current assembly part number: 123
Current assembly revision level: 1
Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Item Number? 2.
Component part number? 235.
Quantity per assembly? 3.
Component relationship has been added to CAD
Component relationship has been added to MRP II
```

```
ud> listing(mrpcomponent).

mrpcomponent(123,1,1,234,2).
mrpcomponent(123,1,2,235,3).


ud> listing(cadcomponent).

cadcomponent(123,1,1,234,2).
cadcomponent(123,1,2,235,3).
```

To demonstrate another system constraint, an MRP II user attempts to insert a relationship consisting of the component from the previous relationship as the parent and the parent from the previous relationship as the component. Since adding such a relationship would result in a looping of the product structure, where an assembly has itself as a lower level component, the system does not allow the relationship to be added.

```
ud> insert(mrpcomponent).

Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Parent part number? 235.
Parent revision level? 1.
Item Number? 1.
Component part number? 123.
Quantity per assembly? 4.
Relationship results in loop, cannot be added
failure
```

6.2.2.3 Adding Component Relationships Requiring a New Assembly Part Number

As a final demonstration of the addition of component relationships, a CAD user adds a third relationship. Following engineering change control procedures, it has

been determined that this addition requires a new assembly part number, so the user uses the option in the **insert(cadcomponent)** operation to create the new part. Following the insersion of the new part, the bill of material from the existing assembly is copied to the new assembly. The system then requests the details of the new relationship for the new part, and it is added. As shown in the listings following the operation, since the first version of the part just added is automatically given a working status in CAD, the copied relationships as well as the one just added are local to CAD. Also note in the listing following the operation that the new part record is automatically inserted with the orignial assembly number in its supersedes part number field. When the part is released using **releasework(cadrev)**, all three relationships will be transferred to MRP II.

```
ud> insert(cadcomponent(123,1,_,_,_)).

Current assembly part number: 123
Current assembly revision level: 1
Does this change require a new assembly part number? yes.
Part Number? 126.
Description? new_assembly1.
Unit of Measure? each.
New Revision Level? 1.
Drawing File Name? file_a3.
Revision has been added to CAD
Part has been added to CAD
Component relationship has been added to CAD
Component relationship has been added to CAD
Product structure has been copied
Item Number? 3.
Component part number? 236.
Quantity per assembly? 1.
Component relationship has been added to CAD


ud> listing(cadpart).

cadpart(123,unknown,unknown,assembly1,each,unknown,unknown).
```

```
cadpart(124,unknown,unknown,assembly2,each,unknown,unknown).
cadpart(125,unknown,unknown,assembly3,each,unknown,unknown).
cadpart(234,unknown,unknown,component1,each,unknown,unknown).
cadpart(235,unknown,unknown,component2,ft,unknown,unknown).
cadpart(236,unknown,unknown,component3,each,unknown,unknown).
cadpart(237,unknown,unknown,component4,lb,unknown,unknown).
cadpart(238,unknown,unknown,component5,each,unknown,unknown).
cadpart(126,unknown,unknown,new_assembly1,each,123,unknown).


ud> listing(cadrev).

cadrev(124,1,7/1/87,unknown,r,file_a2).
cadrev(125,1,8/1/87,unknown,r,file_a3).
cadrev(234,1,6/1/87,unknown,r,file_c1).
cadrev(235,1,6/15/87,unknown,r,file_c2).
cadrev(236,1,7/15/87,unknown,r,file_c3).
cadrev(237,1,7/1/87,unknown,r,file_c4).
cadrev(238,1,6/30/87,unknown,r,file_c5).
cadrev(123,1,unknown,unknown,r,file_a1).
cadrev(126,1,unknown,unknown,w,file_a3).


ud> listing(cadcomponent).

cadcomponent(123,1,1,234,2).
cadcomponent(123,1,2,235,3).
cadcomponent(126,1,3,236,1).
cadcomponent(126,1,2,235,3).
cadcomponent(126,1,1,234,2).


ud> listing(mrppmr).

mrppmr(124,unknown,unknown,assembly2,each,each,1,m,
       unknown,3,unknown,unknown).
mrppmr(125,unknown,unknown,assembly3,each,each,1,m,
       unknown,1,unknown,unknown).
mrppmr(234,unknown,unknown,component1,each,box,12,b,
       unknown,9,unknown,unknown).
mrppmr(235,unknown,unknown,component2,ft,ft,1,b,
       unknown,6,unknown,unknown).
mrppmr(236,unknown,unknown,component3,each,each,1,m,
       unknown,2,unknown,unknown).
mrppmr(237,unknown,unknown,component4,lb,lb,1,b,
```

```
        unknown,14,unknown,unknown).
mrppmr(238,unknown,unknown,component5,each,each,1,m,
        unknown,4,unknown,unknown).
mrppmr(123,unknown,unknown,assembly1,each,unknown,
        unknown,unknown,unknown,unknown,unknown,unknown).


ud> listing(mrprev).

mrprev(124,1,7/1/87,unknown,r).
mrprev(125,1,8/1/87,unknown,r).
mrprev(234,1,6/1/87,unknown,r).
mrprev(235,1,6/15/87,unknown,r).
mrprev(236,1,7/15/87,unknown,r).
mrprev(237,1,7/1/87,unknown,r).
mrprev(238,1,6/30/87,unknown,r).
mrprev(123,1,unknown,unknown,h).


ud> listing(mrpcomponent).

mrpcomponent(123,1,1,234,2).
mrpcomponent(123,1,2,235,3).
```

6.2.2.4 Deleting Component Relationships Via CAD

Next the deletion of component relationships is demonstrated. For these operations, the database is reset to its state prior to the previous operation, leaving only the single, two component bill of material, as shown in the following listing:

```
ud> listing(cadcomponent).

cadcomponent(123,1,1,234,2).
cadcomponent(123,1,2,235,3).


ud> listing(mrpcomponent).

mrpcomponent(123,1,1,234,2).
mrpcomponent(123,1,2,235,3).
```

In this demonstration, one of these two components is deleted by a CAD user. Due to the previously mentioned limitation of the current combination of the interpreter and model, it is not possible for the user to make a revision or part number change to the assembly as part of the delete operation. Hence the user answers "no" to these questions. If the deletion requires either a new part number or revision level for the assembly, the user should insert the appropriate record and make the change to the new bill of material. For this demonstration, the change will be made to the original structure. The user calls the operation **delete(cadcomponent)**, enters the four required values, and the deletion is processed. Because the component relationship exists both in CAD and in MRP II, it is removed from both databases, as shown in the subsequent listings. If the parent part revision had a working status, the relationship would have been local to CAD, and would have been deleted from CAD without any interaction with MRP II.

```
ud> delete(cadcomponent).

Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Parent part number? 123.
Parent revision level? 1.
Item number? 1.
Component part number? 234.
Relationship has been deleted from MRP II
Relationship has been deleted from CAD


ud> listing(cadcomponent).

cadcomponent(123,1,2,235,3).


ud> listing(mrpcomponent).
```

```
mrpcomponent(123,1,2,235,3).
```

6.2.2.5 Deleting Component Relationships Via MRP II

In the next session, the second component relationship is deleted by an MRP II user. Again no part number or revision level change can currently be made as part of the delete operation. The delete process in MRP II is invoked by the **delete(mrpcomponent)** operation, which is essentially the same as the **delete(cadcomponent)** operation in CAD: the system asks for the required values, and the relationship is deleted from both MRP II and CAD, as shown in the listings following the operation.

```
ud> delete(mrpcomponent).
Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Parent part number? 123.
Parent revision level? 1.
Item number? 2.
Component part number? 235.
Relationship has been deleted from CAD
Relationship has been deleted from MRP II


ud> listing(mrpcomponent).


ud> listing(cadcomponent).
```

6.2.2.6 Mass Substitution of Components Via CAD

In the next scenario, a user performs a mass substitution of a component. For this part of the demonstration, the database is returned to its state at the beginning of the product structures demonstration, with the addition of several relationships,

involving several assemblies, to make the operation more realistic. The component relationships are shown in the following listings:

```
ud> listing(cadcomponent).

cadcomponent(123,1,1,234,2).
cadcomponent(123,1,2,235,3).
cadcomponent(124,1,1,234,3).
cadcomponent(124,1,2,236,1).
cadcomponent(125,1,1,234,2).
cadcomponent(125,1,2,237,5).


ud> listing(mrpcomponent).

mrpcomponent(124,1,1,234,3).
mrpcomponent(124,1,2,236,1).
mrpcomponent(125,1,1,234,2).
mrpcomponent(125,1,2,237,5).
```

In this demonstration, the mass substitution is initiated in CAD using operation **substitutepartcad**. Note that because of the nature of this operation, in which the values required are not part of any existing relation, no relation name is included in the call.

After the user enters the old and new part numbers, the system begins searching for occurences of the old part as a component in relationships in the CAD database. Each time it finds one, it asks the user if that particular substitution necessitates a new assembly part number or revision level. Because the current interpreter/model combination cannot properly handle these operations as part of the substitution (which would include a modification to a relation added in the same operation), the user answers "no" to each question. The listings after the operation show that each instance of the old part as a component has been replaced by the new part in

the CAD database. Further, for all of these relationships having a parent that has already been released to MRP II, the substitution is made in MRP II as well.

```
ud> substitutepartcad.

Current part number? 234.
Part number to substitute? 238.

Current assembly part number: 123
Current assembly revision level: 1
Relationship has been deleted from MRP II
Relationship has been deleted from CAD
Component relationship has been added to MRP II
Component relationship has been added to CAD

Current assembly part number: 124
Current assembly revision level: 1
Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Relationship has been deleted from MRP II
Relationship has been deleted from CAD
Component relationship has been added to MRP II
Component relationship has been added to CAD

Current assembly part number: 125
Current assembly revision level: 1
Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Relationship has been deleted from MRP II
Relationship has been deleted from CAD
Component relationship has been added to MRP II
Component relationship has been added to CAD

Part substitution has been completed


ud> listing(cadcomponent).

cadcomponent(123,1,1,238,2).
cadcomponent(123,1,2,235,3).
cadcomponent(124,1,1,238,3).
cadcomponent(124,1,2,236,1).
cadcomponent(125,1,1,238,2).
```

```
cadcomponent(125,1,2,237,5).


ud> listing(mrpcomponent).

mrpcomponent(124,1,2,236,1).
mrpcomponent(124,1,1,238,3).
mrpcomponent(125,1,1,238,2).
mrpcomponent(125,1,2,237,5).
```

6.2.2.7 Mass Substitution of Components Via MRP II

In the next session, the equivalent operation is performed by an MRP II user using operation **substitutepartmrp**. Prior to the call, the database is reset to its state just before the part substitution from CAD so that the same substitutaion may be made:

```
ud> listing(cadcomponent).

cadcomponent(123,1,1,234,2).
cadcomponent(123,1,2,235,3).
cadcomponent(124,1,1,234,3).
cadcomponent(124,1,2,236,1).
cadcomponent(125,1,1,234,2).
cadcomponent(125,1,2,237,5).


ud> listing(mrpcomponent).

mrpcomponent(124,1,1,234,3).
mrpcomponent(124,1,2,236,1).
mrpcomponent(125,1,1,234,2).
mrpcomponent(125,1,2,237,5).
```

Once again the user answers "no" to the creation of a new part number or revision for each assembly in which the old component is found. The substitution is

completed, and the results are shown in the subsequent listings. All of the mrpcomponent records with the old part as a component have been modified to show the new part as a component. The cadcomponent records of the same relationships have also been modified to reflect the substitution. Note that the relationship involving the old part as a component of a CAD parent part revision with a working status has not been updated in CAD, since the operation was invoked by an MRP II user and this relationship does not exist in MRP II. As long as the parent revision is local to CAD, substitutions may be made only by CAD users.

```
ud> substitutepartmrp.

Current part number? 234.
Part number to substitute? 238.

Current assembly part number: 124
Current assembly revision level: 1
Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Relationship has been deleted from CAD
Relationship has been deleted from MRP II
Component relationship has been added to CAD
Component relationship has been added to MRP II

Current assembly part number: 125
Current assembly revision level: 1
Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Relationship has been deleted from CAD
Relationship has been deleted from MRP II
Component relationship has been added to CAD
Component relationship has been added to MRP II

Part substitution has been completed


ud> listing(mrpcomponent).

mrpcomponent(124,1,1,238,3).
```

```
mrpcomponent(124,1,2,236,1).
mrpcomponent(125,1,1,238,2).
mrpcomponent(125,1,2,237,5).


ud> listing(cadcomponent).

cadcomponent(123,1,1,234,2).
cadcomponent(123,1,2,235,3).
cadcomponent(124,1,1,238,3).
cadcomponent(124,1,2,236,1).
cadcomponent(125,1,1,238,2).
cadcomponent(125,1,2,237,5).
```

6.2.2.8 Modifying the Quantity of Components per Assembly

The modification of the quantity per assembly is the subject of the next scenario. Again this process may be initiated by either CAD or MRP II users. Because the two operations, **modifyquantity(cadcomponent)** and **modifyquantity(mrpcomponent)** work in essentially the same manner, only one of them, the CAD operation, will be demonstrated.

When the user enters the operation, he or she is again asked if the change necessitates a new assembly part number or revision level. To avoid the current problem with the system, the user responds "no." In the likely event that either a new part number or revision level were required, the user would perform these operations prior to making the change, and the modification would be made to the new product structure.

The listings following the operation verify that the quantity per assembly has been changed in both the cadcomponent and mrpcomponent records. If the operation were performed on a CAD part revision with a working status, the change would have only been made in CAD, since the part would be local to CAD in that

case. If the operation were made via MRP II, the change would have also been made in both MRP II and CAD.

```
ud> modifyquantity(cadcomponent).

Does this change require a new assembly part number? no.
Does this change require a new assembly revision level? no.
Parent part number? 124.
Parent revision level? 1.
Item number? 1.
Component part number? 238.
New quantity per assembly? 4.
Quantity per assembly has been changed in MRP II
Quantity per assembly has been changed in CAD


ud> listing(cadcomponent).

cadcomponent(123,1,1,234,2).
cadcomponent(123,1,2,235,3).
cadcomponent(124,1,1,238,4).
cadcomponent(124,1,2,236,1).
cadcomponent(125,1,1,238,2).
cadcomponent(125,1,2,237,5).


ud> listing(mrpcomponent).

mrpcomponent(124,1,1,238,4).
mrpcomponent(124,1,2,236,1).
mrpcomponent(125,1,1,238,2).
mrpcomponent(125,1,2,237,5).
```

6.2.2.9 Copying Product Structures Via CAD

The final product structure scenario demonstrates the copying of a product structure of one assembly to another. The product structure consists of the entire collection of component relationships containing the "copy from" assembly as the parent. This operation is intended for use when a change to an assembly calls for a new part

number. In such cases, the user could insert the new part using **insert(cadpart)**; the bill of material from the previous assembly can then be copied to the new assembly, and the change made to the new bill of material.

In the following operation, a bill of material is copied from a released part revision in CAD to a working part revision which will have a similar product structure. The existing relationships are shown in the following listing:

```
ud> listing(cadcomponent).

cadcomponent(124,1,2,236,1).
cadcomponent(124,1,1,238,4).


ud> listing(mrpcomponent).

mrpcomponent(124,1,2,236,1).
mrpcomponent(124,1,1,238,4).
```

To copy the bill of material, the operation **copybomcad** is used. As with the **substitutepartcad** operation, the values required do not correspond to any existing relation, so no relation is used in the call.

The user enters the part and revision of the assembly to copy the bill of material from, and the part and revision of the assembly to copy the bill of material to. In the subsequent listings, it can be seen that all of the relationships with the "copy from" assembly as the parent have, in fact, been copied to new relationships with the "copy to" assembly as parent. The CAD user can next use any of the previously described operations to complete the construction of the new product structure. Because the part revision of the copy to assembly is local to CAD, the copied relationships exist only in the CAD database. As previously demonstrated, the relationships will be

copied to the MRP II database at the time of the release of the new assembly from CAD.

```
ud> copybomcad.

From part number? 124.
From revision level? 1.
To part number? 123.
To revision level? 1.
Component relationship has been added to CAD
Component relationship has been added to CAD
Product structure has been copied


ud> listing(cadcomponent).

cadcomponent(124,1,2,236,1).
cadcomponent(124,1,1,238,4).
cadcomponent(123,1,1,238,4).
cadcomponent(123,1,2,236,1).


ud> listing(mrpcomponent).

mrpcomponent(124,1,2,236,1).
mrpcomponent(124,1,1,238,4).
```

If the copy to assembly were to have had a "released" status in CAD, then the new relationship records would have been immediately transferred to MRP II.

6.2.2.10 Copying Product Structures Via MRP II

The equivalent operation, **copybommrp**, is next performed by an MRP II user, also as part of the creation of the bill of material for a new assembly. The user is asked for the same information requested of the CAD user, and the transaction is processed. As the listings following the operation show, the relationships are copied to the new assembly as both mrpcomponent and cadcomponent records.

Subsequent to this operation, the MRP II user would modify the copied assembly to complete the creation of the assembly for the new part.

```
ud> copybommrp.

From part number? 124.
From revision level? 1.
To part number? 125.
To revision level? 1.
Component relationship has been added to CAD
Component relationship has been added to MRP II
Component relationship has been added to CAD
Component relationship has been added to MRP II
Product structure has been copied


ud> listing(mrpcomponent).

mrpcomponent(124,1,2,236,1).
mrpcomponent(124,1,1,238,4).
mrpcomponent(125,1,1,238,4).
mrpcomponent(125,1,2,236,1).


ud> listing(cadcomponent).

cadcomponent(124,1,2,236,1).
cadcomponent(124,1,1,238,4).
cadcomponent(123,1,1,238,4).
cadcomponent(123,1,2,236,1).
cadcomponent(125,1,1,238,4).
cadcomponent(125,1,2,236,1).
```

The sample operations performed in this section represent the basic capabilities of the interoperability system. The model is a simplification of the activities and data exchange involved in a typical organization, a result of the desire to remain as general as possible in the solution to MRP II/CAD integration. In its present version, the usefullness of the model is limited by its isolation from actual CAD

and MRP II systems, forcing the users to interact directly with the interoperability system. As the research progresses, the sophistication of the model will increase and the interoperability system will become more transparent to the user, interacting with the application programs directly.

Chapter 7

Conclusions

The need to reexamine the traditional, fragmented approach to manufacturing and production automation has become quite clear. Though technology has provided significant advances in many of the individual production activities, there has been a distinct lack of long-term planning concerning the issue of how all of the various pieces might eventually work together to provide support for the entire range of production and manufacturing activities, from design conception to distribution and customer service.

There are many reasons why current efforts are often narrowly focused. Perhaps the most significant is the fact that many of the systems available today, including Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Robotics, and so on, fall short of what has been envisioned to be their potential capabilities. Thus considerable resources are being expended to further develop these individual technologies, often with little consideration of related activities.

Computer Integrated Manufacturing (CIM) offers the promise of uniting the various computer aided systems and tools into a single comprehensive manufacturing system. Just how this will be achieved is a subject of considerable debate. It has been proposed in this work that perhaps the most practical approach to CIM is to attempt to integrate the existing manufacturing hardware and software systems in

a modular fashion. This provides the flexibility of selecting the best suited system for each individual task, and allows the conservation of the investment in resources, training, and user-familiarity that a company may already have in particular systems.

Manufacturing Resource Planning (MRP II), with its focus on the planning, scheduling, and monitoring of manufacturing activities, is best qualified to become the central "hub" of CIM, coordinating the flow of information among the satellite systems. The satellite systems include: Computer Aided Design, whose role it is to support design activities and supply product information to the rest of the organization; Computer Aided Process Planning, which serves to develop the detailed process plans, or instructions, for manufacturing products; and Computer Aided Manufacturing, which supports the physical transformation of raw materials into finished parts.

The proposed starting point for this system is the integration of MRP II and CAD, involving static part and product structure information. The roles of each application in the integrated system have been defined, with CAD controlling activities relating to the creation and modification of design information, and MRP II controlling activities relating to the procurement or manufacture of all items. In general, CAD users have the primary responsibility of the disposition of all designed parts, while MRP II users are given the responsibility only for parts that are not designed, such as catalogued purchased parts.

Given this framework, the functional description of an integrated MRP II/CAD system has been presented. When developing this description, an intentional effort was made to avoid following the format of any particular CAD or MRP II systems. Instead, the generic capabilities in keeping with the roles just described were used as a basis for the construction of the model. However, actual systems, such as

CONTROL Manufacturing^{TM} from Cincom Systems, and *ICEM*^{TM} from Control Data Corporation have been used to check most of the logic incorporated into the model.

The model considers the basic activities involved in the addition and maintenance of part data and product structures, including engineering changes requiring design revisions. The information maintained by the system includes some of the typical part attributes maintained in CAD systems, and the basic part master data maintained in MRP II systems. It is assumed that there are additional data maintained in each system beyond those considered in the integration. The goal of the system is to provide the ability to enter and modify data from either system as appropriate, while ensuring consistency between them at all times. The transfer of information between the two applications is controlled by *status codes* assigned to the data pertaining to each version of each part in each system. With these status codes, it is possible to control the natural "time lag" between design and manufacturing or purchasing activities that results from MRP II planning, while maintaining data consistency.

To demonstrate the MRP II/CAD integrated model, the functional specifications have been programmed in the Update Dependency Language developed by members of the Computer Science Department at the University of Maryland. This language is designed specifically to provide interoperability between two or more application systems using a declarative, rather than procedural, representation. The resulting code is in the form of operations, each comprised of a series of "rules," that include a set of conditions followed by a list of implied operations. If the conditions of a particular rule are satisfied, the system attempts to perform the implied operations given in that rule. Each rule thus specifies the specific actions to be taken under specific circumstances.

At this early stage in the implementation of the language, actual interoperability is not possible. The model MRP II/CAD system functions as an independent system, not interfaced with either an actual MRP II or CAD system. Even so, the system as programmed demonstrates the basic functionality of the integrated system, though the lack of a commercial interface makes the system somewhat difficult to use. Another consequence of the isolation of the system is that in order to demonstrate particular portions of the integrated system, one must program functions that already exist in the applications, which requires extra coding. More importantly, making the extra functions part of the interoperability system has resulted in some problems with the current version of the Update Dependency Language interpreter. As the Update Dependency Language moves closer to its goal of making true interoperability possible, the extra programming currently required will be considerably reduced, as will the problems associated with it, as many of the longer operations will naturally be broken down into a series of smaller operations.

Database interoperability offers a convenient tool for specifying the behavior of an integrated system, allowing a natural representation of complex interactions simply by enumerating the possible courses of action. This approach can also be used to modify the behavior of a single system. Because the system is comprised of lists of individual cases, it is easy to expand or modify simply by adding or changing cases. One drawback of this approach, however, is that as the size of the interoperability system increases, the performance of the system tends to suffer. The need to enumerate cases such as tests for key attribute instantiation contributes to this problem. A further detriment to system performance is the fact that the current version of the language is interpreted instead of compiled. Nonetheless, the need to express such complex dependency operations merits the use of the language.

Chapter 8

Recommendations for Further Work

The integration of Manufacturing Resource Planning and Computer Aided Design presented in this work is only a first step toward Computer Integrated Manufacturing. There are many areas deserving attention in the future. The first is the enhancement of the current model to include features to assist in maintaining consistency between the two databases, but not often found in commercial systems, such as:

- Automatic generation of part numbers based on any particular numbering system
- An automatic look-up table to match bill of material and purchasing/inventory units of measure and determine the proper conversion factor without the user having to enter this value.

The part number generator could be implemented as an interoperability function called each time a part is created via either application, with the resulting number being eventually propagated to the other application. The conversion factor look-up table would be implemented purely to serve MRP II, since that particular field is not maintained in CAD. An operation such as this demonstrates how the Update

Dependency Language can be used to support a particular application as well as supporting the interoperability of multiple applications.

One area of the current system that may not deserve much effort is the reconstruction of the product structure routines to avoid the current problem which prevents the user from taking advantage of the ability to insert a new assembly part number or revision level while making a modification to a bill of material other than the addition of a new component. This problem occurs because the current version of the Update Dependency Language interpreter does not allow the deletion or modification of a relation added in the same operation, which is the approach currently taken when using this feature; the new part or revision is created, the bill of material from the old part or revision is copied to the new one, and changes are made to the new bill of material. This approach was used because it allowed maximum use of already-existing routines; correcting the problem at this stage may take considerable reprogramming of a large number of routines. Alternatevely, if the interpreter were changed, such that this restriction was removed, no changes would be necessary. On the other hand, as the interpreter evolves to handle true interoperability between actual MRP II and CAD systems, it is anticipated that the problem will disappear, as the chain of operations causing the problem should be naturally broken down into smaller series of operations called successively by the application.

This last statement elicits the mention of an obvious direction for this research to take: the extension of the Update Dependency Language to a true interoperability system. As described in section 5.2.3, the next steps in the development of the Update Dependency Language are to evolve from the single instance of the interpreter used in the current model to two instances of the interpreter running on the same machine, and then to two instances of the interpreter running on different

machines. During these two stages, however, the users of the model system will be insulated from the changes; the model should behave as it does now, though the system may have to be modified somewhat to handle the multiple instances of the interpreter. When the interpreter reaches the stage in its development that it can handle the interoperability of two actual application systems, the interoperability system will need to be reprogrammed to match the cababilities of the particular applications, though the model presented here should serve as a basis for the new system.

As the Update Dependency Language evolves, it is likely that the approach of using an interpreted language will prove to be a serious detriment to system performance. It is therefore recommended that as part of the evolution, the language be rewritten as a compiler, which should significantly improve its speed.

Another extension of this work is the application of the interoperability concept to the integration of other systems involved in CIM. For example, the addition of Computer Aided Process Planning to the model MRP II/CAD system could be programmed using three sets of operations to tie the systems together: one set in CAD, one set in MRP II, and one set in CAPP itself. The basis of interoperability between CAD and CAPP includes a part's geometry, material specification, desired tolerances and surface finish. The basis of interoperability between MRP II and CAPP includes the routings, or process plans, for each manufactured part, though MRP II does not require the same level of detail as provided by CAPP.

Similarly, Computer Aided Manufacturing could be added to the model, though this appears to be a more distant possibility. The two primarly areas of interoperability between CAM and the previous system include MRP II and CAPP, though the type of information being exchanged between these systems differs from the previous cases in that it should be dynamic as opposed to static. MRP II should feed

a shop floor schedule to CAM based on its requirements for end products, and its rough-cut routings obtained from CAPP. CAM should provide feedback to MRP II as the activities progress. CAPP should provide the detailed process plans and part programs that actually "instruct" the machines how to make the parts. Because of the dynamic nature of the interoperability, it would be much harder to adequately demonstrate this integration with the current approach; perhaps adding CAM to the integrated system would require the use of actual application systems.

Bibliography

[1] Anderson, D. C., J. J. Solberg, and R. P. Paul. "Factories of the Future: How will Automation Research be Integrated?," *Computers in Mechanical Engineering*, vol. 2, no. 4 (January 1984), 31-36.

[2] Appleton, Daniel S. "The State of CIM," *Datamation*, vol. 30, no. 21 (December 15, 1984), 66-72.

[3] Baer, Tony. "With Group Technology, No One Reinvents the Wheel," *Mechanical Engineering*, vol. 107, no. 11 (November 1985), 60-69.

[4] Bauer, Curtis P. "Automating Bill of Material Generation," *Auerbach Industry Application Series: Computer-Aided Design, Engineering, and Drafting*, vol. 1 (1984), 1-16.

[5] Bohse, Michael E. and George Harhalakis. "Integrating CAD and MRP," *Auerbach Industry Application Series: Manufacturing Resource Planning*, vol. 1 (1986), 1-12.

[6] Burgam, Patric. "Marrying MRP and CIM," *CAD/CAM Technology*, vol. 2, no. 4 (Winter, 1983), 25-27.

[7] Francis, Philip H. "Toward a Science of Manufacturing," *Mechanical Engineering*, vol. 108, no. 5 (May, 1986), 32-37.

[8] Fox, Kenneth A. "MRP II Providing a Natural 'Hub' for Computer Integrated Manufacturing Systems," *Industrial Engineering*, vol. 16, no. 10 (October, 1984), 44-50.

[9] Gershwin, Stanley B., et. al. "A Control Perspective on Recent Trends in Manufacturing Systems," *IEEE Control Systems Magazine*, vol. 6, no. 2 (April, 1986), 3-15.

[10] Gunn, Thomas G. "CAD/CAM/CIM: Now and in the Future," *I&CS*, vol. 58, no. 4 (April, 1985), 59-64.

[11] Harhalakis, George. "Engineering Changes for Made-to-Order Products: How an MRP II System Should Handle Them," *Engineering Management International*, vol. 4, no. 1 (October, 1986), 19-36.

[12] Harhalakis, George. *An Integrated Production Managment System for Engineered Equipment*, PhD thesis, University of Manchester Institute of Science and Technology (1984).

[13] Harhalakis, George, Michael E. Bohse, and B. J. Davies. "Non-Significant, Self-Validated Part Identification Numbers," to be published in *Engineering Management International.*

[14] Harvey, Robert E. "CAPP: Critical to CAD/CAM Success," *Iron Age*, vol 226, no. 24 (September 23, 1983), 61-69.

[15] Hazen, David C. "The Engineering Research Centers as a Tool for Change in the Culture and Attitudes of Academic Engineering," Presentation at the Second Meeting of the Steering Group on Systems Aspects of Cross Disciplinary Engineering, August 8, 1985.

[16] Kutcher, Mike and Eli Gorin. "Moving Data, not Paper, Enhances Productivity," *IEEE Spectrum*, vol. 20, no. 5 (May, 1983), 84-88.

[17] Mark, Leo and Nick Roussopoulos. "Operational Specification of Update Dependencies," Department of Computer Science, University of Maryland.

[18] Melkanoff, Michael. "The CIMS Database: Goals, Problems, Case Studies, and Proposed Approaches Outlined," *Industrial Engineering*, vol. 16, no 11 (November, 1984), 78-92.

[19] Schmuland, Rodney H. "Managing Bill of Material and CAD/CAM Data," *Auerbach Industry Applications Series: Computer Aided Design, Engineering, and Drafting*, vol. 1 (1985), 1-14.

[20] Ssemakula, M. and B. J. Davies. "Integrated Process Planning and NC Programming for Prismatic Parts," Proceedings, First International Machine Tool Conference, IFS Publication, June 1984.

[21] Wight, Oliver. *MRP II: Unlocking America's Productivity Potential*, O.W. Ltd. Publications, 1981.

[22] Yeomans, R. W., A. Choudry. and P. J. W. ten Hagen. *Design Rules for a CIM System*, North Holland, 1985.

Appendix A

Update Dependency Operations for the Model MRP II/CAD System

```
% ********** Operations on relation cadpart
% *****


% ***** Routine to insert part records into CAD
% *****

insert(cadpart)

-->    insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).


insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum))

% *** Part has already been inserted
%
-->    nonvar(Pnum) /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
       /\ nonvar(Buom) /\ nonvar(Spnum) /\ nonvar(Sbnum)
       /\ cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum).

% *** Request part number if not provided
%
-->    var(Pnum),
       write('Part Number? '),
       read(Pnum),
       insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Request description if not provided
```

```
%
-->   var(Des)
      /\ nonvar(Pnum),
      write('Description? '),
      read(Des),
      insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Request unit of measure if not provided
%
-->   var(Buom)
      /\ nonvar(Pnum) /\ nonvar(Des),
      write('Unit of Measure? '),
      read(Buom),
      insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Let drawing number be unknown if not provided
%
-->   var(Dnum)
      /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom),
      insert(cadpart(Pnum,unknown,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Let drawing size be unknown if not provided
%
-->   var(Dsize)
      /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom)
      /\ nonvar(Dnum),
      insert(cadpart(Pnum,Dnum,unknown,Des,Buom,Spnum,Sbnum)).

% *** Let supersedes part number be unknown if not provided
%
-->   var(Spnum)
      /\ nonvar(Buom) /\ nonvar(Pnum) /\ nonvar(Des)
      /\ nonvar(Dnum) /\ nonvar(Dsize),
      insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,unknown,Sbnum)).

% *** Let superseded by part number be unknown if not provided
%
-->   var(Sbnum)
      /\ nonvar(Buom) /\ nonvar(Pnum) /\ nonvar(Des)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Spnum),
      insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,unknown)).

% *** Same part number with different attributes already exists
%
-->   nonvar(Pnum) /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
```

```
      /\ nonvar(Buom) /\ nonvar(Spnum) /\ nonvar(Sbnum)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      write('Part Number Already Exists in CAD'),
      nl,
      fail.

% *** Part being added via CAD
%
-->   nonvar(Pnum) /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Spnum) /\ nonvar(Sbnum)
      /\ ~(cadpart(Pnum,_,_,_,_,_,_))
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_)),
      add(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      insert(cadrev(Pnum,Rev,unknown,unknown,w,Dfname)),
      write('Part has been added to CAD'),
      nl.

% *** Part being added via MRP II
%
-->   nonvar(Pnum) /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Spnum) /\ nonvar(Sbnum)
      /\ ~(cadpart(Pnum,_,_,_,_,_,_))
      /\ mrppmr(Pnum,Dnum,Dsize,Des,Buom,_,_,_,_,_,Spnum,Sbnum),
      add(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      checksup(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      write('Part has been added to CAD'),
      nl.



% ***** Routine to delete a part from CAD
% *****

delete(cadpart)

-->   delete(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).


delete(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum))

% *** Stop if part does not exist
%
-->   nonvar(Pnum)
      /\ ~(cadpart(Pnum,_,_,_,_,_,_)).
```

```
% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part Number? '),
      read(Pnum),
      delete(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Part is used in a product structure, cannot be deleted
%
-->   nonvar(Pnum)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~findnone(cadcomponent(Pnum,_,_,_,_)),
      write('Part is used in a Structure--cannot be deleted'),
      nl,
      fail.

% *** Delete revision records associated with the part
%
-->   nonvar(Pnum)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ cadrev(Pnum,Rev,_,_,_,_)
      /\ findnone(cadcomponent(Pnum,_,_,_,_)),
      delete(cadrev(Pnum,Rev,_,_,_,_)),
      delete(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Delete the part from CAD
%
-->   nonvar(Pnum)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~(cadrev(Pnum,_,_,_,_,_))
      /\ findnone(cadcomponent(Pnum,_,_,_,_)),
      remove(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      delete(latestcadrev(Pnum,_)),
      delete(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)),
      write('Part has been deleted from CAD'),
      nl.



% ***** Routine to modify fields in CAD part records
% *****

modify(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum))
```

```
% *** Stop if desired part record is the same as existing one
%
-->   nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Spnum) /\ nonvar(Sbnum)
      /\ nonvar(Pnum)
      /\ cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum).

% *** Stop if part number is not provided
%
-->   var(Pnum).

% *** Stop if part does not exist in CAD
%
-->   nonvar(Pnum)
      /\ ~(cadpart(Pnum,_,_,_,_,_,_)).

% *** Instantiate drawing number to current value if not provided
%
-->   var(Dnum)
      /\ nonvar(Pnum)
      /\ cadpart(Pnum,Dnum,_,_,_,_,_),
      modify(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Instantiate drawing size to current value if not provided
%
-->   var(Dsize)
      /\ nonvar(Dnum)
      /\ nonvar(Pnum)
      /\ cadpart(Pnum,_,Dsize,_,_,_,_),
      modify(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Instantiate description to current value if not provided
%
-->   var(Des)
      /\ nonvar(Dnum) /\ nonvar(Dsize)
      /\ nonvar(Pnum)
      /\ cadpart(Pnum,_,_,Des,_,_,_),
      modify(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Instantiate BOM unit of measure to current value if
% *** not provided
%
-->   var(Buom)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Pnum)
```

```
      /\ cadpart(Pnum,_,_,_,_,Buom,_,_),
      modify(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Instantiate supersedes part number to current value if
% *** not provided
%
-->   var(Spnum)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Pnum)
      /\ cadpart(Pnum,_,_,_,_,_,Spnum,_),
      modify(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Instantiate superseded by part number to current value if
% *** not provided
%
-->   var(Sbnum)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Spnum) /\ nonvar(Pnum)
      /\ cadpart(Pnum,_,_,_,_,_,_,Sbnum),
      modify(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Modify part record
%
-->   nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des) /\ nonvar(Buom)
      /\ nonvar(Spnum) /\ nonvar(Sbnum)
      /\ nonvar(Pnum)
      /\ cadpart(Pnum,_,_,_,_,_,_,_)
      /\ ~(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      remove(cadpart(Pnum,_,_,_,_,_,_,_)),
      add(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,_,_,_,_,_,
             Spnum,Sbnum)).



% ***** Internal routine to perform supersession of parts
% ***** in CAD
% *****

checksup(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum))

% *** Part does not supersede another
%
-->   cadpart(Pnum,_,_,_,_,_,unknown,_).
```

```
% *** Superseded part does not exist in CAD
%
-->    cadpart(Pnum,_,_,_,_,_,_)
       /\ ~(cadpart(Pnum,_,_,_,_,unknown,_))
       /\ ~(cadpart(Spnum,_,_,_,_,_,_)),
       write('Superceded Part Number Does Not Exist in CAD'),
       nl,
       fail.

% *** Perform supersession
%
-->    cadpart(Pnum,_,_,_,_,_,_)
       /\ ~(cadpart(Pnum,_,_,_,_,unknown,_))
       /\ cadpart(Spnum,_,_,_,_,_,_),
       modify(cadpart(Spnum,_,_,_,_,_,Pnum)),
       substitutepartcad(Spnum,Pnum),
       obsolete(cadpart(Spnum,_,_,_,_,_,_)).



% ***** Routine to make a part obsolete in CAD
% *****

obsolete(cadpart)

-->    obsolete(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).


obsolete(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum))

% *** Request part number if not provided
%
-->    var(Pnum),
       write('Part number? '),
       read(Pnum),
       obsolete(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).

% *** Part number as entered does not exist in CAD
%
-->    nonvar(Pnum)
       /\ ~cadpart(Pnum,_,_,_,_,_,_),
       write('Part does not exist in CAD'),
       nl,
       fail.
```

```
% *** Make all released or hold revisions obsolete
%
-->    nonvar(Pnum)
       /\ cadpart(Pnum,_,_,_,_,_,_),
       checkrev(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)),
       write('Part has been made obsolete in CAD'),
       nl.
```

```
% ********** Operations on relation cadrev
% *****


% ***** Routine to insert revision records into CAD
% *****

insert(cadrev)

-->    insert(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).


insert(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname))

% *** Revision has already been inserted
%
-->    nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
       /\ nonvar(Eend)  /\ nonvar(Dfname)
       /\ cadrev(Pnum,Rev,Estart,Eend,Dfname).

% *** Request part number if not provided
%
-->    var(Pnum),
       write('Part Number? '),
       read(Pnum),
       insert(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Request revision level if not provided
%
-->    var(Rev)
       /\ nonvar(Pnum),
       write('New Revision Level? '),
       read(Rev),
       insert(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Request drawing file name if not provided
%
-->    var(Dfname)
       /\ nonvar(Pnum) /\ nonvar(Rev),
       write('Drawing File Name? '),
       read(Dfname),
       insert(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Let effectivity start date be unknown if not provided
%
```

```
-->   var(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Dfname),
      insert(cadrev(Pnum,Rev,unknown,Eend,Cstat,Dfname)).

% *** Let effectivity end date be unknown if not provided
%
-->   var(Eend)
      /\ nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Dfname)
      /\ nonvar(Estart),
      insert(cadrev(Pnum,Rev,Estart,unknown,Cstat,Dfname)).

% *** Part number entered does not exist in CAD
%
-->   nonvar(Pnum)
      /\ nonvar(Rev) /\ nonvar(Estart) /\ nonvar(Eend)
      /\ nonvar(Dfname)
      /\ ~(cadpart(Pnum,_,_,_,_,_,_)),
      write('Part Does Not Exist in CAD'),
      nl,
      fail.

% *** Same revision level exists with different attributes
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ nonvar(Eend) /\ nonvar(Dfname)
      /\ cadrev(Pnum,Rev,_,_,_,_)
      /\ ~(cadrev(Pnum,Rev,Estart,Eend,_,Dfname)),
      write('Revision Level Already Exists in CAD'),
      nl,
      fail.

% *** Revision is being added via CAD
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ nonvar(Eend) /\ nonvar(Dfname)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,_))
      /\ ~(mrprev(Pnum,Rev,_,_,_)),
      add(cadrev(Pnum,Rev,Estart,Eend,w,Dfname)),
      copybom(cadrev(Pnum,Rev,_,_,_,_)),
      write('Revision has been added to CAD'),
      nl.

% *** Revision is being added via MRP II
%
```

```
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ nonvar(Eend) /\ nonvar(Dfname)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,_))
      /\ mrprev(Pnum,Rev,Estart,Eend,Mstat),
      checkrev(cadrev(Pnum,Rev,Estart,Eend,Mstat,Dfname)),
      add(cadrev(Pnum,Rev,Estart,Eend,r,Dfname)),
      make(latestcadrev(Pnum,Rev)),
      write('Revision has been added to CAD'),
      nl.



% ***** Internal routine to delete a specific revision of a part
% ***** in CAD
% *****

delete(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname))

% *** Stop if revision level does not exist
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ ~(cadrev(Pnum,Rev,_,_,_,_)).

% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part Number? '),
      read(Pnum),
      delete(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Request revision level if not provided
%
-->   var(Rev)
      /\ nonvar(Pnum),
      write('Revision Level? '),
      read(Rev),
      delete(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Delete an MRP II-generated revision being deleted
% *** via MRP II
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,_,inapp)
      /\ ~(mrprev(Pnum,Rev,_,_,_)),
```

```
      remove(cadrev(Pnum,Rev,_,_,_,inapp)).

% *** CAD user trying to delete MRP II-generated part,
% *** not allowed
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,_,inapp)
      /\ mrprev(Pnum,Rev,_,_,_),
      write('Revision can only be deleted via MRP II'),
      nl,
      fail.

% *** Delete revision has working status
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,w,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,inapp)),
      remove(cadrev(Pnum,Rev,_,_,w,_)),
      write('Revision information has been deleted from CAD'),
      nl.

% *** Revision has released status in CAD, cannot be deleted
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,r,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,inapp)),
      write('Revision has Released Status--Cannot be Deleted'),
      nl,
      fail.

% *** Revision has hold status in CAD, cannot be deleted
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,h,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,inapp)),
      write('Revision has Hold Status--Cannot be Deleted'),
      nl,
      fail.

% *** Delete revision from CAD
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,o,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,inapp)),
      remove(cadrev(Pnum,Rev,_,_,_,_)),
```

```
      delete(mrprev(Pnum,Rev,_,_,_)),
      write('Revision has been deleted from CAD'),
      nl.


% ***** Internal routine to modify a cad revision record
% *****

modify(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname))

% *** Stop if desired record is the same as existing one
%
-->   nonvar(Estart) /\ nonvar(Eend) /\ nonvar(Cstat)
      /\ nonvar(Dfname) /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname).

% *** Stop if part number is not provided
%
-->   var(Pnum).

% *** Stop if revision level is not provided
%
-->   var(Rev)
      /\ nonvar(Pnum).

% *** Stop if revivision does not exist
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ ~(cadrev(Pnum,Rev,_,_,_,_)).

% *** Instantiate effectivity start date to current value if
% *** not provided
%
-->   var(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,Estart,_,_,_),
      modify(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Instantiate effectivity end date to current value if
% *** not provided
%
-->   var(Eend)
      /\ nonvar(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev)
```

```
      /\ cadrev(Pnum,Rev,_,Eend,_,_),
      modify(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Instantiate Cad status to current value if not provided
%
-->   var(Cstat)
      /\ nonvar(Estart) /\ nonvar(Eend)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,Cstat,_),
      modify(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Instantiate drawing file to current value if not provided
%
-->   var(Dfname)
      /\ nonvar(Estart) /\ nonvar(Eend) /\ nonvar(Cstat)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,_,Dfname),
      modify(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Make modification to revision record
%
-->   nonvar(Estart) /\ nonvar(Eend) /\ nonvar(Cstat)
      /\ nonvar(Dfname) /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,_,_)
      /\ ~(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)),
      remove(cadrev(Pnum,Rev,_,_,_,_)),
      add(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)),
      modify(mrprev(Pnum,Rev,Estart,Eend,_)).



% ***** Routine to release a working revision record in CAD
% *****

releasework(cadrev)

-->   releasework(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).


releasework(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname))

% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part Number? '),
```

```
      read(Pnum),
      releasework(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Request revision level if not provided
%
-->   var(Rev)
      /\ nonvar(Pnum),
      write('Revision Level? '),
      read(Rev),
      releasework(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Part entered does not exist in CAD
%
-->   nonvar(Pnum)
      /\ nonvar(Rev)
      /\ ~(cadpart(Pnum,_,_,_,_,_,_)),
      write('Part Number Does Not Exist in CAD'),
      nl,
      fail.

% *** Revision level entered does not exist in CAD
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,_)),
      write('Revision Level Does Not Exist in CAD'),
      nl,
      fail.

% *** Revision level entered does not have working status
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,_,_)
      /\ ~(cadrev(Pnum,Rev,_,_,w,_)),
      write('Part Does Not Have Working Status'),
      nl,
      fail.

% *** Release first version of a part
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,Estart,Eend,w,Dfname)
      /\ cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)),
      checkrev(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)),
```

```
      make(latestcadrev(Pnum,Rev)),
      modify(cadrev(Pnum,Rev,_,_,r,_)),
      insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,unknown,unknown,
      unknown,unknown,unknown,Spnum,Sbnum)),
      insert(newpmr(Pnum)),
      insert(mrprev(Pnum,Rev,unknown,unknown,h)),
      insert(newrev(Pnum,Rev)),
      checksup(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      release(cadcomponent(Pnum,Rev,_,_,_)),
      write('Revision has been released in CAD'),
      nl.

% *** Release subsequent revisions to a part
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,Estart,Eend,w,Dfname)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_),
      checkrev(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)),
      make(latestcadrev(Pnum,Rev)),
      modify(cadrev(Pnum,Rev,_,_,r,_)),
      insert(mrprev(Pnum,Rev,Estart,Eend,h)),
      insert(newrev(Pnum,Rev)),
      release(cadcomponent(Pnum,Rev,_,_,_)),
      write('Revision has been released in CAD'),
      nl.



% ***** Internal routine to process obsolescence of old revisions
% ***** when a new revision is released or inserted in CAD
% *****

checkrev(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname))

% *** Make released revisions obsolete
%
-->   cadrev(Pnum,R,_,_,r,_),
      obsolete(cadrev(Pnum,R,_,_,_,_)),
      checkrev(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Make hold revisions obsolete
%
-->   cadrev(Pnum,R,_,_,h,_)
      /\ ~(cadrev(Pnum,_,_,_,r,_)),
      obsolete(cadrev(Pnum,R,_,_,_,_)),
```

```
      checkrev(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** All revisions have been made obsolete
%
-->   ~(cadrev(Pnum,_,_,_,r,_))
      /\ ~(cadrev(Pnum,_,_,_,h,_)).



% ***** Internal routine to copy bills of material from the
% ***** previous revision of an assembly to a new one in CAD
% *****

copybom(cadrev)

-->   copybom(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

copybom(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname))
% *** Stop if part number not provided
%
-->   var(Pnum).

% *** Stop if revision level not provided
%
-->   var(Rev)
      /\ nonvar(Pnum).

% *** Stop if the part revision is the first one for the part
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ ~latestcadrev(Pnum,_).

% *** Stop if the part revision is the latest one
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ latestcadrev(Pnum,R)
      /\ R=Rev.

% *** copy the bom from the last revision to the new one
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ latestcadrev(Pnum,R)
      /\ ~(R=Rev),
      copybomcad(Pnum,R,Pnum,Rev).
```

```
% ***** Routine to place a hold on a part in CAD
% *****

hold(cadrev)

-->   hold(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).


hold(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname))

% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part Number? '),
      read(Pnum),
      hold(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Request revision level if not provided
%
-->   var(Rev)
      /\ nonvar(Pnum),
      write('Revision Level? '),
      read(Rev),
      hold(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Part entered does not exist in CAD
%
-->   nonvar(Pnum)
      /\ nonvar(Rev)
      /\ ~(cadpart(Pnum,_,_,_,_,_,_)),
      write('Part Does Not Exist in CAD'),
      nl,
      fail.

% *** Revision level entered does not exist in CAD
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,_)),
      write('Revision Level Does Not Exist in CAD'),
      nl,
      fail.
```

```
% *** Revision is already on hold in CAD
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,h,_),
      write('Revision Already Has Hold Status'),
      nl,
      fail.

% *** Revision has working status
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,w,_),
      write('Revision Has Working Status'),
      nl,
      fail.

% *** Revision has obsolete status
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,o,_),
      write('Revision Has Obsolete Status'),
      nl,
      fail.

% *** Give revision hold status
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,r,_),
      hold(mrprev(Pnum,Rev,_,_,h)),
      modify(cadrev(Pnum,Rev,_,_,h,_)),
      write('Revision has been given hold status in CAD'),
      nl.



% ***** Routine to release a part on hold in CAD
% *****

releasehold(cadrev)

-->   releasehold(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).


releasehold(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname))
```

```
% *** Request part number if not provided
%
-->    var(Pnum),
       write('Part Number? '),
       read(Pnum),
       releasehold(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Request revision level if not provided
%
-->    var(Rev)
       /\ nonvar(Pnum),
       write('Revision Level? '),
       read(Rev),
       releasehold(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname)).

% *** Part entered does not exist in CAD
%
-->    nonvar(Pnum)
       /\ nonvar(Rev)
       /\ ~(cadpart(Pnum,_,_,_,_,_,_)),
       write('Part Does Not Exist in CAD'),
       nl,
       fail.

% *** Revision level entered does not exist in CAD
%
-->    nonvar(Pnum) /\ nonvar(Rev)
       /\ cadpart(Pnum,_,_,_,_,_,_)
       /\ ~(cadrev(Pnum,Rev,_,_,_,_)),
       write('Revision Level Does Not Exist in CAD'),
       nl,
       fail.

% *** Revision is not on hold
%
-->    nonvar(Pnum) /\ nonvar(Rev)
       /\ cadrev(Pnum,Rev,_,_,_,_)
       /\ ~(cadrev(Pnum,Rev,_,_,h,_)),
       write('Revision Not On Hold'),
       nl,
       fail.

% *** Rerelease revision
%
-->    nonvar(Pnum) /\ nonvar(Rev)
```

```
      /\ cadrev(Pnum,Rev,_,_,h,_),
      modify(cadrev(Pnum,Rev,_,_,r,_)),
      insert(rereleased(Pnum,Rev)),
      write('Revision has been rereleased in CAD'),
      nl.



% ***** Internal routine to make a specific revision of a part
% ***** obsolete in CAD
% *****

obsolete(cadrev(Pnum,Rev,Estart,Eend,Cstat,Dfname))

% *** Stop if part number if not provided
%
-->   var(Pnum).

% *** Stop if revision level not provided
%
-->   var(Rev)
      /\ nonvar(Pnum).

% *** Stop if part entered does not exist in CAD
%
-->   nonvar(Pnum)
      /\ nonvar(Rev)
      /\ ~(cadpart(Pnum,_,_,_,_,_,_)).

% *** Stop if revision level entered does not exist in CAD
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,_)).

% *** Stop if revision entered has working status, cannot be
% *** made obsolete
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,w,_).

% *** Give revision obsolete status
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ cadrev(Pnum,Rev,_,_,_,_)
```

```
/\ ~(cadrev(Pnum,Rev,_,_,w,_)),
modify(cadrev(Pnum,Rev,_,_,o,_)),
insert(obsolete(Pnum,Rev)),
delete(latestcadrev(Pnum,Rev)),
write('Revision has been given obsolete status in CAD'),
nl.
```

```
% ********** Operations on relation latestcadrev
% *****


% ***** Internal routine to update the latest revision record
% ***** in CAD
% *****

make(latestcadrev(Pnum,Rev))

% *** Stop if record already exists
%
-->   latestcadrev(Pnum,Rev).

% *** Update existing record for new revision
%
-->   latestcadrev(Pnum,R)
      /\ ~latestcadrev(Pnum,Rev),
      delete(latestcadrev(Pnum,R)),
      insert(latestcadrev(Pnum,Rev)).

% *** Insert record for first revision
%
--> ~latestcadrev(Pnum,R),
      insert(latestcadrev(Pnum,Rev)).



% ***** Internal routine to insert latest revision record
% ***** in CAD
% *****

insert(latestcadrev(Pnum,Rev))

% *** Stop if record already exists
%
-->   latestcadrev(Pnum,Rev).

% *** insert record
%
-->   ~latestcadrev(Pnum,Rev),
      add(latestcadrev(Pnum,Rev)).
```

```
% ***** Internal routine to delete a latest revision record
% ***** from CAD
% *****

delete(latestcadrev(Pnum,Rev))

% *** Stop if record does not exist
%
-->    ~latestcadrev(Pnum,Rev).

% *** Delete record
%
-->    latestcadrev(Pnum,Rev),
       remove(latestcadrev(Pnum,Rev)).
```

```
% ********** Operations on relation cadcomponent
% *****


% ***** Routine to insert a relationship record into CAD
% ***** (top level)

insert(cadcomponent)

-->     insert(cadcomponent(Pnum,Rev,Item,Cpn,Qty)).


insert(cadcomponent(Pnum,Rev,Item,Cpn,Qty))

% *** Assembly number and revision specified, has working status,
% *** don't consider part/revision changes
%
-->    nonvar(Pnum) /\ nonvar(Rev)
       /\ cadrev(Pnum,Rev,_,_,w,_),
       ecnheader(Pnum,Rev),
       insert2(cadcomponent(Pnum,Rev,Item,Cpn,Qty)).

% *** Assembly number and revision specified, does not have
% *** working status, consider part/revision changes
%
-->    nonvar(Pnum) /\ nonvar(Rev)
       /\ ~cadrev(Pnum,Rev,_,_,w,_),
       ecnheader(Pnum,Rev),
       checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
       insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Assembly revision specified, consider part/revision changes
%
-->    var(Pnum) /\ nonvar(Rev),
       ecnheader(Pnum,Rev),
       checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
       insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Assembly number specified, consider part/revision changes
%
-->    nonvar(Pnum) /\ var(Rev),
       ecnheader(Pnum,Rev),
       checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
       insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).
```

```
% *** No assembly details specified, consider part/revision
% *** changes
%
-->   var(Pnum) /\ var(Rev),
      ecnheader(Pnum,Rev),
      checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).



% ***** Internal routine to actually perform insersion of
% ***** relationship into CAD
% *****

insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty))


% *** Stop if record already exists
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ cadcomponent(Ppn,Prn,Item,Cpn,Qty).

% *** Insert relationship via MRP II
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~cadcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ mrpcomponent(Ppn,Prn,Item,Cpn,Qty),
      add(cadcomponent(Ppn,Prn,Item,Cpn,Qty)),
      write('Component Relationship has been added to CAD'),
      nl.

% *** Request parent part number if not provided
%
-->   var(Ppn),
      write('Parent part number? '),
      read(Ppn),
      insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request parent revision level if not provided
%
-->   var(Prn)
      /\ nonvar(Ppn),
      write('Parent revision level? '),
```

```
      read(Prn),
      insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request item number if not provided
%
-->   var(Item)
      /\ nonvar(Ppn) /\ nonvar(Prn),
      write('Item Number? '),
      read(Item),
      insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request component part number if not provided
%
-->   var(Cpn)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item),
      write('Component part number? '),
      read(Cpn),
      insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request Quantity per assembly if not provided
%
-->   var(Qty)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item)
      /\ nonvar(Cpn),
      write('Quantity per assembly? '),
      read(Qty),
      insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Parent part entered does not exist in CAD
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~(cadpart(Ppn,_,_,_,_,_,_)),
      write('Parent part does not exist in CAD'),
      nl,
      fail.

% *** Parent revision level does not exist in CAD
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ cadpart(Ppn,_,_,_,_,_,_)
```

```
      /\ ~(cadrev(Ppn,Prn,_,_,_,_)),
      write('Parent Revision does not exist in CAD'),
      nl,
      fail.

% *** Parent revision is obsolete, cannot be used in a structure
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,o,_),
      write('Parent Revison has obsolete status in CAD'),
      nl,
      fail.

% *** Component part entered does not exist in CAD
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,_,_)
      /\ ~(cadrev(Ppn,Prn,_,_,o,_))
      /\ ~(cadpart(Cpn,_,_,_,_,_,_)),
      write('Component part does not exist in CAD'),
      nl,
      fail.

% *** Component part does not have a released or hold revision
% *** in CAD
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,_,_)
      /\ ~(cadrev(Ppn,Prn,_,_,o,_))
      /\ cadpart(Cpn,_,_,_,_,_,_)
      /\ ~(cadrev(Cpn,_,_,_,r,_))
      /\ ~(cadrev(Cpn,_,_,_,h,_)),
      write('Component Part does not have released or hold
            revision in CAD'),
      nl,
      fail.
```

```
% *** Item number already exists in CAD, component has released
% *** revision in CAD
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,_,_)
      /\ ~(cadrev(Ppn,Prn,_,_,o,_))
      /\ cadrev(Cpn,_,_,_,r,_)
      /\ cadcomponent(Ppn,Prn,Item,_,_),
      write('Item number already exists in CAD'),
      nl,
      fail.

% *** Item number already exists in CAD, component part has a
% *** hold revision on
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,_,_)
      /\ ~(cadrev(Ppn,Prn,_,_,o,_))
      /\ cadrev(Cpn,_,_,_,h,_)
      /\ cadcomponent(Ppn,Prn,Item,_,_),
      write('Item number already exists in CAD'),
      nl,
      fail.

% *** Component has no revisions in MRP II
% *** (component has released revision in CAD)
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,w,_)
      /\ cadrev(Cpn,_,_,_,r,_)
      /\ ~(mrprev(Cpn,_,_,_,_))
      /\ ~(cadcomponent(Ppn,Prn,Item,_,_)),
      write('No Component revisions exist in MRP II'),
      nl,
      fail.
```

```
% *** Component has no revisions in MRP II
% *** (component has hold revision in CAD)
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,w,_)
      /\ cadrev(Cpn,_,_,_,h,_)
      /\ ~(mrprev(Cpn,_,_,_,_))
      /\ ~(cadcomponent(Ppn,Prn,Item,_,_)),
      write('No Component revisions exist in MRP II'),
      nl,
      fail.

% *** Insert component relationship in CAD
% *** (parent part revision has working status)
% *** (component has released revision in CAD)
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,w,_)
      /\ cadrev(Cpn,_,_,_,r,_)
      /\ mrprev(Cpn,_,_,_,_)
      /\ ~(cadcomponent(Ppn,Prn,Item,_,_)),
      add(cadcomponent(Ppn,Prn,Item,Cpn,Qty)),
      ~(makesloop(cadcomponent(Ppn,Prn,Item,Cpn,Qty))),
      write('Component relationship has been added to CAD'),
      nl.

% *** Insert component relationship in CAD
% *** (parent part revision has working status)
% *** (component part has hold revision in CAD)
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,w,_)
      /\ cadrev(Cpn,_,_,_,h,_)
      /\ mrprev(Cpn,_,_,_,_)
      /\ ~(cadcomponent(Ppn,Prn,Item,_,_)),
```

```
      add(cadcomponent(Ppn,Prn,Item,Cpn,Qty)),
      ~(makesloop(cadcomponent(Ppn,Prn,Item,Cpn,Qty))),
      write('Component relationship has been added to CAD'),
      nl.

% *** Insert component relationship in CAD
% *** (parent part revision has released or hold status)
% *** (component part has revision with released status)
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,_,_)
      /\ ~(cadrev(Ppn,Prn,_,_,o,_))
      /\ ~(cadrev(Ppn,Prn,_,_,w,_))
      /\ cadrev(Cpn,_,_,_,r,_)
      /\ ~(cadcomponent(Ppn,Prn,Item,_,_)),
      add(cadcomponent(Ppn,Prn,Item,Cpn,Qty)),
      ~(makesloop(cadcomponent(Ppn,Prn,Item,Cpn,Qty))),
      insert2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)),
      write('Component relationship has been added to CAD'),
      nl.

% *** Insert component relationship in CAD
% *** (parent part revision has released or hold status)
% *** (component part has revision with hold status)
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,Qty))
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ cadrev(Ppn,Prn,_,_,_,_)
      /\ ~(cadrev(Ppn,Prn,_,_,o,_))
      /\ ~(cadrev(Ppn,Prn,_,_,w,_))
      /\ cadrev(Cpn,_,_,_,h,_)
      /\ ~(cadcomponent(Ppn,Prn,Item,_,_)),
      add(cadcomponent(Ppn,Prn,Item,Cpn,Qty)),
      ~(makesloop(cadcomponent(Ppn,Prn,Item,Cpn,Qty))),
      insert2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)),
      write('Component relationship has been added to CAD'),
      nl.
```

```
% ***** Internal routine to check for looping of components
% ***** in CAD prior to the addition of a component
% *****

makesloop(cadcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** This operation will 'fail' if no looping is found

% *** Parent part number is the same as the component part number
%
-->   cadcomponent(Ppn,_,_,Ppn,_),
      write('Relationship results in loop, cannot be added'),
      nl.

% *** Look for relationship with parent as component of component
%
-->   cadcomponent(Cpn,_,_,Ppn,_)
      /\ ~(cadcomponent(Ppn,_,_,Ppn,_)),
      write('Relationship results in loop, cannot be added'),
      nl.

% *** No looping at this level, make a component a parent and try
% *** next level
%
-->   cadcomponent(Cpn,_,_,Com,_)
      /\ ~(cadcomponent(Cpn,_,_,Ppn,_))
      /\ ~(cadcomponent(Ppn,_,_,Ppn,_)),
      makesloop(cadcomponent(Ppn,_,_,Com,_)).



% ***** Routine to delete component relationships from CAD
% ***** (top level)
% *****

delete(cadcomponent)

-->   delete(cadcomponent(Pnum,Rev,Item,Cpn,Qty)).


delete(cadcomponent(Pnum,Rev,Item,Cpn,Qty))

% *** Parent part and revision specified, has working status,
% *** do not consider assembly part number or revision level
% *** changes
```

```
%
-->    nonvar(Pnum) /\ nonvar(Rev)
       /\ cadrev(Pnum,Rev,_,_,w,_),
       ecnheader(Pnum,Rev),
       delete2(cadcomponent(Pnum,Rev,Item,Cpn,Qty)).

% *** Parent part and revision specified, does not have
% *** working status, consider assembly part number or revision
% *** level changes
%
-->    nonvar(Pnum) /\ nonvar(Rev)
       /\ ~cadrev(Pnum,Rev,_,_,w,_),
       ecnheader(Pnum,Rev),
       checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
       delete2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Parent revision only specified, consider assembly part
% *** number or revision level changes
%
-->    var(Pnum) /\ nonvar(Rev),
       ecnheader(Pnum,Rev),
       checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
       delete2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Parent part number only specified, consider assembly part
% *** number or revision level changes
%
-->    nonvar(Pnum) /\ var(Rev),
       ecnheader(Pnum,Rev),
       checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
       delete2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** No parent part details specified, consider assembly part
% *** number or revision level changes
%
-->    var(Pnum) /\ var(Rev),
       ecnheader(Pnum,Rev),
       checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
       delete2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).



% ***** Internal routine to actually delete relationshps from CAD
% *****
```

```
delete2(cadcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** Stop if component relationship does not exist
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,_)).

% *** Request parent part number if not provided
%
-->   var(Ppn),
      write('Parent part number? '),
      read(Ppn),
      delete2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request parent revision level if not provided
%
-->   var(Prn)
      /\ nonvar(Ppn),
      write('Parent revision level? '),
      read(Prn),
      delete2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request item number if not provided
%
-->   var(Item)
      /\ nonvar(Ppn) /\ nonvar(Prn),
      write('Item number? '),
      read(Item),
      delete2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request component part number if not provided
%
-->   var(Cpn)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item),
      write('Component part number? '),
      read(Cpn),
      delete2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Delete component relationship from CAD
% *** (parent revision has working status)
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ cadcomponent(Ppn,Prn,Item,Cpn,_)
      /\ cadrev(Ppn,Prn,_,_,w,_),
      remove(cadcomponent(Ppn,Prn,Item,Cpn,_)),
```

```
      write('Relationship has been deleted from CAD'),
      nl.

% *** Delete component relaionship from CAD
% *** (parent revision does not have working status)
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ cadcomponent(Ppn,Prn,Item,Cpn,_)
      /\ cadrev(Ppn,Prn,_,_,_,_)
      /\ ~(cadrev(Ppn,Prn,_,_,w,_)),
      remove(cadcomponent(Ppn,Prn,Item,Cpn,_)),
      delete2(mrpcomponent(Ppn,Prn,Item,Cpn,_)),
      write('Relationship has been deleted from CAD'),
      nl.



% ***** Internal routine to transfer component relationships
% ***** from CAD to MRP II when the parent part is released
% *****

release(cadcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** Stop if no parent part is specified
%
-->   var(Ppn).

% *** Stop if no parent revision level is specified
%
-->   var(Prn)
      /\ nonvar(Ppn).

% *** Stop if there are no component relationships for the parent
% *** part/revision
%
-->   nonvar(Ppn) /\ nonvar(Prn)
      /\ ~(cadcomponent(Ppn,Prn,_,_,_)).

% *** Relationship local to CAD is found, copy to MRP II
%
-->   nonvar(Ppn) /\ nonvar(Prn)
      /\ cadcomponent(Ppn,Prn,_,_,_)
      /\ findrelease(cadcomponent(Ppn,Prn,It,Cp,Qt)),
      insert2(mrpcomponent(Ppn,Prn,It,Cp,Qt)),
      release(cadcomponent(Ppn,Prn,_,_,_)).
```

```
% *** No more relationships local to CAD, stop
%
-->   nonvar(Ppn) /\ nonvar(Prn)
      /\ cadcomponent(Ppn,Prn,_,_,_)
      /\ ~(findrelease(cadcomponent(Ppn,Prn,It,Cp,Qt))).



% ***** Internal routine to look for components to transfer
% ***** from CAD to MRP II when a new assembly is released
% ***** from CAD
% *****

findrelease(cadcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** Look for a component relationship of parent part/revision
% *** that does not already exist in MRP II
%
-->   cadcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ ~(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).



% ***** Internal routine to search for component relationships
% ***** involving a part to be deleted from CAD
% *****

findnone(cadcomponent(Pnum,Rev,Item,Cpn,Qty))

% *** Look for part as a parent or component in a relationship
% *** record
%
-->   ~cadcomponent(Pnum,_,_,_,_)
      /\ ~cadcomponent(_,_,_,Pnum,_).



% ***** Routine to performrm a mass substitution of one
% ***** component with another in CAD
% *****

substitutepartcad

-->   substitutepartcad(Oldpt,Newpt).
```

```
substitutepartcad(Oldpt,Newpt)

% *** Request part to be substituted if not provided
%
-->   var(Oldpt),
      write('Current part number? '),
      read(Oldpt),
      substitutepartcad(Oldpt,Newpt).

% *** Request part to substitute if not provided
%
-->   var(Newpt)
      /\ nonvar(Oldpt),
      write('Part number to substitute? '),
      read(Newpt),
      substitutepartcad(Oldpt,Newpt).

% *** Part to be substituted as entered does not exist
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ ~cadpart(Oldpt,_,_,_,_,_,_),
      write('Current part number does not exist'),
      nl,
      fail.

% *** Part to substitute as entered does not exist
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ cadpart(Oldpt,_,_,_,_,_,_)
      /\ ~cadpart(Newpt,_,_,_,_,_,_),
      write('Substitute part number does not exist'),
      nl,
      fail.

% *** Subsitute part does not have any released or hold
% *** revisions in CAD
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ cadpart(Oldpt,_,_,_,_,_,_)
      /\ cadpart(Newpt,_,_,_,_,_,_)
      /\ ~cadrev(Newpt,_,_,_,r,_)
      /\ ~cadrev(Newpt,_,_,_,h,_),
      write('Substitute part does not have released or hold
```

```
            revision in CAD'),
      nl,
      fail.

% *** Make substitution, assembly part revision has working
% *** status (part to substitute has a released revision)
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ cadpart(Oldpt,_,_,_,_,_,_)
      /\ cadpart(Newpt,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,r,_)
      /\ findsubcomp(cadcomponent(Pnum,Rev,Item,Oldpt,Qty))
      /\ cadrev(Pnum,Rev,_,_,w,_),
      delete2(cadcomponent(Pnum,Rev,Item,Oldpt,Qty)),
      insert2(cadcomponent(Pnum,Rev,Item,Newpt,Qty)),
      substitutepartcad(Oldpt,Newpt).

% *** Make substitution, assembly part revision does not have
% *** working status  (part to substitute has a released
% *** revision)
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ cadpart(Oldpt,_,_,_,_,_,_)
      /\ cadpart(Newpt,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,r,_)
      /\ findsubcomp(cadcomponent(Pnum,Rev,Item,Oldpt,Qty))
      /\ ~cadrev(Pnum,Rev,_,_,w,_),
      ecnheader(Pnum,Rev),
      checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      delete2(cadcomponent(Ppn,Prn,Item,Oldpt,Qty)),
      insert2(cadcomponent(Ppn,Prn,Item,Newpt,Qty)),
      substitutepartcad(Oldpt,Newpt).

% *** Make substitution, assembly part revision has working
% *** status  (part to substitute has a hold revision)
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ cadpart(Oldpt,_,_,_,_,_,_)
      /\ cadpart(Newpt,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,h,_)
      /\ findsubcomp(cadcomponent(Pnum,Rev,Item,Oldpt,Qty))
      /\ cadrev(Pnum,Rev,_,_,w,_),
      delete2(cadcomponent(Pnum,Rev,Item,Oldpt,Qty)),
      insert2(cadcomponent(Pnum,Rev,Item,Newpt,Qty)),
      substitutepartcad(Oldpt,Newpt).
```

```
% *** Make substitution, assembly part revision does not have
% *** working status  (part to substitute has a hold revision)
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ cadpart(Oldpt,_,_,_,_,_,_)
      /\ cadpart(Newpt,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,h,_)
      /\ findsubcomp(cadcomponent(Pnum,Rev,Item,Oldpt,Qty))
      /\ ~cadrev(Pnum,Rev,_,_,w,_),
      ecnheader(Pnum,Rev),
      checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      delete2(cadcomponent(Ppn,Prn,Item,Oldpt,Qty)),
      insert2(cadcomponent(Ppn,Prn,Item,Newpt,Qty)),
      substitutepartcad(Oldpt,Newpt).

% *** All substitutions have been made
% *** (part to substitute has released revison)
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ cadpart(Oldpt,_,_,_,_,_,_)
      /\ cadpart(Newpt,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,r,_)
      /\ ~findsubcomp(cadcomponent(Pnum,Rev,Item,Oldpt,Qty)),
      write('Part substitution has been completed'), nl.

% *** All substitutions have been made
% *** (part to substitute has hold revision)
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ cadpart(Oldpt,_,_,_,_,_,_)
      /\ cadpart(Newpt,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,h,_)
      /\ ~findsubcomp(cadcomponent(Pnum,Rev,Item,Oldpt,Qty)),
      write('Part substitution has been completed'), nl.



% ***** Internal routine to search for relationships for
% ***** mass component substitution in CAD
% *****

findsubcomp(cadcomponent(Pnum,Rev,Item,Cpn,Qty))

% *** Look for a relationship with the part to be substituted
```

```
% *** as a component, and the parent being the latest revision
% *** of a part
%
-->    cadcomponent(Pnum,Rev,Item,Cpn,Qty)
       /\ latestcadrev(Pnum,Rev).

% *** Look for a relationship with the part to be substituted
% *** as a component, and the parent having working status
%
-->    cadcomponent(Pnum,Rev,Item,Cpn,Qty)
       /\ cadrev(Pnum,Rev,_,_,w,_).



% ***** Routine to modify the quantity per assembly of a
% ***** component in CAD (top level)
% *****

modifyquantity(cadcomponent)

-->    modifyquantity(cadcomponent(Pnum,Rev,Item,Cpn,Qty)).


modifyquantity(cadcomponent(Pnum,Rev,Item,Cpn,Qty))

% *** Parent part and revision specified, has working status,
% *** do not consider assembly part or revision changes
%
-->    nonvar(Pnum) /\ nonvar(Rev)
       /\ cadrev(Pnum,Rev,_,_,w,_),
       modifyquant(cadcomponent(Pnum,Rev,Item,Cpn,Qty)).

% *** Parent part and revision specified, doesn't have working
% *** status, consider assembly part or revision changes
%
-->    nonvar(Pnum) /\ nonvar(Rev)
       /\ ~ cadrev(Pnum,Rev,_,_,w,_),
       ecnheader(Pnum,Rev),
       checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
       modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Parent revision specified, consider assembly part or
% *** revision changes
%
-->    var(Pnum) /\ nonvar(Rev),
```

```
      ecnheader(Pnum,Rev),
      checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Parent part number specified, consider assembly part or
% *** revision changes
%
-->   nonvar(Pnum) /\ var(Rev),
      ecnheader(Pnum,Rev),
      checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** No parent part details specified, consider assembly part or
% *** revsion changes
%
-->   var(Pnum) /\ var(Rev),
      ecnheader(Pnum,Rev),
      checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).



% ***** Internal routine to actually perform modification of
% ***** quantity per assembly in CAD
% *****

modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** Relationship as desired already exists
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ cadcomponent(Ppn,Prn,Item,Cpn,Qty).

% *** Request parent part number if not provided
%
-->   var(Ppn),
      write('Parent part number? '),
      read(Ppn),
      modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request parent revision level if not provided
%
-->   var(Prn)
      /\ nonvar(Ppn),
```

```
      write('Parent revision level? '),
      read(Prn),
      modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request item number if not provided
%
-->   var(Item)
      /\ nonvar(Ppn) /\ nonvar(Prn),
      write('Item number? '),
      read(Item),
      modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request component part number if not provided
%
-->   var(Cpn)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item),
      write('Component part number? '),
      read(Cpn),
      modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request new quantity per assembly if not provided
%
-->   var(Qty)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item)
      /\ nonvar(Cpn),
      write('New quantity per assembly? '),
      read(Qty),
      modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Relationship as entered does not exist
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~(cadcomponent(Ppn,Prn,Item,Cpn,_)),
      write('Component relationship does not exist'),
      nl,
      fail.

% *** Modify quantity per assembly
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ cadcomponent(Ppn,Prn,Item,Cpn,_)
      /\ ~cadcomponent(Ppn,Prn,Item,Cpn,Qty),
      remove(cadcomponent(Ppn,Prn,Item,Cpn,_)),
```

```
      add(cadcomponent(Ppn,Prn,Item,Cpn,Qty)),
      modifyquant(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)),
      write('Quantity per assembly has been changed in CAD'),
      nl.



% ***** Routine to copy a bill of material to another assembly
% ***** in CAD (top level)
% *****

copybomcad

-->   copybomcad(Fpn,Frl,Tpn,Trl).


copybomcad(Fpn,Frl,Tpn,Trl)

% *** Request part number of assembly to copy bom from if
% *** not provided
%
-->   var(Fpn),
      write('From part number? '),
      read(Fpn),
      copybomcad(Fpn,Frl,Tpn,Trl).

% *** Request revision level of assembly to copy bom from if
% *** not provided
%
-->   var(Frl)
      /\ nonvar(Fpn),
      write('From revision level? '),
      read(Frl),
      copybomcad(Fpn,Frl,Tpn,Trl).

% *** Request part number of assembly to copy bom to if
% *** not provided
%
-->   var(Tpn)
      /\ nonvar(Fpn) /\ nonvar(Frl),
      write('To part number? '),
      read(Tpn),
      copybomcad(Fpn,Frl,Tpn,Trl).

% *** Request revision level of assembly to copy bom to if
```

```
% *** not provided
%
-->   var(Trl)
      /\ nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn),
      write('To revision level? '),
      read(Trl),
      copybomcad(Fpn,Frl,Tpn,Trl).

% *** From part number does not exist in CAD
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ ~(cadpart(Fpn,_,_,_,_,_,_,_)),
      write('From part number does not exist in CAD'),
      nl,
      fail.

% *** From revision level does not exist in CAD
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ cadpart(Fpn,_,_,_,_,_,_,_)
      /\ ~cadrev(Fpn,Frl,_,_,_,_),
      write('From revision level does not exist in CAD'),
      nl,
      fail.

% *** To part number does not exist in CAD
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ cadrev(Fpn,Frl,_,_,_,_)
      /\ ~cadpart(Tpn,_,_,_,_,_,_,_),
      write('To part number does not exist in CAD'),
      nl,
      fail.

% *** To revision level does not exist
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ cadrev(Fpn,Frl,_,_,_,_)
      /\ cadpart(Tpn,_,_,_,_,_,_,_)
      /\ ~cadrev(Tpn,Trl,_,_,_,_),
      write('To revision level does not exist'),
      nl,
      fail.

% *** To part revision already has a product structure
```

```
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ cadrev(Fpn,Frl,_,_,_,_)
      /\ cadrev(Tpn,Trl,_,_,_,_)
      /\ ~findnone2(cadcomponent(Tpn,Trl,_,_,_)),
      write('To part number/revision level already has
            a structure'),
      nl,
      fail.

% *** From part revision has no product structure
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ cadrev(Fpn,Frl,_,_,_,_)
      /\ cadrev(Tpn,Trl,_,_,_,_)
      /\ findnone2(cadcomponent(Tpn,Trl,_,_,_))
      /\ ~cadcomponent(Fpn,Frl,_,_,_).

% *** copy product structure
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ cadrev(Fpn,Frl,_,_,_,_)
      /\ cadrev(Tpn,Trl,_,_,_,_)
      /\ findnone2(cadcomponent(Tpn,Trl,_,_,_))
      /\ cadcomponent(Fpn,Frl,_,_,_),
      copybomcad2(Fpn,Frl,Tpn,Trl),
      write('Product structure has been copied'),
      nl.



% ***** Internal routine to make sure the copy to assembly
% ***** has no component relationships in CAD
% *****

findnone2(cadcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** Succeeds if there are no components of the copy-to part
% *** revision
% ***
-->   ~cadcomponent(Ppn,Prn,_,_,_).



% ***** Internal routine to actually copy bill of material in CAD
```

```
% *****

copybomcad2(Fpn,Frl,Tpn,Trl)

% *** Copy a component relationship
%
-->   findbomcad(Fpn,Frl,Tpn,Trl,Item,Cpn,Qty),
      insert2(cadcomponent(Tpn,Trl,Item,Cpn,Qty)),
      copybomcad2(Fpn,Frl,Tpn,Trl).

% *** All relationships have been copied
%
-->   ~findbomcad(Fpn,Frl,Tpn,Trl,Item,Cpn,Qty).



% ***** Internal routine to search for relationships to
% ***** copy in CAD
% *****

findbomcad(Fpn,Frl,Tpn,Trl,Item,Cpn,Qty)

% *** Look for components of the copy from assembly that are not
% *** components of the copy to assembly
%
-->   cadcomponent(Fpn,Frl,Item,Cpn,Qty)
      /\ ~cadcomponent(Tpn,Trl,Item,Cpn,Qty).
```

```
% ********** Other CAD operations without specific
% ********** relations
% *****


% ***** Internal routine to ask about engineering changes
% ***** in CAD
% *****

checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn)

% *** Ask if change necessitates a new assembly part number
%
-->   var(Newpart),
      write('Does this change require a new assembly part
            number? '),
      read(Newpart),
      checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn).

% *** Ask if change necessitates a new assembly revision level
% *** (user answered no to new part number)
%
-->   var(Newrev) /\ nonvar(Newpart) /\ ~(Newpart = 'yes'),
      write('Does this change require a new assembly revision
            level? '),
      read(Newrev),
      checkecncad(Newpart,Newrev,Pnum,Rev,Ppn,Prn).

% *** User answered yes to a new part number, so a new revision
% *** is not
% *** appropriate
%
-->   var(Newrev)
      /\ nonvar(Newpart) /\ (Newpart = 'yes'),
      checkecncad(Newpart,no,Pnum,Rev,Ppn,Prn).

% *** User answered no to new part, new revision
%
-->   nonvar(Newpart) /\ nonvar(Newrev)
      /\ ~(Newpart = 'yes') /\ ~(Newrev = 'yes'),
      Ppn=Pnum,
      Prn=Rev.

% *** Let user insert a new assembly part
%
```

```
-->    nonvar(Newpart) /\ nonvar(Newrev)
       /\ (Newpart = 'yes') /\ ~(Newrev = 'yes'),
       insert(cadpart(Ppn,Dnum,Dsize,Des,Buom,Pnum,Sbnum)),
       cadrev(Ppn,Prn,_,_,_,_),
       copybomcad(Pnum,Rev,Ppn,Prn).

% *** Let user insert a new assembly revision level
%
-->    nonvar(Newpart) /\ nonvar(Newrev)
       /\ (Newrev = 'yes')
       /\ ~(Newpart = 'yes'),
       Ppn=Pnum,
       insert(cadrev(Ppn,Prn,Estart,Eend,Cstat,Dfname)).



% ***** Internal routine to print out current assembly part
% ***** number and revision level for engineering change
% ***** consideration  (Used in both CAD and MRP II)
% *****

ecnheader(Pnum,Rev)

% *** Stop if part number and revision are not provided
%
-->    var(Pnum) /\ var(Rev).

% *** Write assembly number if only it is provided
%
-->    nonvar(Pnum) /\ var(Rev),
       nl, write('Current assembly part number: '), write(Pnum),
       nl.

% *** Write revision level if only it is provided
%
-->    var(Pnum) /\ nonvar(Rev),
       nl, write('Current assembly revision level: '), write(Rev),
       nl.

% *** Write assembly number and revision level if both are
% *** provided
%
-->    nonvar(Pnum) /\ nonvar(Rev),
       nl, write('Current assembly part number: '), write(Pnum),
       nl,
```

```
write('Current assembly revision level: '), write(Rev),
nl.
```

```
% ********** Operations on relation mrppmr
% *****


% ***** Routine to insert part master records into MRP II
% *****

insert(mrppmr)

-->   insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
        Cost,Lt,Spnum,Sbnum)).


insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,Cost,
        Lt,Spnum,Sbnum))

% *** Part master record has already been inserted
%
-->   nonvar(Pnum) /\ nonvar(Dnum) /\ nonvar(Dsize)
      /\ nonvar(Des) /\ nonvar(Buom) /\ nonvar(Spnum)
      /\ nonvar(Sbnum)
      /\ mrppmr(Pnum,Dnum,Dsize,Des,Buom,_,_,_,_,_,Spnum,Sbnum).

% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part Number? '),
      read(Pnum),
      insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Request description if not provided

-->   var(Des)
      /\ nonvar(Pnum),
      write('Description? '),
      read(Des),
      insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Request BOM unit of measure if not provided
%
-->   var(Buom)
      /\ nonvar(Pnum) /\ nonvar(Des),
      write('BOM Unit of Measure? '),
```

```
        read(Buom),
        insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
        Cost,Lt,Spnum,Sbnum)).

% *** Request purchasing/inventory unit of measure if
% *** not provided
%
-->     var(Puom)
        /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom),
        write('Purchasing/Inventory Unit of Measure? '),
        read(Puom),
        insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
        Cost,Lt,Spnum,Sbnum)).

% *** Request UOM conversion factor if not provided
%
-->     var(Cfuom)
        /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom)
        /\ nonvar(Puom),
        write('Unit of Measure Conversion Factor? '),
        read(Cfuom),
        insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
        Cost,Lt,Spnum,Sbnum)).

% *** Request source code if not provided
%
-->     var(Scode)
        /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom)
        /\ nonvar(Puom) /\ nonvar(Cfuom),
        write('Source Code? '),
        read(Scode),
        insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
        Cost,Lt,Spnum,Sbnum)).

% *** Request lead time if not provided
%
-->     var(Lt)
        /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom)
        /\ nonvar(Puom) /\ nonvar(Cfuom) /\ nonvar(Scode),
        write('Lead Time? '),
        read(Lt),
        insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
        Cost,Lt,Spnum,Sbnum)).

% *** Let drawing number be unknown if not provided
```

```
%
-->   var(Dnum)
      /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom)
      /\ nonvar(Puom) /\ nonvar(Cfuom) /\ nonvar(Scode)
      /\ nonvar(Lt),
      insert(mrppmr(Pnum,inapp,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Let drawing size be unknown if not provided
%
-->   var(Dsize)
      /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom)
      /\ nonvar(Puom) /\ nonvar(Cfuom) /\ nonvar(Scode)
      /\ nonvar(Lt) /\ nonvar(Dnum),
      insert(mrppmr(Pnum,Dnum,inapp,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Let cost be unknown if not provided
%
-->   var(Cost)
      /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom)
      /\ nonvar(Puom) /\ nonvar(Cfuom) /\ nonvar(Scode)
      /\ nonvar(Lt) /\ nonvar(Dnum) /\ nonvar(Dsize),
      insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      unknown,Lt,Spnum,Sbnum)).

% *** Let supersedes part number be unknown if not provided
%
-->   var(Spnum)
      /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom)
      /\ nonvar(Puom) /\ nonvar(Cfuom) /\ nonvar(Scode)
      /\ nonvar(Lt) /\ nonvar(Dnum) /\ nonvar(Dsize)
      /\ nonvar(Cost),
      insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,unknown,Sbnum)).

% *** Let superseded by part number be unknown if not provided
%
-->   var(Sbnum)
      /\ nonvar(Pnum) /\ nonvar(Des) /\ nonvar(Buom)
      /\ nonvar(Puom) /\ nonvar(Cfuom) /\ nonvar(Scode)
      /\ nonvar(Lt) /\ nonvar(Dnum) /\ nonvar(Dsize)
      /\ nonvar(Cost) /\ nonvar(Spnum),
      insert(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,unknown)).
```

```
% *** Same part number with different attributes already
% *** exists in MRP II
%
-->   nonvar(Pnum) /\ nonvar(Dnum) /\ nonvar(Dsize)
      /\ nonvar(Des) /\ nonvar(Buom) /\ nonvar(Puom)
      /\ nonvar(Cfuom) /\ nonvar(Scode) /\ nonvar(Cost)
      /\ nonvar(Lt) /\ nonvar(Spnum) /\ nonvar(Sbnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)),
      write('Part Number Already Exists in MRP II'),
      nl,
      fail.

% *** Part number does not exist in MRP II, but does exist
% *** in CAD with different attributes (has not been released
% ***to MRP II yet)
%
-->   nonvar(Pnum) /\ nonvar(Dnum) /\ nonvar(Dsize)
      /\ nonvar(Des) /\ nonvar(Buom) /\ nonvar(Puom)
      /\ nonvar(Cfuom) /\ nonvar(Scode) /\ nonvar(Cost)
      /\ nonvar(Lt) /\ nonvar(Spnum) /\ nonvar(Sbnum)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_))
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      write('Part Number Already Exists in CAD'),
      nl,
      fail.

% *** Part is being inserted via MRP II
%
-->   nonvar(Pnum) /\ nonvar(Dnum) /\ nonvar(Dsize)
      /\ nonvar(Des) /\ nonvar(Buom) /\ nonvar(Puom)
      /\ nonvar(Cfuom) /\ nonvar(Scode) /\ nonvar(Cost)
      /\ nonvar(Lt) /\ nonvar(Spnum) /\ nonvar(Sbnum)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_))
      /\ ~(cadpart(Pnum,_,_,_,_,_,_)),
      add(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,Cost,
      Lt,Spnum,Sbnum)),
      insert(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)),
      insert(mrprev(Pnum,Rev,Estart,Eend,Mstat)),
      checksup(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)),
      write('Part has been added to MRP II'),
```

```
      nl.

% *** Part is being inserted via the release of a CAD part
%
-->   nonvar(Pnum) /\ nonvar(Dnum) /\ nonvar(Dsize)
      /\ nonvar(Des) /\ nonvar(Buom) /\ nonvar(Puom)
      /\ nonvar(Cfuom) /\ nonvar(Scode) /\ nonvar(Cost)
      /\ nonvar(Lt) /\ nonvar(Spnum) /\ nonvar(Sbnum)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_))
      /\ cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum),
      add(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,Cost,
      Lt,Spnum,Sbnum)),
      write('Part has been added to MRP II'),
      nl.



% ***** Routine to delete a part from MRP II
% *****

delete(mrppmr)

-->   delete(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).


delete(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,Cost,
        Lt,Spnum,Sbnum))

% *** Stop if part does not exist in MRP II
%
-->   nonvar(Pnum)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)).

% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part Number? '),
      read(Pnum),
      delete(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Part is used in a product structure, cannot be deleted
%
-->   nonvar(Pnum)
```

```
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~findnone(mrpcomponent(Pnum,_,_,_,_)),
      write('Part is used in a Structure--cannot be deleted'),
      nl,
      fail.

% *** Part is on order, cannot be deleted
%
-->   nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ onorder(Pnum,_,_)
      /\ findnone(mrpcomponent(Pnum,_,_,_,_)),
      write('Part is on order--cannot be deleted'),
      nl,
      fail.

% *** Part is in inventory, cannot be deleted
%
-->   nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ inventory(Pnum,_)
      /\ findnone(mrpcomponent(Pnum,_,_,_,_))
      /\ ~(onorder(Pnum,_,_))
      /\ write('Part is in inventory--cannot be deleted'),
      nl,
      fail.

% *** Delete revision records associated with part
%
-->   nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ mrprev(Pnum,Rev,_,_,_)
      /\ findnone(mrpcomponent(Pnum,_,_,_,_))
      /\ ~(onorder(Pnum,_,_))
      /\ ~(inventory(Pnum,_)),
      delete(mrprev(Pnum,Rev,_,_,_)),
      delete(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Delete CAD-generated part from MRP II
-->   nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~(mrprev(Pnum,_,_,_,_))
      /\ ~(cadpart(Pnum,inapp,inapp,_,_,_,_))
      /\ findnone(mrpcomponent(Pnum,_,_,_,_))
```

```
      /\ ~(onorder(Pnum,_,_))
      /\ ~(inventory(Pnum,_)),
      remove(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)),
      delete(newpmr(Pnum)),
      delete(latestmrprev(Pnum,_)),
      write('Part has been deleted from MRP II'),
      nl.

% *** Delete MRP II-generated part from MRP II
%
-->   nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~(mrprev(Pnum,_,_,_,_))
      /\ cadpart(Pnum,inapp,inapp,_,_,_,_)
      /\ findnone(mrpcomponent(Pnum,_,_,_,_))
      /\ ~(onorder(Pnum,_,_))
      /\ ~(inventory(Pnum,_)),
      remove(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)),
      delete(latestmrprev(Pnum,_)),
      delete(cadpart(Pnum,_,_,_,_,_,_)),
      write('Part has been deleted from MRP II'),
      nl.



% ***** Routine to modify fields in MRP II part master record
% *****

modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,Cost,
        Lt,Spnum,Sbnum))

% *** Stop if desired part record is the same as existing one
%
-->   nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Puom) /\ nonvar(Cfuom)
      /\ nonvar(Scode) /\ nonvar(Cost) /\ nonvar(Lt)
      /\ nonvar(Spnum) /\ nonvar(Sbnum) /\ nonvar(Pnum)
      /\ mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,Cost,
        Lt,Spnum,Sbnum).

% *** Stop if part number not provided
%
-->   var(Pnum).

% *** Stop if part number does not exist in MRP II
```

```
%
-->   nonvar(Pnum)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)).

% *** Instantiate drawing number to current value if
% *** not provided
%
-->   var(Dnum)
      /\ nonvar(Pnum)
      /\ mrppmr(Pnum,Dnum,_,_,_,_,_,_,_,_,_,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Instantiate drawing size to current value if not provided
%
-->   var(Dsize)
      /\ nonvar(Dnum)
      /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,Dsize,_,_,_,_,_,_,_,_,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Instantiate description to current value if not provided
%
-->   var(Des)
      /\ nonvar(Dnum) /\ nonvar(Dsize)
      /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,Des,_,_,_,_,_,_,_,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Instantiate BOM unit of measure to current value if
% *** not provided
%
-->   var(Buom)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,Buom,_,_,_,_,_,_,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Instantiate purchasing/inventory UOM to current value if
% *** not provided
%
-->   var(Puom)
```

```
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,Puom,_,_,_,_,_,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Instantiate UOM conversion factor to current value if
% *** not provided
%
-->   var(Cfuom)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Puom) /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,Cfuom,_,_,_,_,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Instantiate source code to current value if not provided
%
-->   var(Scode)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Puom) /\ nonvar(Cfuom)
      /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,Scode,_,_,_,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Instantiate cost to current value if not provided
%
-->   var(Cost)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Puom) /\ nonvar(Cfuom)
      /\ nonvar(Scode) /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,Cost,_,_,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Instantiate lead time to current value if not provided
%
-->   var(Lt)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Puom) /\ nonvar(Cfuom)
      /\ nonvar(Scode) /\ nonvar(Cost) /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,Lt,_,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).
```

```
% *** Instantiate supersedes part number to current value if
% *** not provided
%
-->   var(Spnum)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Puom) /\ nonvar(Cfuom)
      /\ nonvar(Scode) /\ nonvar(Cost) /\ nonvar(Lt)
      /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,Spnum,_),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Instantiate superseded by part number to current value if not
% *** provided
-->   var(Sbnum)
      /\ nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Puom) /\ nonvar(Cfuom)
       /\ nonvar(Scode) /\ nonvar(Cost) /\ nonvar(Lt)
      /\ nonvar(Spnum) /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,Sbnum),
      modify(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)).

% *** Modify part record
%
-->   nonvar(Dnum) /\ nonvar(Dsize) /\ nonvar(Des)
      /\ nonvar(Buom) /\ nonvar(Puom) /\ nonvar(Cfuom)
      /\ nonvar(Scode) /\ nonvar(Cost) /\ nonvar(Lt)
      /\ nonvar(Spnum) /\ nonvar(Sbnum) /\ nonvar(Pnum)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)),
      remove(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)),
      add(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
      Cost,Lt,Spnum,Sbnum)),
      modify(cadpart(Pnum,Dnum,Dsize,Des,Buom,Spnum,Sbnum)).



% ***** Internal routine to process supersession of parts
% ***** in MRP II
% *****

checksup(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,Cost,
```

```
        Lt,Spnum,Sbnum))

% *** Part does not supersede another
%
-->     mrppmr(Pnum,_,_,_,_,_,_,_,_,_,unknown,_),
        nl.

% *** Superseded part does not exist in MRP II
%
-->     mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_)
        /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,unknown,_))
        /\ ~(mrppmr(Spnum,_,_,_,_,_,_,_,_,_,_,_)),
        write('Superceded Part Number Does Not Exist in MRP II'),
        nl,
        fail.

% *** Perform supersession
%
-->     mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_)
        /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,unknown,_))
        /\  mrppmr(Spnum,_,_,_,_,_,_,_,_,_,_,_),
        modify(mrppmr(Spnum,_,_,_,_,_,_,_,_,_,_,Pnum)),
        substitutepartmrp(Spnum,Pnum).



% ***** Internal routine to check for minimum part master data
% ***** entries prior to the release of a part in MRP II
% *****

checkentries(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,Scode,
        Cost,Lt,Spnum,Sbnum))

% *** Stop if part number not provided
%
-->     var(Pnum).

% *** Stop if the part does not exist in MRP II
%
-->     nonvar(Pnum)
        /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_)).

% *** Request purchasing/inventory unit of measure if it is
% *** unknown in the part master record
```

```
%
-->   var(Puom)
      /\ mrppmr(Pnum,_,_,_,_,unknown,_,_,_,_,_,_)
      /\ nonvar(Pnum),
      write('Purchasing/Inventory Unit of Measure? '),
      read(Puom),
      checkentries(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,
      Scode,Cost,Lt,Spnum,Sbnum)).

% *** Instantiate purchasing/inventory unit of measure if it
% *** is already in the part master record
%
-->   var(Puom)
      /\ ~(mrppmr(Pnum,_,_,_,_,unknown,_,_,_,_,_,_))
      /\ mrppmr(Pnum,_,_,_,_,Puom,_,_,_,_,_,_)
      /\ nonvar(Pnum),
      checkentries(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,
      Scode,Cost,Lt,Spnum,Sbnum)).

% *** Request UOM conversion factor if it is unknown in the
% *** part master record
%
-->   var(Cfuom)
      /\ mrppmr(Pnum,_,_,_,_,_,unknown,_,_,_,_,_)
      /\ nonvar(Puom)
      /\ nonvar(Pnum),
      write('Unit of Measure Conversion Factor? '),
      read(Cfuom),
      checkentries(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,
      Scode,Cost,Lt,Spnum,Sbnum)).

% *** Instantiate UOM conversion factor if it is already in
% *** the part master record
%
-->   var(Cfuom)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,unknown,_,_,_,_,_))
      /\ mrppmr(Pnum,_,_,_,_,_,Cfuom,_,_,_,_,_)
      /\ nonvar(Puom)
      /\ nonvar(Pnum),
      checkentries(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,
      Scode,Cost,Lt,Spnum,Sbnum)).

% *** Request source code if it is unknown in the part
% *** master record
%
```

```
-->   var(Scode)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,unknown,_,_,_,_)
      /\ nonvar(Puom) /\ nonvar(Cfuom)
      /\ nonvar(Pnum),
      write('Source Code? '),
      read(Scode),
      checkentries(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,
      Scode,Cost,Lt,Spnum,Sbnum)).

% *** Instantiate source code if it is already in the part
% *** master record
%
-->   var(Scode)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,unknown,_,_,_,_))
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,Scode,_,_,_,_)
      /\ nonvar(Puom) /\ nonvar(Cfuom)
      /\ nonvar(Pnum),
      checkentries(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,
      Scode,Cost,Lt,Spnum,Sbnum)).

% *** Request lead time if it is unknown in the part
% *** master record
%
-->   var(Lt)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,unknown,_,_)
      /\ nonvar(Puom) /\ nonvar(Cfuom) /\ nonvar(Scode)
      /\ nonvar(Pnum),
      write('Lead Time? '),
      read(Lt),
      checkentries(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,
      Scode,Cost,Lt,Spnum,Sbnum)).

% *** Instantiate lead time if it is already in the part
% *** master record
%
-->   var(Lt)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,unknown,_,_))
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,Lt,_,_)
      /\ nonvar(Puom) /\ nonvar(Cfuom) /\ nonvar(Scode)
      /\ nonvar(Pnum),
      checkentries(mrppmr(Pnum,Dnum,Dsize,Des,Buom,Puom,Cfuom,
      Scode,Cost,Lt,Spnum,Sbnum)).

% *** Update mrppmr record
%
```

```
-->    nonvar(Puom) /\ nonvar(Cfuom) /\ nonvar(Scode)
       /\ nonvar(Lt)
       /\ nonvar(Pnum)
       /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_),
       modify(mrppmr(Pnum,_,_,_,_,Puom,Cfuom,Scode,_,Lt,_,_)).
```

```
% ********** Operations on relation mrprev
% *****


% ***** Routine to insert revsion records into MRP II
% *****

insert(mrprev)
-->    insert(mrprev(Pnum,Rev,Estart,Eend,Mstat)).


insert(mrprev(Pnum,Rev,Estart,Eend,Mstat))

% *** Revision has already been inserted
%
-->    nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
       /\ nonvar(Eend) /\ nonvar(Mstat)
       /\ mrprev(Pnum,Rev,Estart,Eend,Mstat).

% *** Request part number if not provided
%
-->    var(Pnum),
       write('Part Number? '),
       read(Pnum),
       insert(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Request revision level if not provided
%
-->    var(Rev)
       /\ nonvar(Pnum),
       write('Revision Level? '),
       read(Rev),
       insert(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Request desired initial status if not provided
%
-->    var(Mstat)
       /\ nonvar(Pnum) /\ nonvar(Rev),
       write('Status Code? '),
       read(Mstat),
       insert(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Make sure status code is either r or h
%
-->    nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Mstat)
```

```
      /\ ~Mstat='r' /\ ~Mstat='h',
      write('Status must be either r or h'),
      nl,
      insert(mrprev(Pnum,Rev,Estart,Eend,Mstat2)).

% *** Request effectivity start date if not provided and
% *** status code is r
%
-->   var(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Mstat)
      /\ Mstat='r',
      write('Effectivity Start Date? '),
      read(Estart),
      insert(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Let effectivity start date be unknown if not provided
% *** and status code is h
-->   var(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Mstat)
      /\ Mstat='h',
      insert(mrprev(Pnum,Rev,unknown,Eend,Mstat)).

% *** Let effectivity end date be unknown if not provided
%
-->   var(Eend)
      /\ nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ nonvar(Mstat)
      /\ ~(~Mstat='r' /\ ~Mstat='h'),
      insert(mrprev(Pnum,Rev,Estart,unknown,Mstat)).

% *** Part entered does not exist in MRP II
%
-->   nonvar(Pnum)
      /\ nonvar(Rev) /\ nonvar(Estart) /\ nonvar(Eend)
      /\ nonvar(Mstat)
      /\ ~(~Mstat='r' /\ ~Mstat='h')
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_,_,_)),
      write('Part Number Does Not Exist in MRP II'),
      nl,
      fail.

% *** Same revision level with different attributes already
% *** exists in MRP II
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
```

```
      /\ nonvar(Eend) /\ nonvar(Mstat)
      /\ ~(~Mstat='r' /\ ~Mstat='h')
      /\ mrprev(Pnum,Rev,_,_,_)
      /\ ~(mrprev(Pnum,Rev,Estart,Eend,Mstat)),
      write('Revision Level Already Exists in MRP II'),
      nl,
      fail.

% *** MRP II user attempting to insert a revision to a
% *** CAD-generated part, not allowed
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ nonvar(Eend) /\ nonvar(Mstat)
      /\ ~(~Mstat='r' /\ ~Mstat='h')
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~mrprev(Pnum,Rev,_,_,_)
      /\ ~cadrev(Pnum,Rev,_,_,_,_)
      /\ cadpart(Pnum,_,_,_,_,_,_)
      /\ ~cadpart(Pnum,inapp,inapp,_,_,_,_),
      write('Revisions to this part may be made only by
             CAD users'),
      nl,
      fail.

% *** Revision being inserted via MRP II with hold status
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ nonvar(Eend) /\ nonvar(Mstat)
      /\ Mstat='h'
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~(mrprev(Pnum,Rev,_,_,_))
      /\ ~(cadrev(Pnum,Rev,_,_,_,_))
      /\ cadpart(Pnum,inapp,inapp,_,_,_,_),
      add(mrprev(Pnum,Rev,Estart,Eend,Mstat)),
      insert(cadrev(Pnum,Rev,Estart,Eend,r,inapp)),
      copybom(mrprev(Pnum,Rev,_,_,_)),
      write('Revision has been added to MRP II'),
      nl.

% *** Revision being inserted via MRP II with released status
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ nonvar(Eend) /\ nonvar(Mstat)
      /\ Mstat='r'
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)
```

```
      /\ ~(mrprev(Pnum,Rev,_,_,_))
      /\ ~(cadrev(Pnum,Rev,_,_,_,_))
      /\ cadpart(Pnum,inapp,inapp,_,_,_,_),
      add(mrprev(Pnum,Rev,Estart,Eend,Mstat)),
      insert(cadrev(Pnum,Rev,Estart,Eend,r,inapp)),
      copybom(mrprev(Pnum,Rev,_,_,_)),
      checkrev(mrprev(Pnum,Rev,Estart,_,_)),
      write('Revision has been added to MRP II'),
      nl.

% *** Revision being inserted via CAD
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ nonvar(Eend) /\ nonvar(Mstat)
      /\ ~(~Mstat='r' /\ ~Mstat='h')
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~(mrprev(Pnum,Rev,_,_,_))
      /\ cadrev(Pnum,Rev,_,_,_,_),
      add(mrprev(Pnum,Rev,Estart,Eend,h)),
      write('Revision has been added to MRP II'),
      nl.



% ***** Internal routine to delete a specific revision of a
% ***** part in MRP II
% *****

delete(mrprev(Pnum,Rev,Estart,Eend,Mstat))

% *** Stop if revision does not exist
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ ~(mrprev(Pnum,Rev,_,_,_)).

% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part Number? '),
      read(Pnum),
      delete(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Request revision level if not provided
%
-->   var(Rev)
```

```
      /\ nonvar(Pnum),
      write('Revision Level? '),
      read(Rev),
      delete(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Delete revision via CAD
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,_,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,_)),
      remove(mrprev(Pnum,Rev,_,_,_)),
      delete(newrev(Pnum,Rev)),
      delete(rereleased(Pnum,Rev)),
      delete(obsolete(Pnum,Rev)),
      write('Revision information has been deleted
            from MRP II'),
      nl.

% *** Delete MRP II-generated revision
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,_,_)
      /\ cadrev(Pnum,Rev,_,_,_,inapp),
      remove(mrprev(Pnum,Rev,_,_,_)),
      delete(newrev(Pnum,Rev)),
      delete(rereleased(Pnum,Rev)),
      delete(obsolete(Pnum,Rev)),
      delete(cadrev(Pnum,Rev,_,_,_,_)),
      write('Revision information has been deleted
            from MRP II'),
      nl.

% *** Revision was CAD-generated, has released status
% *** in CAD, cannot delete
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,_,_)
      /\ cadrev(Pnum,Rev,_,_,r,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,inapp)),
      write('Revision has released status in CAD--cannot
            delete'),
      nl,
      fail.

% *** Revision was CAD-generated, has hold status in CAD,
```

```
% *** cannot delete
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,_,_)
      /\ cadrev(Pnum,Rev,_,_,h,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,inapp)),
      write('Revision has hold status in CAD--cannot delete'),
      nl,
      fail.

% *** Delete CAD-generated revision from MRP II
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,_,_)
      /\ cadrev(Pnum,Rev,_,_,o,_)
      /\ ~(cadrev(Pnum,Rev,_,_,_,inapp)),
      remove(mrprev(Pnum,Rev,_,_,_)),
      delete(newrev(Pnum,Rev)),
      delete(rereleased(Pnum,Rev)),
      delete(obsolete(Pnum,Rev)),
      write('Revision information has been deleted from
             MRP II'),
      nl.



% ***** Internal routine to modify an MRP II revision record
% *****

modify(mrprev(Pnum,Rev,Estart,Eend,Mstat))

% *** Stop if desired record is the same as existing one
%
-->   nonvar(Estart) /\ nonvar(Eend) /\ nonvar(Mstat)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,Estart,Eend,Mstat).

% *** Stop if part number not provided
%
-->   var(Pnum).

% *** Stop if revision level not provided
%
-->   var(Rev)
      /\ nonvar(Pnum).
```

```
% *** Stop if revision level does not exist in MRP II
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ ~(mrprev(Pnum,Rev,_,_,_)).

% *** Instantiate effectivity start date to current value if
% *** not provided
%
-->   var(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,Estart,_,_),
      modify(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Instantiate effectivity end date to current value if
% *** not provided
%
-->   var(Eend)
      /\ nonvar(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,Eend,_),
      modify(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Instantiate MRP II status to current value if not provided
%
-->   var(Mstat)
      /\ nonvar(Estart) /\ nonvar(Eend)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,_,Mstat),
      modify(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Modify the revision record
%
-->   nonvar(Estart) /\ nonvar(Eend) /\ nonvar(Mstat)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,_,_)
      /\ ~(mrprev(Pnum,Rev,Estart,Eend,Mstat)),
      remove(mrprev(Pnum,Rev,_,_,_)),
      add(mrprev(Pnum,Rev,Estart,Eend,Mstat)),
      modify(cadrev(Pnum,Rev,Estart,Eend,_,_)).



% ***** Routine to release a revision record from hold in MRP II
% *****
```

```
releasehold(mrprev)

-->   releasehold(mrprev(Pnum,Rev,Estart,Eend,Mstat)).


releasehold(mrprev(Pnum,Rev,Estart,Eend,Mstat))

% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part Number? '),
      read(Pnum),
      releasehold(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Request revision level if not provided
%
-->   var(Rev)
      /\ nonvar(Pnum),
      write('Revision Level? '),
      read(Rev),
      releasehold(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Request effectivity date if value in record is unknown
%
-->   var(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,unknown,_,_),
      write('Effectivity Start Date? '),
      read(Estart),
      releasehold(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Instantiate effectivity start date if previously known
%
-->   var(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,Estart,_,_)
      /\ ~(mrprev(Pnum,Rev,unknown,_,_)),
      releasehold(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Part entered does not exist in MRP II
%
-->   nonvar(Pnum)
      /\ nonvar(Rev) /\ nonvar(Estart)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)),
```

```
      write('Part Number Does Not Exist in MRP II'),
      nl,
      fail.

% *** Revision level entered does not exist in MRP II
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ nonvar(Estart)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~(mrprev(Pnum,Rev,_,_,_)),
      write('Revision Level Does Not Exist in MRP II'),
      nl,
      fail.

% *** Revision level entered already has released status
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ nonvar(Estart)
      /\ mrprev(Pnum,Rev,_,_,r),
      write('Revision Has Released Status'),
      nl,
      fail.

% *** Revision level entered is on hold in CAD, cannot be
% *** released
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ nonvar(Estart)
      /\ mrprev(Pnum,Rev,_,_,h)
      /\ cadrev(Pnum,Rev,_,_,h,_),
      write('Revision has hold status in CAD--cannot
             be released'),
      nl,
      fail.

% *** Rerelease part/revision in MRP II
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ nonvar(Estart)
      /\ cadrev(Pnum,Rev,_,_,_,_)
      /\ ~cadrev(Pnum,Rev,_,_,h,_)
      /\ mrprev(Pnum,Rev,_,_,h)
      /\ latestmrprev(Pnum,Rev),
      modify(mrprev(Pnum,Rev,_,_,r)),
      delete(rereleased(Pnum,Rev)),
```

```
      write('Revision has been released in MRP II'),
      nl.

% *** Release part/revision for the first time in MRP
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ nonvar(Estart)
      /\ cadrev(Pnum,Rev,_,_,_,_)
      /\ ~(cadrev(Pnum,Rev,_,_,h,_))
      /\ mrprev(Pnum,Rev,_,_,h)
      /\ ~latestmrprev(Pnum,Rev),
      checkentries(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_,_,_)),
      checkrev(mrprev(Pnum,Rev,Estart,_,_)),
      delete(newpmr(Pnum)),
      delete(newrev(Pnum,Rev)),
      delete(rereleased(Pnum,Rev)),
      modify(mrprev(Pnum,Rev,Estart,_,r)),
      write('Revision has been released in MRP II'),
      nl.



% ***** Internal routine to make the newly released revision the
% ***** latest one in MRP II
% *****

checkrev(mrprev(Pnum,Rev,Estart,Eend,Mstat))

% *** Stop if parent part number not provided
%
-->   var(Pnum).

% *** Stop if parent revision level not provided
%
-->   var(Rev)
      /\ nonvar(Pnum).

% *** Stop if effectivity start date not provided
%
-->   var(Estart)
      /\ nonvar(Pnum) /\ nonvar(Rev).

% *** Make first version of part the latest one and put
% *** effectivity start date as effectivity end date of latest
% *** revision of part being superseded
```

```
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ ~latestmrprev(Pnum,R)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,Spnum,_)
      /\ latestmrprev(Spnum,R),
      modify(mrprev(Spnum,R,_,Estart,_)),
      make(latestmrprev(Pnum,Rev)).

% *** Make first version of part the latest one, no part being
% *** superseded
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ ~latestmrprev(Pnum,R)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,Spnum,_)
      /\ ~ latestmrprev(Spnum,R),
      make(latestmrprev(Pnum,Rev)).

% *** Make new version of part the latest one, update previous
% *** revision's effectivity end date

-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ latestmrprev(Pnum,R)
      /\ mrprev(Pnum,R,_,unknown,_),
      modify(mrprev(Pnum,R,_,Estart,_)),
      make(latestmrprev(Pnum,Rev)).

% *** Make new version of part the latest one
%
-->   nonvar(Pnum) /\ nonvar(Rev) /\ nonvar(Estart)
      /\ latestmrprev(Pnum,R)
      /\ ~mrprev(Pnum,R,_,unknown,_),
      make(latestmrprev(Pnum,Rev)).



% ***** Internal routine to copy bills of material from the
% ***** previous revision of an assembly to the new one in
% ***** MRP II
% *****

copybom(mrprev)

-->   copybom(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

copybom(mrprev(Pnum,Rev,Estart,Eend,Mstat))
```

```
% *** Stop if the part number is not provided
%
-->   var(Pnum).

% *** Stop if the revision level is not provided
%
-->   var(Rev)
      /\ nonvar(Pnum).

% *** Stop if the revision is the first for the part
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ ~latestmrprev(Pnum,_).

% *** Stop if the revision is the latest one
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ latestmrprev(Pnum,R)
      /\ R=Rev.

% *** copy the bom from the last revision to the new one
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ latestmrprev(Pnum,R)
      /\ ~(R=Rev),
      copybommrp(Pnum,R,Pnum,Rev).



% ***** Routine to place a hold on a part in MRP II
% *****

hold(mrprev)

-->   hold(mrprev(Pnum,Rev,Estart,Eend,Mstat)).


hold(mrprev(Pnum,Rev,Estart,Eend,Mstat))

% *** Revision is already on hold
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,_,h).
```

```
% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part Number? '),
      read(Pnum),
      hold(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Request revision level if not provided
%
-->   var(Rev)
      /\ nonvar(Pnum),
      write('Revision Level? '),
      read(Rev),
      hold(mrprev(Pnum,Rev,Estart,Eend,Mstat)).

% *** Part entered does not exist in MRP II
%
-->   nonvar(Pnum)
      /\ nonvar(Rev)
      /\ ~(mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_,_)),
      write('Part Does Not Exist in MRP II'),
      nl,
      fail.

% *** Revision level entered does not exist in MRP II
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ mrppmr(Pnum,_,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~(mrprev(Pnum,Rev,_,_,_)),
      write('Revision Level Does Not Exist in MRP II'),
      nl,
      fail.

% *** Give revision hold status
%
-->   nonvar(Pnum) /\ nonvar(Rev)
      /\ mrprev(Pnum,Rev,_,_,r),
      modify(mrprev(Pnum,Rev,_,_,h)),
      write('Revision has been given hold status in MRP II'),
      nl.
```

```
% ********** Operations on relation lastestmrprev
% *****


% ***** Internal routine to update the latest revision record
% ***** in MRP II
% *****

make(latestmrprev(Pnum,Rev))

% *** Stop if record already exists
%
-->   latestmrprev(Pnum,Rev).

% *** Update existing record for new revision
%
-->   latestmrprev(Pnum,R)
      /\ ~latestmrprev(Pnum,Rev),
      delete(latestmrprev(Pnum,R)),
      insert(latestmrprev(Pnum,Rev)).

% *** Insert record for first revision
%
--> ~latestmrprev(Pnum,R),
      insert(latestmrprev(Pnum,Rev)).



% ***** Internal routine to insert a latest revision record
% ***** in MRP II
% *****

insert(latestmrprev(Pnum,Rev))

% *** Stop if record already exists
%
-->   latestmrprev(Pnum,Rev).

% *** Insert record
%
-->   ~latestmrprev(Pnum,Rev),
      add(latestmrprev(Pnum,Rev)).
```

```
% ***** Internal routine to delete a latest revision record
% ***** from MRP II
% *****

delete(latestmrprev(Pnum,Rev))

% *** Stop if record does not exist
%
-->   ~latestmrprev(Pnum,Rev).

% *** Delete record
%
-->   latestmrprev(Pnum,Rev),
      remove(latestmrprev(Pnum,Rev)).
```

```
% ********** Operations on relation mrpcomponent
% *****


% ***** Routine to insert relationship records into MRP II
% ***** (top level)
% *****

insert(mrpcomponent)

-->    insert(mrpcomponent(Pnum,Rev,Item,Cpn,Qty)).


insert(mrpcomponent(Pnum,Rev,Item,Cpn,Qty))

% *** Check for assembly part number or revision change prior
% *** to insersion
%
-->    ecnheader(Pnum,Rev),
       checkecnmrp(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
       insert2(mrpcomponent(Pnum,Rev,Item,Cpn,Qty)).



% ***** Internal routine to actually perform insersion of
% ***** relationships into MRP II
% *****

insert2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** Relationship has already been inserted
%
-->    nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
       /\ nonvar(Qty)
       /\ mrpcomponent(Ppn,Prn,Item,Cpn,Qty).

% *** Request parent part number if not provided
%
-->    var(Ppn),
       write('Parent part number? '),
       read(Ppn),
       insert2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request parent revision level if not provided
%
```

```
-->   var(Prn)
      /\ nonvar(Ppn),
      write('Parent revision level? '),
      read(Prn),
      insert2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request item number if not provided
%
-->   var(Item)
      /\ nonvar(Ppn) /\ nonvar(Prn),
      write('Item Number? '),
      read(Item),
      insert2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request component part number if not provided
%
-->   var(Cpn)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item),
      write('Component part number? '),
      read(Cpn),
      insert2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request quantity per assebly if not provided
%
-->   var(Qty)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item)
      /\ nonvar(Cpn),
      write('Quantity per assembly? '),
      read(Qty),
      insert2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Parent part as entered does not exist
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item)
      /\ nonvar(Cpn) /\ nonvar(Qty)
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ ~(mrppmr(Ppn,_,_,_,_,_,_,_,_,_,_,_,_)),
      write('Parent part does not exist in MRP II'),
      nl,
      fail.

% *** Parent revision as entered does not exist
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
```

```
      /\ mrppmr(Ppn,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ ~(mrprev(Ppn,Prn,_,_,_)),
      write('Parent revision does not exist in MRP II'),
      nl,
      fail.

% *** Component part as entered does not exist
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ mrprev(Ppn,Prn,_,_,_)
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ ~(mrppmr(Cpn,_,_,_,_,_,_,_,_,_,_,_,_)),
      write('Component part number does not exist in MRP II'),
      nl,
      fail.

% *** Component part as entered has no revisions in MRP II
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ mrprev(Ppn,Prn,_,_,_)
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ mrppmr(Cpn,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~(mrprev(Cpn,_,_,_,_)),
      write('Component part has no revisions in MRP II'),
      nl,
      fail.

% *** Item number already exists
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ mrprev(Ppn,Prn,_,_,_)
      /\ mrprev(Cpn,_,_,_,_)
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty)
      /\ mrpcomponent(Ppn,Prn,Item,_,_),
      write('Item number already exists in MRP II'),
      nl,
      fail.

% *** Insert component relationship in MRP II
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
```

```
      /\ nonvar(Qty)
      /\ mrprev(Ppn,Prn,_,_,_)
      /\ mrprev(Cpn,_,_,_,_)
      /\ ~(mrpcomponent(Ppn,Prn,Item,_,_)),
      add(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)),
      ~(makesloop(mrpcomponent(Ppn,Prn,Item,Cpn,Qty))),
      insert2(cadcomponent(Ppn,Prn,Item,Cpn,Qty)),
      write('Component relationship has been added to MRP II'),
      nl.


% ***** Internal routine to check for looping of relationships
% ***** in MRP II
% *****

makesloop(mrpcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** This operation will 'fail' if no looping is found

% *** Parent and component parts are the same
%
-->   mrpcomponent(Ppn,_,_,Ppn,_),
      write('Relationship results in loop, cannot be added'),
      nl.

% *** Look for relationship with parent part as component of
% *** component part
%
-->   mrpcomponent(Cpn,_,_,Ppn,_)
      /\ ~(mrpcomponent(Ppn,_,_,Ppn,_)),
      write('Relationship results in loop, cannot be added'),
      nl.

% *** No looping found at this level, make a component a parent
% *** and look at next level
% ***
-->   mrpcomponent(Cpn,_,_,Com,_)
      /\ ~(mrpcomponent(Cpn,_,_,Ppn,_))
      /\ ~(mrpcomponent(Ppn,_,_,Ppn,_)),
      makesloop(mrpcomponent(Ppn,_,_,Com,_)).



% ***** Routine to delete component relationships from MRP II
% ***** (top level)
```

```
% *****

delete(mrpcomponent)

-->   delete(mrpcomponent(Pnum,Rev,Item,Cpn,Qty)).


delete(mrpcomponent(Pnum,Rev,Item,Cpn,Qty))

% *** Consider assembly part number and revsion level changes
% *** prior to deletion
%
-->   ecnheader(Pnum,Rev),
      checkecnmrp(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      delete2(mrpcomponent(Ppn,Prn,Item,Cpn,_)).



% ***** Internal routine to actually perform deletion  of
% ***** relationships from MRP II
% *****

delete2(mrpcomponent(Ppn,Prn,Item,Cpn,_))

% *** Stop if component relationship does not exist
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ ~(mrpcomponent(Ppn,Prn,Item,Cpn,_)).

% *** Request parent part number if not provided
%
-->   var(Ppn),
      write('Parent part number? '),
      read(Ppn),
      delete2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request parent revision level if not provided
%
-->   var(Prn)
      /\ nonvar(Ppn),
      write('Parent revision level? '),
      read(Prn),
      delete2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request item number if not provided
```

```
%
-->   var(Item)
      /\ nonvar(Ppn) /\ nonvar(Prn),
      write('Item number? '),
      read(Item),
      delete2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request component part number if not provided
%
-->   var(Cpn)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item),
      write('Component part number? '),
      read(Cpn),
      delete2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Delete component relationship from MRP II
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ mrpcomponent(Ppn,Prn,Item,Cpn,_),
      remove(mrpcomponent(Ppn,Prn,Item,Cpn,_)),
      delete2(cadcomponent(Ppn,Prn,Item,Cpn,_)),
      write('Relationship has been deleted from MRP II'),
      nl.



% ***** Internal routine to search for component relationships
% ***** involving a part to be deleted from MRP II
% *****

findnone(mrpcomponent(Pnum,Rev,Item,Cpn,Qty))

% *** Look for part as either a parent or component in a
% *** relationship
%
-->   ~mrpcomponent(Pnum,_,_,_,_)
      /\ ~mrpcomponent(_,_,_,Pnum,_).



% ***** Routine to perform mass substitution of one component
% ***** with another in MRP II
% *****

substitutepartmrp
```

```
-->    substitutepartmrp(Oldpt,Newpt).


substitutepartmrp(Oldpt,Newpt)

% *** Request part to be substituted if not provided
%
-->    var(Oldpt),
       write('Current part number? '),
       read(Oldpt),
       substitutepartmrp(Oldpt,Newpt).

% *** Request part to substitute if not provided
%
-->    var(Newpt)
       /\ nonvar(Oldpt),
       write('Part number to substitute? '),
       read(Newpt),
       substitutepartmrp(Oldpt,Newpt).

% *** Part to be substituted as entered does not exist
%
-->    nonvar(Oldpt) /\ nonvar(Newpt)
       /\ ~mrppmr(Oldpt,_,_,_,_,_,_,_,_,_,_,_,_),
       write('Current part number does not exist in MRP II'),
       nl,
       fail.

% *** Part to substitute does not exist
%
-->    nonvar(Oldpt) /\ nonvar(Newpt)
       /\ mrppmr(Oldpt,_,_,_,_,_,_,_,_,_,_,_,_)
       /\ ~mrppmr(Newpt,_,_,_,_,_,_,_,_,_,_,_,_),
       write('Substitute part number does not exist in MRP II'),
       nl,
       fail.

% *** Part to substitute has no released or hold revisions in CAD
%
-->    nonvar(Oldpt) /\ nonvar(Newpt)
       /\ mrppmr(Oldpt,_,_,_,_,_,_,_,_,_,_,_,_)
       /\ mrppmr(Newpt,_,_,_,_,_,_,_,_,_,_,_,_)
       /\ ~cadrev(Newpt,_,_,_,r,_)
       /\ ~cadrev(Newpt,_,_,_,h,_),
```

```
      write('Substitute part has no released or hold revisions
             in CAD'),
      nl,
      fail.

% *** Make a substitution, part to substitute has a released
% *** revision in
% *** CAD
% ***
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ mrppmr(Oldpt,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ mrppmr(Newpt,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,r,_)
      /\ findsubcomp(mrpcomponent(Pnum,Rev,Item,Oldpt,Qty)),
      ecnheader(Pnum,Rev),
      checkecnmrp(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      delete2(mrpcomponent(Ppn,Prn,Item,Oldpt,Qty)),
      insert2(mrpcomponent(Ppn,Prn,Item,Newpt,Qty)),
      substitutepartmrp(Oldpt,Newpt).

% *** Make a substitution, part to substitute has a hold
% *** revision in CAD
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ mrppmr(Oldpt,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ mrppmr(Newpt,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,h,_)
      /\ findsubcomp(mrpcomponent(Pnum,Rev,Item,Oldpt,Qty)),
      ecnheader(Pnum,Rev),
      checkecnmrp(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      delete2(mrpcomponent(Ppn,Prn,Item,Oldpt,Qty)),
      insert2(mrpcomponent(Ppn,Prn,Item,Newpt,Qty)),
      substitutepartmrp(Oldpt,Newpt).

% *** All substitutions made, part to substitute has a released
% *** revision in CAD
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ mrppmr(Oldpt,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ mrppmr(Newpt,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,r,_)
      /\ ~findsubcomp(mrpcomponent(Pnum,Rev,Item,Oldpt,Qty)),
      write('Part substitution has been completed'), nl.

% *** All substitutions made, part to substitute has a hold
```

```
% *** revision in CAD
%
-->   nonvar(Oldpt) /\ nonvar(Newpt)
      /\ mrppmr(Oldpt,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ mrppmr(Newpt,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ cadrev(Newpt,_,_,_,h,_)
      /\ findsubcomp(mrpcomponent(Pnum,Rev,Item,Oldpt,Qty)),
      write('Part substitution has been completed'), nl.


% ***** Internal routine to search for relationships for
% ***** mass substitution of components in MRP II
% *****

findsubcomp(mrpcomponent(Pnum,Rev,Item,Oldpt,Qty))

% *** Look for a component relationship with the part to be
% *** substituted as a component of the latest revision level
% *** of a parent part
%
-->   mrpcomponent(Pnum,Rev,Item,Oldpt,Qty)
      /\ latestmrprev(Pnum,Rev).



% ***** Routine to modify the quantity per assembly of a
% ***** component in MRP II (top level)
% *****

modifyquantity(mrpcomponent)

-->   modifyquantity(mrpcomponent(Pnum,Rev,Item,Cpn,Qty)).


modifyquantity(mrpcomponent(Pnum,Rev,Item,Cpn,Qty))

% *** Consider assembly part number and revision level changes
% *** prior to modifying the quantity per assembly
%
-->   ecnheader(Pnum,Rev),
      checkecnmrp(Newpart,Newrev,Pnum,Rev,Ppn,Prn),
      modifyquant(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).
```

```
% ***** Internal routine to actually perform modification of
% ***** quantity per assembly in MRP II
% *****

modifyquant(mrpcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** Desired relationship already exists
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ mrpcomponent(Ppn,Prn,Item,Cpn,Qty).

% *** Request parent part number if not provided
%
-->   var(Ppn),
      write('Parent part number? '),
      read(Ppn),
      modifyquant(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request parent revision level if not provided
%
-->   var(Prn)
      /\ nonvar(Ppn),
      write('Parent revision level? '),
      read(Prn),
      modifyquant(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request item number if not provided
%
-->   var(Item)
      /\ nonvar(Ppn) /\ nonvar(Prn),
      write('Item number? '),
      read(Item),
      modifyquant(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request component part number if not provided
%
-->   var(Cpn)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item),
      write('Component part number? '),
      read(Cpn),
      modifyquant(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Request new quantity per assembly if not provided
%
```

```
-->   var(Qty)
      /\ nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item)
      /\ nonvar(Cpn),
      write('New quantity per assembly? '),
      read(Qty),
      modifyquant(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)).

% *** Stop if parent part revision has a working status in CAD
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ cadrev(Ppn,Prn,_,_,w,_).

% *** Component relationship as entered does not exist in MRP II
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~cadrev(Ppn,Prn,_,_,w,_)
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,_),
      write('Component relationship does not exist in MRP II'),
      nl,
      fail.

% *** Modify quantity in relationship
%
-->   nonvar(Ppn) /\ nonvar(Prn) /\ nonvar(Item) /\ nonvar(Cpn)
      /\ nonvar(Qty)
      /\ ~cadrev(Ppn,Prn,_,_,w,_)
      /\ mrpcomponent(Ppn,Prn,Item,Cpn,_)
      /\ ~mrpcomponent(Ppn,Prn,Item,Cpn,Qty),
      remove(mrpcomponent(Ppn,Prn,Item,Cpn,_)),
      add(mrpcomponent(Ppn,Prn,Item,Cpn,Qty)),
      modifyquant(cadcomponent(Ppn,Prn,Item,Cpn,Qty)),
      write('Quantity per assembly has been changed in MRP II'),
      nl.



% ***** Routine to copy a bill of material to another assembly
% ***** in MRP II (top level)
% *****

copybommrp

-->   copybommrp(Fpn,Frl,Tpn,Trl).
```

```
copybommrp(Fpn,Frl,Tpn,Trl)

% *** Request part number of assembly to copy bom from if
% *** not provided
%
-->    var(Fpn),
       write('From part number? '),
       read(Fpn),
       copybommrp(Fpn,Frl,Tpn,Trl).

% *** Request revision level of assembly to copy bom from if
% *** not provided
%
-->    var(Frl)
       /\ nonvar(Fpn),
       write('From revision level? '),
       read(Frl),
       copybommrp(Fpn,Frl,Tpn,Trl).

% *** Request part number of assembly to copy bom to if
% *** not provided
%
-->    var(Tpn)
       /\ nonvar(Fpn) /\ nonvar(Frl),
       write('To part number? '),
       read(Tpn),
       copybommrp(Fpn,Frl,Tpn,Trl).

% *** Request revision level of assembly to copy bom to if
% *** not provided
%
-->    var(Trl)
       /\ nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn),
       write('To revision level? '),
       read(Trl),
       copybommrp(Fpn,Frl,Tpn,Trl).

% *** From part number does not exist in MRP II
%
-->    nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
       /\ ~(mrppmr(Fpn,_,_,_,_,_,_,_,_,_,_)),
       write('From part number does not exist in MRP II'),
       nl,
```

```
      fail.

% *** From revision level does not exist in MRP II
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ mrppmr(Fpn,_,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~mrprev(Fpn,Frl,_,_,_),
      write('From revision level does not exist in MRP II'),
      nl,
      fail.

% *** To part number does not exist in MRP II
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ cadrev(Fpn,Frl,_,_,_)
      /\ ~mrppmr(Tpn,_,_,_,_,_,_,_,_,_,_,_,_,_),
      write('To part number does not exist in MRP II'),
      nl,
      fail.

% *** To revision level does not exist in MRP II
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ mrprev(Fpn,Frl,_,_,_)
      /\ mrppmr(Tpn,_,_,_,_,_,_,_,_,_,_,_,_,_)
      /\ ~mrprev(Tpn,Trl,_,_,_),
      write('To revision level does not exist in MRP II'),
      nl,
      fail.

% *** To part revision already has a structure
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ mrprev(Fpn,Frl,_,_,_)
      /\ mrprev(Tpn,Trl,_,_,_)
      /\ ~findnone2(mrpcomponent(Tpn,Trl,_,_,_)),
      write('To part number/revision level already has
           a structure'),
      nl,
      fail.

% *** From part revision has no bom
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ mrprev(Fpn,Frl,_,_,_)
```

```
      /\ mrprev(Tpn,Trl,_,_,_)
      /\ findnone2(mrpcomponent(Tpn,Trl,_,_,_))
      /\ ~mrpcomponent(Fpn,Frl,_,_,_).

% *** copy relationships
%
-->   nonvar(Fpn) /\ nonvar(Frl) /\ nonvar(Tpn) /\ nonvar(Trl)
      /\ mrprev(Fpn,Frl,_,_,_)
      /\ mrprev(Tpn,Trl,_,_,_)
      /\ findnone2(mrpcomponent(Tpn,Trl,_,_,_))
      /\ mrpcomponent(Fpn,Frl,_,_,_),
      copybommrp2(Fpn,Frl,Tpn,Trl),
      write('Product structure has been copied'),
      nl.



% ***** Internal routine to check that the copy to assembly
% ***** has no component relations before the copy
% *****

findnone2(mrpcomponent(Ppn,Prn,Item,Cpn,Qty))

% *** succeeds if the to part revision has no components
%
--> ~mrpcomponent(Ppn,Prn,_,_,_).


% ***** Internal routine to actually copy bills of material
% ***** in MRP II
% *****

copybommrp2(Fpn,Frl,Tpn,Trl)

% *** copy a relationship
%
-->   findbommrp(Fpn,Frl,Tpn,Trl,Item,Cpn,Qty),
      insert2(mrpcomponent(Tpn,Trl,Item,Cpn,Qty)),
      copybommrp2(Fpn,Frl,Tpn,Trl).

% *** All relationships have been copied
-->   ~findbommrp(Fpn,Frl,Tpn,Trl,Item,Cpn,Qty).
```

```
% ***** Internal routine to search for relationships to
% ***** copy in MRP II
% *****

findbommrp(Fpn,Frl,Tpn,Trl,Item,Cpn,Qty)

% *** Look for components of from part revision that are not
% *** components of the to part revision
%
-->    mrpcomponent(Fpn,Frl,Item,Cpn,Qty)
       /\ ~mrpcomponent(Tpn,Trl,Item,Cpn,Qty).
```

```
% ********** Operations on relation newpmr
% *****


% ***** Internal routine to insert a new part message to
% ***** MRP II users
% *****

insert(newpmr(Pnum))

% *** Stop if record already exists
%
-->   newpmr(Pnum).

% *** Insert record
%
-->   ~(newpmr(Pnum)),
      add(newpmr(Pnum)).



% ***** Internal routine to delete new part message from MRP II
% *****

delete(newpmr(Pnum))

% *** Stop if record does not exist
%
-->   ~(newpmr(Pnum)).

% *** Delete record
%
-->   newpmr(Pnum),
      remove(newpmr(Pnum)).
```

```
% ********** Operations on relation newrev
% *****


% ***** Internal routine to insert a new revision message to
% ***** MRP II
% *****

insert(newrev(Pnum,Rev))

% *** Stop if record already exists
%
-->   newrev(Pnum,Rev).

% *** Insert record
%
-->   ~(newrev(Pnum,Rev)),
      add(newrev(Pnum,Rev)).


% ***** Internal routine to delete new revision message
% ***** from MRP II
% *****

delete(newrev(Pnum,Rev))

% *** Stop if record does not exist
%
-->   ~(newrev(Pnum,Rev)).

% *** Delete record
%
-->   newrev(Pnum,Rev),
      remove(newrev(Pnum,Rev)).
```

```
% ********** Operations on relation rereleased
% *****


% ***** Internal routine to insert rereleased message to
% ***** MRP II
% *****

insert(rereleased(Pnum,Rev))

% *** Stop if record already exists
%
-->   rereleased(Pnum,Rev).

% *** Insert record
%
-->   ~(rereleased(Pnum,Rev)),
      add(rereleased(Pnum,Rev)).



% ***** Internal routine to delete a rereleased message
% ***** from MRP II
% *****

delete(rereleased(Pnum,Rev))

% *** Stop if record does not exist
%
-->   ~(rereleased(Pnum,Rev)).

% *** Delete record
%
-->   rereleased(Pnum,Rev),
      remove(rereleased(Pnum,Rev)).
```

```
% ********** Operations on relation obsolete
% *****


% ***** Internal routine to insert an obsolete message into
% ***** MRP II
% *****

insert(obsolete(Pnum,Rev))

% *** Stop if record already exists
%
-->   obsolete(Pnum,Rev).

% *** Insert record
%
-->   ~(obsolete(Pnum,Rev)),
      add(obsolete(Pnum,Rev)).



% ***** Internal routine to delete obsolete message
% ***** from MRP II
% *****

delete(obsolete(Pnum,Rev))

% *** Stop if record does not exist
%
-->   ~(obsolete(Pnum,Rev)).

% *** Delete record
%
-->   obsolete(Pnum,Rev),
      remove(obsolete(Pnum,Rev)).
```

```
% ********** Operations on relation inventory
% *****


% ***** Routine to insert an inventory record into MRP II
% *****

insert(inventory)

-->    insert(inventory(Pnum,Qty)).


insert(inventory(Pnum,Qty))

% *** Stop if record already exists
%
-->    nonvar(Pnum) /\ nonvar(Qty)
       /\ inventory(Pnum,Qty).

% *** Request part number if not provided
%
-->    var(Pnum),
       write('Part number? '),
       read(Pnum),
       insert(inventory(Pnum,Qty)).

% *** Request total inventory quantity if not provided
%
-->    var(Qty)
       /\ nonvar(Pnum),
       write('Total quantity in inventory? '),
       read(Qty),
       insert(inventory(Pnum,Qty)).

% *** update inventory record to reflect new quantity
%
-->    nonvar(Pnum) /\ nonvar(Qty)
       /\ ~inventory(Pnum,Qty),
       delete(inventory(Pnum,Qold)),
       add(inventory(Pnum,Qty)),
       write('Inventory record has been updated'),
       nl.
```

```
% ***** Routine to delete inventory records from MRP II
% *****

delete(inventory)

-->   delete(inventory(Pnum,Qty)).


delete(inventory(Pnum,Qty))

% *** Stop if record does not exist
%
-->   nonvar(Pnum)
      /\ ~inventory(Pnum,Qty).

% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part number? '),
      read(Pnum),
      delete(inventory(Pnum,Qty)).

% *** Delete record
%
-->   nonvar(Pnum)
      /\ inventory(Pnum,_),
      remove(inventory(Pnum,_)).
```

```
% ********** Operations on relation onorder
% *****


% ***** Routine to insert an on-order record in MRP II
% *****

insert(onorder)

-->    insert(onorder(Pnum,Qty,Ordno)).


insert(onorder(Pnum,Qty,Ordno))

% *** Stop if record already exists
%
-->    nonvar(Pnum) /\ nonvar(Qty) /\ nonvar(Ordno)
       /\ onorder(Pnum,Qty,Ordno).

% *** Request part number if not provided
%
-->    var(Pnum),
       write('Part Number? '),
       read(Pnum),
       insert(onorder(Pnum,Qty,Ordno)).

% *** Request quantity on order if not provided
%
-->    var(Qty)
       /\ nonvar(Pnum),
       write('Quantity? '),
       read(Qty),
       insert(onorder(Pnum,Qty,Ordno)).

% *** Request order number if not provided
%
-->    var(Ordno)
       /\ nonvar(Pnum) /\ nonvar(Qty),
       write('Order number? '),
       read(Ordno),
       insert(onorder(Pnum,Qty,Ordno)).

% *** Insert record
%
-->    nonvar(Pnum) /\ nonvar(Qty) /\ nonvar(Ordno)
```

```
      /\ ~onorder(Pnum,Qty,Ordno),
      add(onorder(Pnum,Qty,Ordno)),
      write('On-order record has been added'),
      nl.


% ***** Routine to delete an on-order record from MRP II
% *****

delete(onorder)

-->   delete(onorder(Pnum,Qty,Ordno)).


delete(onorder(Pnum,Qty,Ordno))

% *** Stop if record does not exist
%
-->   nonvar(Pnum) /\ nonvar(Ordno)
      /\ ~onorder(Pnum,_,Ordno).

% *** Request part number if not provided
%
-->   var(Pnum),
      write('Part number? '),
      read(Pnum),
      delete(onorder(Pnum,Qty,Ordno)).

% *** Request order number if not provided
%
-->   var(Ordno)
      /\ nonvar(Pnum),
      write('Order number? '),
      read(Ordno),
      delete(onorder(Pnum,Qty,Ordno)).

% *** Delete record
%
-->   nonvar(Pnum) /\ nonvar(Ordno)
      /\ onorder(Pnum,_,Ordno),
      remove(onorder(Pnum,_,Ordno)),
      write('On order record deleted'),
      nl.
```

```
% ********** Other MRP II operations without specific
% ********** relations
% *****


% ***** Internal Routine to ask about engineering changes
% ***** for assembly level changes in MRP II
% *****

checkecnmrp(Newpart,Newrev,Pnum,Rev,Ppn,Prn)

% *** Ask user if change necessitates a new assembly
% *** part number
%
-->   var(Newpart),
      write('Does this change require a new assembly
             part number? '),
      read(Newpart),
      checkecnmrp(Newpart,Newrev,Pnum,Rev,Ppn,Prn).

% *** Ask user if change necessitates a new assembly revision
% *** level (user indicated a new part number was not necessary)
%
-->   var(Newrev)
      /\ nonvar(Newpart) /\ ~(Newpart = 'yes'),
      write('Does this change require a new assembly revision
             level? '),
      read(Newrev),
      checkecnmrp(Newpart,Newrev,Pnum,Rev,Ppn,Prn).

% *** User indicated a new assembly part number is needed, so
% *** a new revision level is not appropriate
%
-->   var(Newrev)
      /\ nonvar(Newpart) /\ (Newpart = 'yes'),
      checkecnmrp(Newpart,no,Pnum,Rev,Ppn,Prn).

% *** User indicated that neither a new assembly part number
% *** nor revision level is needed
%
-->   nonvar(Newpart) /\ nonvar(Newrev)
      /\ ~(Newpart = 'yes') /\ ~(Newrev = 'yes'),
      Ppn=Pnum,
      Prn=Rev.
```

```
% *** Let the user insert a new assembly part
%
-->   nonvar(Newpart) /\ nonvar(Newrev)
      /\ (Newpart = 'yes') /\ ~(Newrev = 'yes'),
      insert(mrppmr(Ppn,Dnum,Dsize,Des,Buom,Poum,Cfuom,Scode,
      Cost,Lt,Pnum,Spnum)),
      mrprev(Ppn,Prn,_,_,_),
      copybommrp(Pnum,Rev,Ppn,Prn).

% *** Let the user insert a new assembly revision
%
-->   nonvar(Newpart) /\ nonvar(Newrev)
      /\ (Newrev = 'yes')
      /\ ~(Newpart = 'yes'),
      Ppn=Pnum,
      insert(mrprev(Ppn,Prn,Estart,Eend,Mstat)).



% ***** Internal routine to print out current assembly part
% ***** number and revision level for engineering change
% ***** consideration  (Used by both CAD and MRP II)
% *****

ecnheader(Pnum,Rev)

% *** Stop if part number and revision are not provided
%
-->   var(Pnum) /\ var(Rev).

% *** Write assembly number if only it is provided
%
-->   nonvar(Pnum) /\ var(Rev),
      nl, write('Current assembly part number: '), write(Pnum),
      nl.

% *** Write revision level if only it is provided
%
-->   var(Pnum) /\ nonvar(Rev),
      nl, write('Current assembly revision level: '), write(Rev),
      nl.

% *** Write assembly number and revision level if both are
% *** provided
%
```

```
-->   nonvar(Pnum) /\ nonvar(Rev),
      nl, write('Current assembly part number: '), write(Pnum),
      nl,
      write('Current assembly revision level: '), write(Rev),
      nl.
```