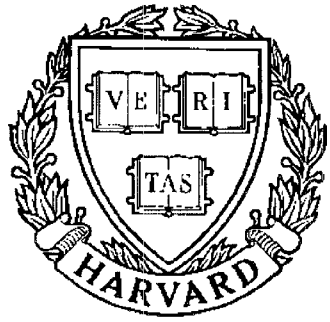
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



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

Formal Representation, Verification and Implementation of Rule Based Information Systems for Integrated Manufacturing (INSIM)

by G. Harhalakis, C.P. Lin, L. Mark, P.R. Muro

TR 91-19

Formal Representation, Verification and Implementation of Rule Based Information Systems for Integrated Manufacturing (INSIM)

G. Harhalakis, C.P. Lin, L. Mark, P.R. Muro

Systems Research Center
University of Maryland

February 20, 1991

Abstract

Full control and management of information flow has not yet been achieved, mainly because of the data inconsistencies and lack of established functional relationships among different manufacturing application systems. Research toward CIM has been concentrating on the computerization of individual functions of manufacturing , such as computer aided design and shop floor control, and the integration of data relations, such as global database frameworks and distributed database management systems. A mechanism to control the information flow among all of the manufacturing manufacturing application systems, in order to streamline factory activities based on company-specific and company-wide policies and procedures is proposed here. The goal is to achieve a fully integrated manufacturing management system. The INformation Systems for Integrated Manufacturing (INSIM) reflects a design methodology to build a knowledge base to serve as the control mechanism. The methodology includes knowledge acquisition, graphical modeling, systematic validation and automated implementation. This design methodology features an enhnaced graphic modeling tool - Updated Petri Nets (UPN) - which is capable of modeling database updates and retrievals, under specific constraints and conditions and uses a hierarchical modeling approach. Finally, a software package based on the INSIM methodology was developed and a prototype rule based system in Update Dependencies language - a special rule specification language - is being implemented. It assimilates the functionality of information flow between Computer Aided Design, Process Planning, Manufacturing Resource Planning and Shop Floor Control.

Contents

List of Figures

1 Introduction

1.1 Computer Integrated Manufacturing Systems

Due to the substantial improvements in computer technologies and increasing competition in the manufacturing industry, more computerized manufacturing applications have been developed to automate various manufacturing functions in most of the existing factories, and to design new factories. Factory automation started with the introduction of NC machine tools in the early 50s, and continued up to the recent establishment of flexible manufacturing systems and the overall control of the actual flow of products and materials on a factory floor. Current research and computer software developed in the area of manufacturing automation has been quite intensive in dealing with product and process design, production planning, and job execution. However, the design of such systems was made in a functional fashion that emphasized "local" solutions, using closed and self-contained architectures. This, together with the use of heterogeneous databases and incompatible computer operating systems have led to "islands of automation" (figure 1) which suffer from data inconsistencies and lack of control of functional interactions between manufacturing application systems. Current and future trends for the use of computers in manufacturing include the control and the integration of information flow of a production operation into a computer-controlled factory management system. The concept of a factory level Computer Integrated Manufacturing (CIM) system is to fill the gap between the high level production management and the low level factory automation.

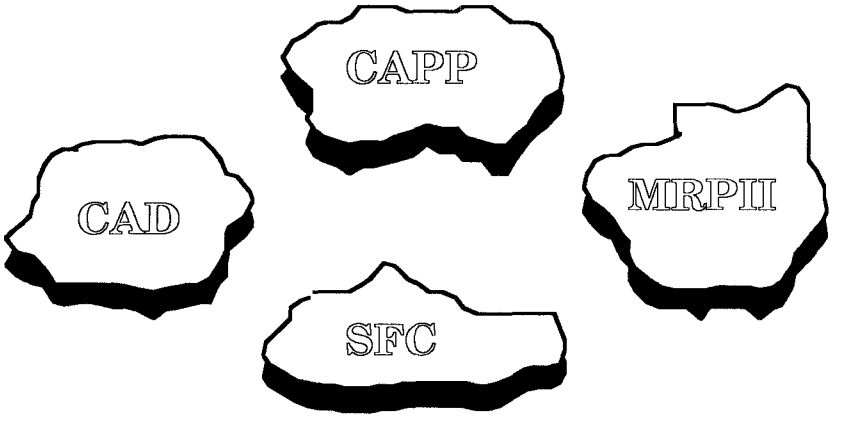


Figure 1: Islands of Automation

The primary objective of our research is to develop a methodology of acquiring domain knowledge (company policy), to verify and implement it as a Knowledge Based System, in order to control the information flow among all those computer based manufacturing

application systems. Information flow control is unique in our resaerch, other research whose primary objective is to develop a consistent database framework or a standard communication protocol for data transformation(figure 2). Our control mechanism, accompanied with existing distributed database management systems, can achieve a fully integrated manufacturing information system. Our view of the control in information system is:

Data ownership : Each application system has authority to create, update and delete specific data entities, in reflecting to specific company policies.

Precedence constraints : The pre-conditions of and sequences among activities within all the application systems involved has to be defined according to specific company policies.

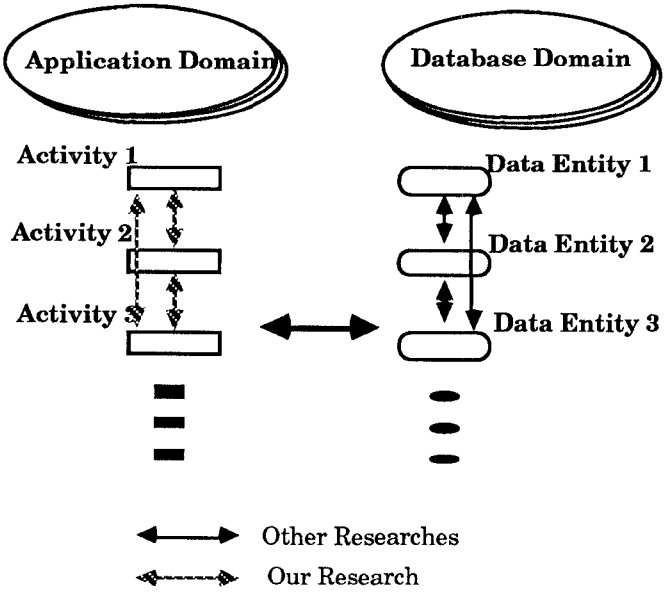


Figure 2: Difference between the objective of our research and other research

The main characteristics of the manufacturing environment dealt with here are described below.

Islands of automation : Different computer based manufacturing application systems have been developed and used independently, without communicating with each

other. The information flow between them is carried out through unreliable paper work, and it is controlled by various company departments without a unified set of procedural rules [ANDE 84].

Emergence of new applications : New computer based manufacturing application systems are being introduced and these will also need to be integrated into the existing integrated system [APPL 84]. Thus, the need for modularity.

Distributed systems : Most factories are already using distributed computer systems, with multiple databases, that serve specific applications in the factory. This has become the current trend in computer technology [JABL 88]. Thus, the need for communications.

Integrity and security : Data integrity and security in individual databases have become major issues of concern in information integration [DATE 86]. Thus, the need for controlled asscessibility.

Our research, aiming at linking product and process design, manufacturing operations and production management, focuses on the control of information floow between each of the key manufacturing applications at the factory level, including Computer Aided Design (CAD), Computer Aided Process Planning (CAPP), Manufacturing Resource Planning (MRP II), and Shop Floor Control (SFC) systems. This linkage between manufacturing application systems involves both the static semantic knowledge of data commonalities and the dynamic control of functional relationships. The common data entities, which form the basis of the integrated system, can be classified in two categories in a manufacturing environment: Static and Dynamic. The former define the various entities of the distributed system, while the latter deal with the functioning of the system as it operates to satisfy the market demand.

1. Static
    - Product Data : part master data, part revison records, and bills of material.
    - Process Data : routings or process plans.
    - Resource Data : work centers and labor resources.
2. Dynamic
    - Planning Data : purchase orders and manufacturing orders.

The functional relationships, which form a shell on top of the integrated system to control the sequences of functions within and among those manufacturing application systems, and the database updates and retrievals, are purely domain dependent and can

be very different depending on specific company policies. The methodology, however, for knowledge aquisition, modeling, validation and implementation is totally generic.

The three major research issues to be addressed are:

Design of the CIM architecture The CIM architecture is concerned with defining the functionalities of each manufacturing application system and the relationships between them.

Development of a methodology for the design of the KBS
The design and maintenance of a Knowledge Based System(KBS) to control the functional relationships and information flow within the elements of the integrated system is the major task of this research. Based on a graphical modeling tool, Petri Nets, a hierarchical graphic representations will be developed to model the KBS, to verify the properties of it, and to validate the company policy.

Implementation A procedure to automatically translate the KBS representation tool into a specific rule specification language needs to be developed, and a prototye system to be created and tested.

1.2 Multi-Database versus Central Database

The basic approach of information integration used in our research is to build a knowledge based system to control an integrated information flow between multiple databases of all existing manufacturing applications, and to cater for the addition of new applications as they emerge. Alternative approaches include:

- Design and installation of a complete new, fully integrated system [LU 86]
- Design of a single centralized data base scheme to be accessed by all exsiting application systems [HSU 87].

The first approach involves substantial investment in replacing computer hardware and software, and retraining of all related personnel. The second approach requires that all the data of the application systems being integrated to be converted into a common database schema and all the application systems to be rewritten, which will require substantial effort and will still limit the incorporation of new computer based manufacturing applications. These two alternative approaches may apply to small factories with less amount of data to be processed and less sophisticated systems to operate. Our approach, on the other hand, uses the existing application systems and tries to reduce the amount of modification to them. It is also capable of incorporating new applications in a modular fashion without the need to design the entire system.

The next section reviews current research related to the development of Computer Integrated Manufacturing systems and Petri Net applications. The third section presents our CIM system specification and its architecture, and the fourth section details the design methodology of the knowledge base. The fifth section describes the Updated Petri Nets (UPN) and the sixth section presents top-down refinement and Petri Net synthesis techniques. The seventh section details the mathematical formalism of the model representation and the algorithm for structuring the knowledge base. The eighth section discusses some specific properties of the knowledge base and how inconsistencies and modeling errors can be addressed through analytical methods. The nineth section presents translation of the knowledge base into a specifically designed rule specification language and the overall implementation strategy. The last section presents our conclusions with recommendations for future work.

2 Literature Survey on CIM and Applications of Petri Nets

2.1 CIM System Architecture and SFC

The National Institute of Standards and Technology (NIST), within its Automated Manufacturing Research Facility (AMRF), has addressed issues related to standardized interfaces for data-sharing, which are important for system integration [JONE 86]. The NIST has developed a manufacturing system architecture, shown in figure 3, which is based on a five-level hierarchical structure: Facility, Shop, Cell, Workstation, and Equipment levels of control. Current efforts focus on the design of a real-time production scheduler at the Shop level and the Cell level [DAVI 88] [JONE 89], but not yet on the design of a complete factory integrated system.

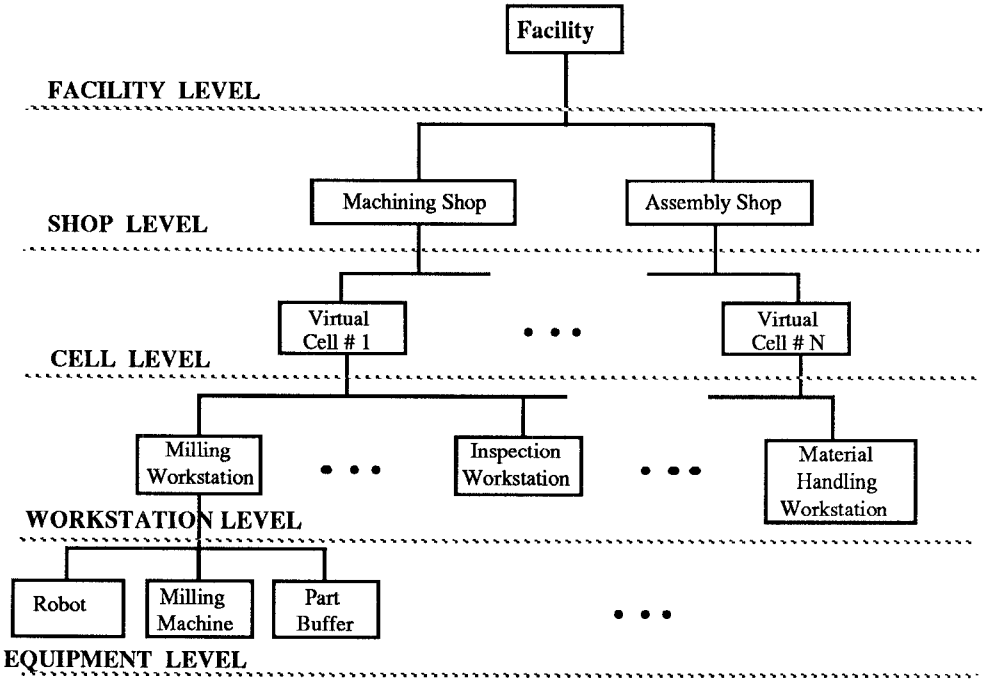


Figure 3: NIST CIM System Architecture.

Project #418 [BONN 87] of the European Strategic Program for Research and development in Information Technology (ESPRIT) has resulted in a hierarchical structure with multi-level dynamic planning and scheduling and a GRAI modeling method for decision making. ESPRIT project #932 [MEYE 87] provides an information flow diagram for factory analysis, derived from the hierarchical NIST model, using the SADT modeling method being reviewed later on in this section. The research of ESPRIT focuses on

the design of a control architecture at Shop and Workcell levels and some results have been already achieved. However, there is no provision for the control of information flow between the factory level and the job shop level.

The Computer Aided Manufacturing - International (CAM-i) has established an Intelligent Manufacturing Management Program (IMMP) to develop the concepts for an advanced factory management and control system. The objectives are to ensure efficient use of resources and to facilitate decision-making at the lowest level and at the last possible moment. So far, such a decision making system has been developed and implemented, known as MADEMA (MAnufacturing DEcision MAking) at MIT [CHRY 87]. However, this system does not address control and integration of information flow between system modules. Rather, it concentrates on decision making rules for dispatching of work orders to work centers.

PRISM (Productivity Improvement Systems for Manufacturing) [FRAN 87], developed in AT&T Bell Laboratories, presents an integrated architecture of various computerized information systems, supporting manufacturing execution functions, and a useful tool for controlling information flow at AT&T's factories. It provides a logical architecture for the Shop Floor Control system. It concentrates on the scheduling and control of physical Shop Floor activities through its MOVES system, and the interchange of product and process data between its Shop Floor Control system and the other manufacturing application systems (eg. its own product and process design system "FOCUS"). Although this system provides the communication between various manufacturing application systems including CAD, CAPP, and SFC, it does not intend to incorporate the control of information flow based on specific company rules. However, we have adopted their SFC definition and functionality in defining the relationship between SFC and each of the CAD, CAPP, MRP II applications. A detailed description of the functionality of a generic SFC system will be described in section 3.1.2.

2.2 Database Development for CIM

The Integrated Manufacturing Database Administration System (IMDAS) developed by the NIST [SU 86] emphasizes on extending the traditional database management technology, to synchronize the data processing through each module of the CIM system. The application programs communicate with IMDAS through a common interface, using an SQL-like language, referencing data names from a common data dictionary. Depending on the transactions invoked by the application programs, IMDAS retrieves or updates data through a common interface to the underlying distributed databases. IMDAS is one of the earliest developments of database management systems interfacing with distributed manufacturing application systems. The IMDAS has influenced the interfacing of our Knowledge Based System with the database management system of each existing manufacturing application.

A manufacturing information management design method was developed at Rensselaer Polytechnic Institute,[HSU 87] which uses a two-stage entity relationship (TSER) modeling method and a knowledge-based control methodology in a metadatabase framework. This work provided the design concept and the necessary methodologies to establish a metadatabase for the integration of CAD and MRP systems. However, they have concentrated rather in the development of a database framework for data integrity and consistency, which is also part of our research goal, but not in the development of a knowledge base for controlling the data updates and retrievals based on a given company policy.

The CIM Data Engine (CDE) developed at TRW [SEPE 87], similar to the concept of IMDAS, translates the queries from the application systems and interfaces with the databases where the required data resides. The basic function of the CDE is that of a data translator and synchronizing processor for the CIM network. It is trying not to change the existing application systems, and the only change is an addition of interface software that transmits all transactions, and requests information from the other applications, to the CIM engine. Their concept of information integration is very similar to ours , except that, like most of the other CIM systems, it does not address the issue of defining precedence constraints, which reflect the specific company policy, to control the information flow between different manufacturing application systems.

2.3 Knowledge Based Expert Systems For CIM

A knowledge-based expert system can be the heart of CIM, since it can accommodate a lot of rules and constraints based on manufacturing expertise. Building high-performance knowledge-based expert systems for advanced manufacturing and automation is now the most active research subject within CIM and AI. The Knowledge-Based Engineering Systems Research Laboratory at University of Illinois-Urbana-Champaign, has developed a framework for knowledge-based computer-integrated manufacturing [LU 86] which can be used to perform common manufacturing tasks, such as monitoring, diagnostics, control, simulation, and scheduling. Basically, this framework provides a way for manufacturing applications to access data and knowledge residing in the other applications, rather than to define the precedence constraints of the transactions performed for the control of the information flow. The MKS (Manufacturing Knowledge System) at Stanford University [PAN 89] provides a set of tools for modeling a manufacturing environment (including process, equipment, facilities, and operational procedures). However, their approach emphasizes in the provision of a single consistent representation of the manufacturing domain that is shared by all applications, in the form of a common database schema. This requires major changes to merge new functions within the existing application systems, as well as major redesigning when incorporating new application systems.

2.4 System Modeling Tools

Two well known graphic modeling methods are being reviewed: SADT [ROSS 85] (Systematic Analysis and Design Technique) and Petri Nets [PETE 81]. SADT is an extended modeling method spun from IDEF (ICAM Definition) developed by the United States Air Force in their Integrated Computer Aided Manufacturing (ICAM) program [STAF 83]. It can be used to define the information flow between all the basic subsystems within a factory, and is very useful in terms of modeling static data functions and relationships of manufacturing systems through a hierarchical structure and the information flow between each of the functions. However, it can not model the precedence constraints or the sequences of events occurring in a manufacturing environment, because of its limited modeling structure which has basically two entities: boxes, representing the functions or operations, and arrows, representing the information exchanges between operations. With only these two entities, it can not represent both the precedence constraints of starting or ending an operation and the concurrency of operations. Strictly speaking, SADT can only be used to define the functionalities of each application systems and the inputs/outputs between them.

On the other hand, Petri Nets are ideal for modeling dynamically and formally analyzing complex dynamic relationships of interacting systems. They were initially developed and used mainly for advanced computer integrated system design, both in hardware and software, such as artificial intelligence in network systems [COUR 83], and for flexible manufacturing systems [CROC 86]. Most recent applications of Petri Nets in manufacturing systems are focusing again on the shop floor level, with a large number of work stations, robots, and transportation systems, to be handled by a central controller. Timed Petri Nets are primarily used for scheduling and sequencing [RAVI 87] [MERA 87] [DRID 85].

2.5 High Level Petri Nets

Research in modeling manufacturing systems has been quite extensive in recent years on system validation and performance evaluation using High Level Petri Nets (HLPN), such as Timed Petri Nets, Stochastic Timed Petri Nets, Predicate Transition Nets, and Colored Petri Nets [ALAN 84] [ALLA 84] [KAMA 86] [CROC 87]. In modeling information systems for the control of information within and between manufacturing applications, there has been some work done already, [HARH 88] [HARH 90] in relation of CAD and CAPP, with MRP II systems. However, the previous work has been done by using Generalized Petri Nets (GPN) and problems were encountered when the complexity of the system increased. The number of places and transitions in the model became very large and the representation was unwieldy. Also, if the systems under consideration have several identical processes, it will be necessary to have several identical subnets in the

corresponding model. To resolve these problems, we chose to use the Colored Petri Nets (CPN) which will allow the model designer to work at different aggregation levels. The main advantage of CPNs over GPN is the possibility of getting a compact representation of a large and complex system.

2.6 Conclusion

As a result of this review, it is evident that, these efforts, which mainly concentrate on either the operation integration issues at the shop floor level or the data integrity and consistency issues at higher levels, are quite different from our CIM system, which controls the information flow at a higher level-the facility level, based on knowledge expressed in the form of specific company policies. It is, therefore, our belief that our work indeed complements existing CIM architectures, using a hierarchical approach. Also in contrast to the single database CIM systems, we propose a database interoperability approach, that has already been used successfully in developing a knowledge-based prototype CAD/CAPP/MRP II integrated system [MARK 87] [HARH 88] [HARH 90], at the Department of Mechanical Engineering and the Systems Research Center of the University of Maryland. It is aiming at being independent of the diversities of different hardware and software, and provides the possibility of adding to existing computer facilities in a modular way rather than replacing them.

3 CIM System Specification and Architecture

3.1 System Specification

3.1.1 Integrated System Description

The integrated system is intended for a discrete-parts, make-to-stock environment, where MRP II, CAD, CAPP, and SFC systems are best utilized. As described previously, the design of the model is based on the information flow established between the four functional areas. The common entities involved in this integrated model are Part Master Data (including Part revisions), Bills of Materials, Workcenters, Process Plans, and Manufacturing Orders as shown in figure 4 .

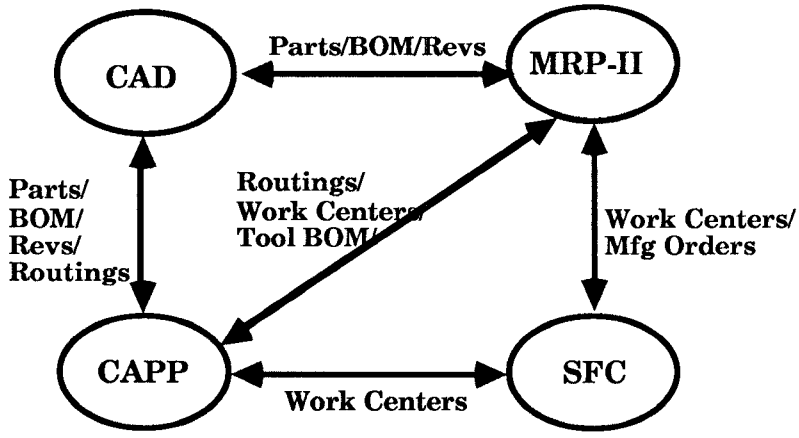


Figure 4: Data Commonalities between CAD, CAPP, MRP II, and SFC

The model does not attempt to provide a 'bridge box' so to say, allowing users to hook up any existing commercial MRP II, CAD, CAPP, and SFC systems. The goal is to demonstrate the viability of achieving the integration and control of information flow, using generic operations on generic entities. In order to remain as general as possible, this model does not emulate any particular MRP II, CAD, CAPP, or SFC packages. It relies only on the basic functions available to most such commercial systems. In an actual implementation, the system would simply act as a controller between existing CAD, CAPP, MRP II, and SFC packages, utilizing their respective capabilities. It must be understood that the appropriate model has to be developed, when attempting to control and integrate specific MRP II, CAD, CAPP, and SFC software, depending upon their specific features and characteristics, and when a specific company policy is given. Therefore, the value of our work lies mostly in developing a methodology for designing a CIM system, rather than the CIM system itself.

3.1.2 Role of each application system

To maintain the genericity of the model, only the most basic data carried by CAD, CAPP, MRP II, and SFC have been incorporated. If necessary, the model can be easily extended to reflect additional data and functions specific to particular commercial packages.

Specifying the roles of the respective areas, ie; CAD, CAPP, MRP II, and SFC is of most importance when designing the model. CAD, being the center of design activity, is the primary controller of product design information. The evaluation of design alternatives, creation of new product parts, and the modification of existing parts is performed within CAD, though using inputs from other departments. Marketing and Manufacturing are two major contributors to information regarding product designs. In addition, CAD initiates the bills of materials for all product assemblies. An important problem commonly encountered is that, as a function, manufacturing succeeds design. Therefore any manufacturing problems occurring due to part design specification, are relayed to CAD only after designs have been finalized. It is therefore necessary for CAPP, the originator of process plans in the system, to work in concert with CAD, as the design of a part is ongoing. This approach known as concurrent engineering reduces the product development cycle and enhances competitiveness.

CAPP is solely responsible for developing manufacturing process plans as explained earlier. It organizes the manufacturing activities to be performed on a part, into specific operations, each being assigned to a particular workcenter, and each requiring tools, jigs, fixtures and set up and run times. CAPP on the other hand can initiate its own parts and bills of materials as they relate to necessary tools, jigs, and fixtures for production purposes. In addition, most CAPP systems maintain detailed workcenter files, the information being used while preparing process plans.

MRP II plays a coordinating and monitoring role. It plans for and monitors the actual procurement of raw materials and manufacture of parts, respectively. It can also initiate non-product parts such as tools and supply items, into the system. In addition, it records process plans as generated by CAPP, and also product structures of assembly parts, to provide them later to the SFC module. Workcenter data are maintained here, with MRP II having sole discretion as to their initiation and deletion in the system.

While the definitions and functionalities of CAD, CAPP, and MRP II systems are quite clear and widely accepted, the functions and inputs/outputs of a Shop Floor Control system in a CIM environment are not yet well defined. Shop Floor Control is basically a system which directly controls the transformation of planned manufacturing orders into a set of jobs, and the transformation of raw materials into products. The basic activities of a Shop Floor Control module are listed below [MELN 85]:

- Capacity planning and resource allocation based on inputs from MRP II.
- Short-term capacity adjusting by using alternative routings, planning overtime, etc.

- Feedback for reporting machine performance and status, job completion stage, and actual labor and material usage.

Our definition of a generic Shop Floor Control system and identification of the input/output requirements, has been influenced by a study of Shop Floor Control systems from three major research projects (NIST, ESPRIT, MADEMA) in the literature. The Productivity Improvement Systems for Manufacturing (PRISM) developed in AT&T Bell Laboratories has been our primary guideline. The basic functions and inputs/outputs of a generic Shop Floor Control system are shown in figure 5 .

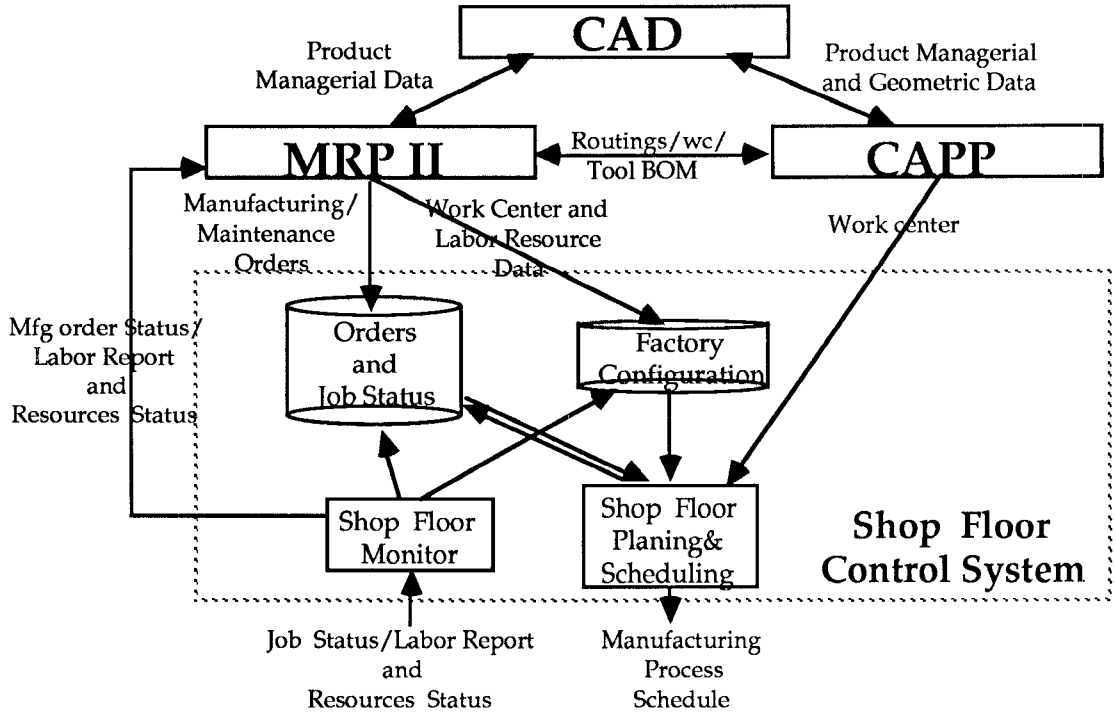


Figure 5: Definition of a generic shop floor control system.

SFC is designed to communicate with an MRP II Planning system and to perform job scheduling and monitoring using detail routing information from CAPP. Once a market demand arrives, MRP II will generate planned purchasing and manufactruing orders. In doing this, lead times from Part Master Records and existing inventories are taken into account. SFC will then schedule the manufacturing jobs, based on the current load and the detail routing information from CAPP. It will then dispatch these job orders down to the shop floor. During production, SFC will constantly monitor the job status, work

center status, actual material and labor consumption, and report them back to the MRP II for costing and updating purposes. Finally SFC is supposed to react to and provide realtime solutions in the event of disturbances, such as machine breakdowns, critical labor absenteeism and material shortages. Issues that can not be resolved at the SFC level must be communicated to MRP II for further action, [NAGI 88].

3.2 Overall CIM Information Flow Architecture

Our CIM architecture concentrates on the integration of manufacturing applications at the Factory level, referring to the Facility level of the NIST hierarchical architecture, as depicted in figure 6. CAD, CAPP, MRP II, and SFC can be integrated together through a general Distributed Database Management System (DDBMS). The Knowledge Based System acts as a shell on top of all the application systems through the DDBMS to control the information flow with the consideration of functional relationships derived from a company policy. In order to build a prototype of the CAD/CAPP/MRP II/SFC integrated System, we have defined data structures of the common data entities involved in the various manufacturing applications of our integrated system and their relations, which will be stored in a DDBMS.

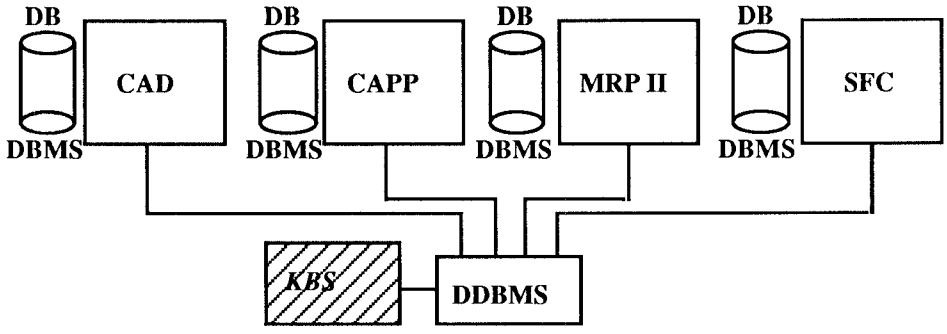
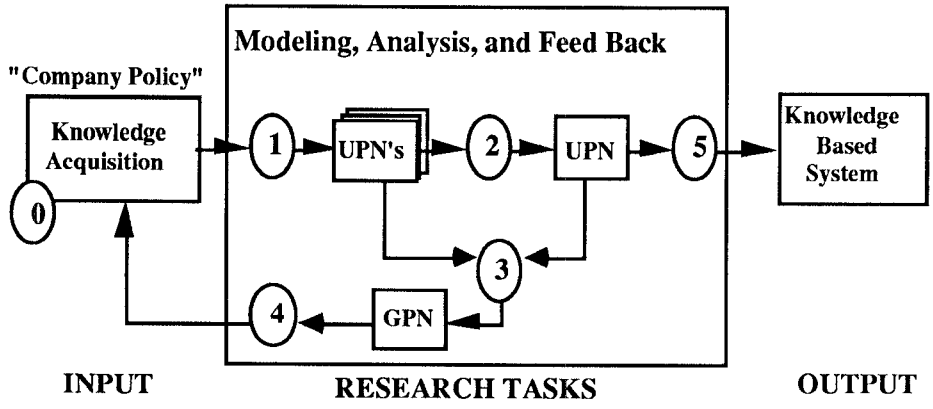


Figure 6: Overall CIM Information Flow Architecture

4 Knowledge Base Design Methodology

As mentioned above, the design and maintenance of a Knowledge Based System(KBS) to control the functional relationships and information flow within the integrated system is a major task of this research. Our design methodology for the Knowledge based system is depicted in figure 7. It starts from user defined rule specifications , reflecting specific company policy, which will then be modeled using a special set of Colored Petri Nets -



0. Expression of company policy for the integration of specific application systems (CAD/CAPP/MRP II/SFC)

1. Modeling of the knowledge base using a formal language, Updated Petri Nets (UPN), a sub set of Colored Petri Nets.

2. Synthesis Rules to combine modeled scenarios of the company policy into an integrated system.

3. Transform the UPN into Generalized Petri Nets (GPN) for Knowledge Base Verification.

4. Analysis, discovery of inconsistencies and incompleteness, and feedback.

5. Translation from UPN to the Knowledge Based System.

Figure 7: Knowledge Base Design Methodology

UPN(Updated Petri Net) and a hierarchical modeling methodology, discussed respectively in sections 5 and 6. The next step is to convert UPN model into GPN for analysis purposes, and feed the results back to the users to resolve (i) conflicting business rules and (ii) errors introduced during the modeling phase. After the model has been validated, a parser will then translate the UPN model into the Knowledge Base system in the form of some kind of a rule specification language. The knowledge-based system will be a software package that will control the data-flow and access between several data bases. In short, the input will be a set of business rules and the output will be an AI program for controlling functions, access and updates of data within the manufacturing applications involved. An overview of the major phases of this project is presented below.

4.1 Development of the expert rules (Company Policy)

The design of the model is based on the information flow established between all the manufacturing applications, namely CAD/CAPP/MRP II and SFC. The expert rules embedded in the knowledge base are extracted from company expertise, which can be given through a number of individual interviews and group meetings with experts, from all manufacturing application systems to be integrated, and managers responsible for making company policy. Therefore, substantial effort may be required for gathering all expert rules to form the knowledge based system. However, since we are here to develop and demonstrate our design methodology, our prototype will only include limited rules extracted from our own industrial experience and other industries involved with this and other projects in the CIM Laboratory.

4.1.1 Scenarios of the Proposed CIM System

The development of a knowledge based system usually starts from designing a set of abstract rules for the entities within the system. A set of scenarios which represent these abstract rules of the proposed CIM system is listed below.

Part Data

- Adding New Product Parts in CAD
- Adding New Product Part Revisions of Parts in CAD
- Adding New Non-Product Parts in CAPP
- Adding New Non-Product Part Revisions of Parts in CAPP
- Adding New Non-Product Parts in MRP II
- Adding New Non-Product Revisions in MRP II
- Making Parts Obsolete
- Deleting Parts

Product Structures (Bills of Material)

- Adding Component Relationships in CAD
- Adding Component Relationships in CAPP
- Deleting Component Relationships

- Substituting Components in Relationships
- Changing the Required Quantity of a Component
- Copying Relationships from One Assembly to Another

Process Plans

- Establishing New Process Plans in CAPP
- Modifying Process Plans in CAPP
- Deleting Process Plans in CAPP

Work Centers

- Establishing New Work Centers in MRP II
- Modifying Work Center in MRP II, CAPP, and SFC
- Deleting Work Centers in MRP II

Manufacturing Orders

- Adding Orders in MRP II
- Modifying Orders in MRP II
- Modifying Order BOM/Routing in SFC
- Updating Job Status in SFC (include actual material issued and time taken)
- Deleting Orders from MRP II

4.1.2 Status Codes

The flow of information within the system is controlled using a set of status codes assigned to each set of data within each functional area. The status codes are designed to provide for the efficient transfer of information between the individual systems while controlling the sequence of various part and process design and manufacturing related activities. The following are the status codes used in the system.

Working : The "working" status is given to CAD part data related to designs that have not yet been finalized. In a similar fashion it is given to process plans in CAPP, workcenters in CAPP, and shop orders in SFC. In the case of workcenters it is intended to signify that the workcenter data has not been completely input in CAPP.

Released : The "released" status is indicative of an entity becoming active in the system. It is applied to CAD and MRP II part revision data; CAPP, SFC, and MRP II workcenter data; CAPP and MRP II process plan data; SFC and MRP II order data .

Hold : The "hold" status is normally given to an entity, when it is being reviewed for possible revision or replacement, and the entity should not be used while on hold. In addition, it can be given to a workcenter in the case of a extended breakdown. It is used for CAD and MRP II part revision data; CAPP, SFC, and MRP II workcenter data; CAPP and MRP II process plan data; SFC and MRP II order data.

Obsolete : Data related to entities that are no longer considered active, are given the "obsolete" status. This code is used by CAD part revisions and CAPP routings. However, MRP II and SFC handle obsolescence with the use of effectivity start and effectivity end dates. Therefore they do not require this status code.

4.2 Modeling of the Knowledge Base

Although the General Petri Nets initially adopted in this research, can handle the modeling of the knowledge base, it has become necessary to define complex semantics for the nets in order to handle the complexity of the knowledge base, due to the involvement of more applications and their entities, and the growing size of it. Hence we have started developing the UPN, and a hierarchical modeling methodology with a systematic approach for the synthesis of separate nets with or without common places [JENG 90]. UPN ae defined and presented in sections 5 and 6

The hierarchical modeling methodology facilitates the modeling task. It incorporates: (a) a top-down stepwise refinement technique for the modeling of each scenario from an abstract and aggregate level to a detailed level and (b) a synthesis technique for synthesizeing separate nets, which represent different scenarios of the system, to form a coherent net. The approach necessitates the development of new Petri Nets modeling entities which include two types of transitions; one to represent primitive operations, and the other to represent compound operations which can be further exploded into sub-nets. The design process, for each scenario (each set of functional relations), starts from an abstract net with both primitive and compound transitions, and continues by exploding each compound transition until no compound transition exists. This improved modeling capability enables the structuring of Petri Nets in a progressive manner, and facilitate the transformation of the "company policy" of figure 7 to a formal model.

4.3 Knowledge Base Verification

Several analysis techniques for Petri Nets, including reachability trees, behavioral nets, and net invariants, will be used in this research [PETE 81] [JENS 86] [MART 84]. The net invariants, which represent mutually exclusive conditions within the "company policy", can reveal logical conflicts in the specification of the original rules and possibly errors introduced during the modeling process. The reachability tree can be used to detect any deadlock or inconsistencies in the model. The behavioral net can be used to detect redundancies in the net and is a useful tool for reducing the complexity of the model. (see section 8) The programs for computerizing these analysis methods have been developed and applied extensively. Some reduction rules [LEE 85] have also been investigated for reducing the complexcity of nets for further analysis.

However, these analysis techniques were developed for Generalized Petri Nets (GPN), but not applied to Colored Petri Nets (CPN) of which a great diversity of linear functions are associated to the arcs. Therefore, unlikely analysis algorithms for GPN that use integer matrices, analysis algorithms for CPN need to manipulate matrices composed by linear functions. This fact introduces a high complexity in the development and execution of these algorithms. An alternative approach will be to *unfolded* UPN into GPN before they can be analyzed. The unfolding algorithm has been developed and presented in section 8.

4.4 Implementation

We have adopted a fairly new concept in systems integration, known as database interoperability. It is being realized through the development of the Update Dependencies language in the Department of Computer Science, at the University of Maryland [MARK 87]. Database interoperability can be described as the concatenation of the schemata of each of the databases of the application systems, along with a rule set constructed for each separate database, called update and retrieval dependencies. These update and retrieval dependencies control inter-database consistency through inter-database operation calls. We propose the use of Update Dependencies as a special rule specification language, to be used for the implementation of our Knowledge Based System. The algorithm of automatic translation between the UPN and the UD language has been developed and presented in section 9, which reduces dramatically the implementation effort. The details of the Update Dependencies are presented in section 9 and appendix D.

5 Updated Petri Nets for Information Systems: UPN

Petri Nets were originally developed by Carl Adam Petri in his doctoral thesis, 1962, at the University of Darmstadt, West Germany. There have been many reports and papers published on Petri Nets with a wide variety of applications due to their generality and modeling power. Petri Nets can be applied to most systems in representing graphically not only sequential but also concurrent activities. Because of their mathematical representation, they can be formulated into state equations, algebraic equations, and other mathematical models. Therefore, Petri Nets can be analyzed mathematically for the verification of system models. A survey of literature where various types of Petri Nets are used in modeling various systems in general and manufacturing systems in particular, has been presented in section 2.4 and 2.5.

5.1 Motivation For Enhanced Modeling Capabilities

5.1.1 Facilitating Modeling Using A Special Set of Colored Petri Nets - UPN

In our previous modeling approach General Petri Nets (GPN) were used [HARH 90]. However, the number of rules in the knowledge base grew very quickly as more application systems and new data entities were involved, which require much more effort to model the already complex system correctly.

1. Rules introduced by the user make extensive use of variable values, such as "Work center ID" and "Work center Status" as field names of the work center records. With the previous modeling approach, places are interpreted in plain English and do not have specific syntax to represent the structure of data records. For example, a place representing "Work center with r status" does not specify that the work center records have a field named "Status". This information is important to model the rules specification in a consistent and complete manner for both, to clearly represent the logic and to be able to make a more direct translation to knowledge based systems.

2. Every time a new condition (database state) check was needed, a new place was created to represent this checking, with an associated interpretation. However, this newly created place will always have relationships with some places in the other existing scenarios. For example, a place representing "work center with h status and na state" has relationship with and is a sub set of the place representing "work center with h status". This fact can not be considered due to the lack of generality in the syntax of the General Petri Nets previously used, which makes it impossible to

establish relations between places with the similar or identical meaning in different scenarios. This causes integrity problems and modeling errors in the net.

These problems can be partially solved by defining a semantics for the representation. On the other hand, if the system to be modelled is large and fairly complex, the number of places and transitions in the model becomes very large and the representation will be unwieldy. Also, if the systems under consideration have several identical processes, it will be necessary to have several identical subnets in the corresponding model. To resolve these problems, we use a special type of the high level Petri Nets called Colored Petri Nets (CPN). A colored Petri net [JENS 81] is a generalization of a Petri net in which information is aggregated in tokens and arcs. Tokens are distinguishable and are assigned different colors, and arcs are labeled with associated functions. The use of CPN will allow the model designer to work at different aggregation levels.

CPN have more modeling power as GPN. The main advantage of CPN over GPN is the possibility of getting a compact representation of a large and complx system. The definitions of liveness, boundness, invariants, and other properties are similar to those in GPN. However, computations for the verification of these properties can be quite involved since the arcs are not labeled by integers but by functions, so the elements of the incidence matrix are functions and not integers.

We can find an important additional advantage in using CPN for our descriptions if we relax the formalism somewhat. The user does not know a priori some entity features (e.g. number of work cells, parts, operations, etc.). Therefore, the user can not anticipate the complete sets of colors of these features in the model. What we can reflect in the CPN model are the processes for these generic entities. The underlying idea here is to generate a Petri net model that can describe the information as close to the user specification as posible. Once we have this net we can selectively focus the analysis effort in particular areas within the large model. Therefore, we propose to use CPN in model not only the rule base, but also the database changes which will ensure the consistency in representing database status in the CIM system.

5.1.2 Adopting A Hierarchical Design Methodology

The Petri Net designer starts from important abstractions when the Petri net model is created, and then explodes details of the company policy from the abstract model. The previous modeling approach does not have the capability to handle this type of hierarchical design process.

We have employed a new modeling methodology in order to model the company policy with a hierarchical representation, as described in section 4.2, starting from the abstract company policy and exploding it into detailed a domain knowledge base. The detailed

modeling techniques of this hierarchical modeling methodology will be presented in section 6.

5.2 Definition of UPN

This section describes a formalism for knowledge representation of an information system modeled by a special set of Colored Petri Nets, named Updated Petri Nets: UPN. We have extended the primitives of the classical PN's descriptions in order to reflect, more closely, the terminology and semantics involved in our application domain. These primitives do not contribute with new concepts into Petri Net theory (in the sense that they do not increase its analytical capabilities) but allow a procedure to automate and formalize the interpretation process in the domain of information systems.

A UPN net is a directed graph with two types of nodes, places which represent facts or predicates, and transitions which represent rules or implications. Enabling and causal conditions and information flow specifications are represented by arcs connecting places and transitions. Metaknowledge and hierarchical net descriptions are represented by metatransitions (compound operation). We have divided the representation of UPN components in the following four groups:

- Data
- Facts
- Rules
- Metarules

In the following we define the syntax and semantics of these components. To illustrate these aspects, we focus in an example scenario, creation of a work center via MRP II and, particularly, in one of its subnets, the insertion of a work center in MRP II as shown in figure 8, which will be used along this section. The specification of the example is expressed in natural language (see appendix A) and then represented using UPN primitives.

5.2.1 Data Specification

In an information system environment, the user needs to refer to atomic data, and establish relations between different data by structuring information into composed data objects. UPN allows the specification of the following classes of information (in general we refer to any of these data classes by the generic name of "color"):

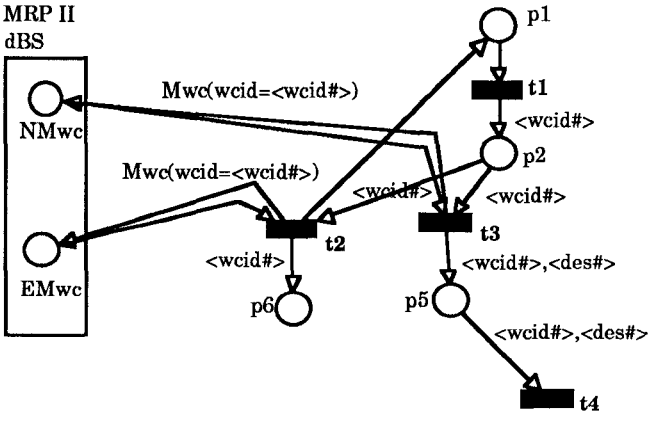


Figure 8: Subnet of the work center creation Scenario to insert a work center via MRP II

Token is data with no information used to represent dicotomical assertions. Formally it can be associated with the integer number 1. It is represented graphically by the symbol: "•".

Atomic data are constants with a lexical representation. Each atomic data is represented by an identifier, whose syntax is an alpha-numerical sequence. Different atomic data will be distinguished by using different identifiers. Analogous atomic data in the information system, will be grouped into sets.

As an example, let us suppose that a part in CAD can be in four different status: w (working), r (release), h (hold), or o (obsolete). We can describe them as four constant symbols (w, r, h, o) which can be grouped to form the status set: $STS = \{w, r, h, o\}$.

Data structure refers to composed data. The data structures used in UPN are a subset of the Cartesian product $S_1 \times S_2 \times \ldots \times S_n$, where S_i is a set of atomic data. A tuple is a member of a data structure $< s_1, s_2, \ldots, s_n >$ where $s_i \in S_i$.

Given the database domain of this work, where the main purpose is the specification of manipulation operations over database records, a special primitive is used to model this data structures. Each record can contain different record fields. Records in UPN are identified by a special field called the identification field and the field components by a field name. Components will be separated by commas (",") and enclosed by "(" and ")" (we will also refer to them as "composed colors").

5.2.1.1 Data sets

Data that is related can be grouped into sets. A data set D can belong to one of the following two classes:

Fixed : when the data set is completely known in advance. It is possible to detail all possible and relevant data occurrences.

An example of this class is the type defined by the set status which has been described above: $MwcSTS = \{w, r, h, o\}$. The information system can specify specific elements of that set (e.g. if work center status is equal to "release" then ...).

Non fixed : when not all of the data set components are completely known in advance. This class refers to information such as part or work center identification numbers. We know, for instance, that the identification can be an integer number with eight digits but we do not know which are the specific numbers which will be provided by the user (in general we do not even know what will be the quantity of parts or work centers in the system). The data set is formed by all instantations of work center identifications: $MwcWCID = \{id/id$ is a work center identification$\}$. This kind of data sets can only be used when the information system makes generic references to that data (e.g. if "any" work center identification number then ...).

5.2.1.2 Example of data specification using UPN

To illustrate the methodology proposed to model data specifications, let us consider the data expressed in natural language shown in section A.1 of the appendix. This example deals with the modeling of specifications related with a work center record. The work center record will be represented in UPN by a data structure and named by the symbol *wc*.

Conventions First, we introduce some syntax conventions to facilitate the identification of the different entities and its relation with the To identify the database for each module of the CAD/CAPP/MRP II/SFC integrated system, a unique letter has been assigned to each of these databases as listed below.

Terminology to identify data bases:

CAD: D
CAPP: P
MRP II: M
SFC: S

Atomic data sets will be named in capital letters. When this data set refers to the data of a specific field in a composed data, we will name it by writing the data base identification followed by the record name identification and followed by the field name in capital letters (e.g. $MwcSTS = \{h, r\}, MwcSTE = \{na, av\}, MwcWCID = \{id/id$ is a work cell identification number$\}$). If there is no doubt to identify the record or the database of some field we will refer just to the field name (e.g. STS and STE).

Color definitions We identify below the fields specified by the user for the different database records (CAD, CAPP, MRP II and SFC).

Fields in the work center record in MRP II are:

wcid:	identification number (no fixed)	
sts:	work center status code	
	h (hold), r (release)	$STS = \{h, r\}$
ste:	work center state	
	na (not available), av (available)	$STE = \{na, av\}$
des:	description (no fixed)	
dep:	department (no fixed)	
cap:	capacity (no fixed)	
res:	resource code (no fixed)	
rat:	rate code (no fixed)	
dh:	dispatch horizon (no fixed)	
esd:	effective start date (no fixed)	
eed:	effective end date (no fixed)	
lp:	wc load profile (no fixed)	

The complete work center record is represented by:

$$Mwc(wcid, sts, ste, des, dep, cap, res, rat, dh, esd, eed, lp)$$

Structured data sets are formed by the Cartesian product of their composed data sets. For example, the data set for a general Mwc record is as follows:

$$MWC = WCID \times STS \times STE \times DES \times DEP \times CAP \times RES \times RAT \times DH \times ESD \times EED \times LP$$

Data representations for CAPP and SFC can be found in appendix B.

5.2.2 Fact specification and classes of places

Fundamental parts of the description of an information system are the facts it contains. Each fact declares a piece of information about some data, or data structure, in the system. The set of all facts form the state of the domain knowledge. To be able to model the access and exchange of information in a consistent and generic way, we need to define specific semantics to describe facts. This problem becomes more important if we consider a modular modeling methodology where the user must be allowed to create different scenarios which can be linked together later on shared knowledge.

Facts in UPN will be represented by places. Places are one of the two kind of nodes in a UPN net and are graphically represented as circles. The fact asserted by one place is determined by the place name and its content. This content defines the marking of the place; we will refer to the marking of place p by $M(p)$. For example, if color A is in place

inprocess, we can interpret that as the fact: "*A is in process*" or $inprocess(A)$ in logic syntax.

Each place p has associated to it a non-empty data set $C(p)$, with $M(p) \in C(p)$, and an initial marking $M_0(p) \in C(p)_{MS}$, where $C(p)_{MS}$ is the set of all finite multi-set over $C(p)$.

Continuing with the previous manufacturing data base example. Let us represent the fact that a work center record exists in MRP II data base. We will represent the information of each database record in two places: one, identified by symbol N, to represent the fact that the record does not exist yet in that database (the record is identified by its identification number) and another place, identified by symbol E, to represent the record exists exists in the database (the record is represented by its identification number and the rest of its fields). So, we represent facts about a work center record in MRP II with two places: $EMwc$, to describe the records that have been already introduced in the SFC database, and $NMwc$, which represents the negation of this fact. These facts can be seen in figure 8 where they are used to represent some user specifications. The interpretation for the rest of the places (p_1, p_2, p_5, p_6) is shown in the figure.

The marking of $EMwc$, $M(EMwc)$, includes all the existing work center records in MRP II at a given time. So, the possible data in place $EMwc$ is restricted to be a work center record, so $C(EMwc) = MWC$, where MWC is as defined before. On the other hand, place $NSwc$ has associated the color set of all possible work center identification numbers because no more information is needed for a non existing record, $C(NMwc) = WCID$. For the rest of the places: $C(p_1) = C(p_2) = C(p_5) = C(p_6) = WCID$.

In order to make use of a modular methodology, UPN allows the splitting of the net structure into separate subnets to create scenarios. Each subnet can be designed and verified separately from the each other, and then the user can follow an incremental process by merging different subnets. Attending to scope considerations for facts, places can be of two different types:

Local scope : when the place is used to represent a local fact (relevant in one specific scenario or subnet) which can not be used as enabled or causal conditions for rules/transitions belonging to other scenarios or subnets. Typically are used to represent intermediate state facts within the decision process. Examples of local places are p_1, p_2, p_5, p_6 in figure 8.

Global scope : when the place is used to represent a fact relevant (accessible) to different scenarios or subnets. Global places must be refered by the same name in all of its occurrences. Typical examples of global places are facts related with database state information such as places $EMwc$ and $NMwc$ defined above (figure 8).

In relation with the interface, there are two special types of places to specify the

interchange of information with the user interface system (these places must have local scope):

Input places represent facts whose initialization will be generated by an external action (i.e. by a request from the user through the interface). They do not necessary need incoming arcs and therefore the net will stall and wait for the user's input to continue firing. For example, after the creation of a work center record in MRP II, the net will need the user to trigger the releasing transaction in order to continue firing the release transition for releasing the work center record. Input places are also called source places. An example of input place is represented by place p_1 shown in figure 8, it needs the introduction of some value in $< wc\# >$.

Output places are used to represent situations where some information must exit (i.e. some information to be displayed). They are used to indicate the termination of a token (tokens arriving at the output place will disappear from the net). For example, an output place represents an inconsistency found in the system and an error message was printed on user's screen. Output places are also called sink places. An example of output place is represented by place p_6 shown in figure 8, the value provided in $< wc\# >$ will be displayed.

From the net point of view, the concept of input and output places allows the conservation of tokens in the net model and, on the other hand, provide a mechanism to specify terminal facts during consistency and completeness verification of the model knowledge.

Guidance for place interpretation [1]

5.2.3 Rule specification

Another important part in modeling information systems is the representation of information flow specifications. Here, we are considering domains where the user specifies

[1] Any interpretation can be assigned to any place but this absolute flexibility can produce several design problems such as: facilitate the possibility for inconsistences to arise, lead to an unnecessary explosion of net components and increase the difficulty to automate the design cycle. In order to reduce these inconveniences, we will provide some guidance:

1. Use a negative predicate only if it is very clear that it is going to be used separately from its affirmative one.
2. Restrict at a minimum the use of negative predicates.
3. Converting rules between a negative predicate and the associated set of predicates, ex: work center is not in w status means it is in either r, h, or o status. $\neg WC(sts = h) \equiv WC(sts = STS - \{h\})$.

information flow policies using "if then" rules. Rules are expressed in UPN by means of other type of net nodes called transitions. Transitions are graphically represented as rectangles. In the same way as places, any transition t has a data set, $C(t)$, associated with it.

The net in figure 8 shows four transitions (t_1, t_2, t_3, t_4) and its associated interpretation. Its associated data sets are: $C(t_2) = C(t_3) = MWC, C(t_1) = C(t_4) = WCID$.

In the rest of this section we explain how several components of the rules (preconditions, postconditions, variables, etc) are represented in UPN.

5.2.3.1 Variables

We will characterize some relations to generic data (in the sense that the relation can hold for different data occurrences) by using variables. Each variable $v : D$ has a name v and an associated data set D. The scope of a variable is the transition including its surrounding arcs.

The name is an alpha-numerical sequence closed by symbols "<" and ">". Variables belonging to non fixed data sets, will be identified by adding the symbol "#" at the end of the alpha-numerical sequence of its name. For example we can write $< wc\# >$ to identify a generic work center identification number, $< wc\# >: WCID$.

5.2.3.2 Arcs

Arcs are relations which connect facts (places) and rules (transitions) and constitute the rule preconditions and postconditions.

Arc functions:

Each arc has attached to it an arc-function F containing a set of variables or constants $\{v_1 : D_1, v_2 : D_2, \cdots, v_n : D_n\}$, where D_i identifies the data set of the variable or constant v_i. Moreover it is required that F defines a mapping from $\{D_1 \times D_2 \times \cdots \times D_n\}$ into $C(p)_{MS}$, where the place p is source/destination of the arc.

Our experience in modeling rule specifications with UPN has shown that the majority of policies can be modeled with a small number of simple arc functions. We will restrict the UPN models to use one of the following functions:

identity function: $< c_i > \longrightarrow < c_i >$
i.e. arc t_1, p_2 in figure 8 : $< wc\# > \longrightarrow < wc\# >$
decolorizing function: $< c_i > \longrightarrow \bullet$; is the token or neutral color
i.e. arc t_2, p_1 in figure 8 : $< wc\# > \longrightarrow \bullet$
ith-projection function: $(< c_1 >, < c_2 >, \cdots, < c_i >, \cdots, < c_n >) \longrightarrow (< c_i >)$
i.e. arc $t_2, EMwc$ in figure 8 :
$(wcid, sts, ste, des, dep, cap, res, rat, dh, esd, eed, lp) \longrightarrow wcid$
instead to write all the fields, we will explicitly identify the fields which are involved and we consider the rest implicitly represented (see below):

$(wcid =< wc\# >) \longrightarrow < wc\# >$

Arc types

Arcs identify information flow and flow conditions. In order to differentiate these flow features UPN provide different types of arcs:

Enabling arcs are directed arcs which connect a place/action with a transition/rule and define a precondition for the transition/action. They indicate which data must mark each place in order to enable a transition as well as which data must be removed from that place on firing.

If the content of the place is a data structure UPN, we will refer to data in specific fields by using projection functions. The projection positions are indicated by the field names. We will indicate the values associated to that fields with the symbol "=". For example to indicate an available state (av) in some work center record ($wcid$) in shop floor control we must write: $Swc(wcid =< wc\# >, ste = av)$. In this case, UPN checks for the presence of a Swc data structure with some id and a specific ste.

Additionally, UPN allow the reference to data with no relation to the control of the information flow. It can be used to specify the coping of some fields from a data structure to another. The presence of that information in the data structure is not a condition for the enabling of the rule and, therefore, UPN do not check for that. We will indicate this non conditional specification in the same as with the conditioned one but using the symbol "−". In the previous example, we can refer to the department information (dep) in a non conditional sense we must write: $Swc(id =< wc\# >, ste = av, dep - < dep\# >)$.

We will identify the condition part of an enabling arc from place p to transition t by $Ic\_(p,t)$ and its no condition part by $In\_(p,t)$. It is verified that $Ic\_(p,t), In\_(p,t) \in [C(t)_{MS} \longrightarrow C(p)_{MS}]_L$ where $[...]_L$ denote a set of linear functions (in UPN this set is restricted to be: identities, decolored and ith-projections functions).

Causal arcs are directed arcs which connect a transition/action with a place/fact and define a postcondition for the transition/action. Causal arcs describe modifications to be performed in the state of the net when the transition/rule is fired, and more concretely, they indicate which colors must be added to a place on firing.

We will identify the condition part of a causal arc from transition t to place p by $Ic_+(p,t)$ and its no condition part by $In_+(p,t)$. It is verified that $Ic_+(p,t), In_+(p,t) \in [C(t)_{MS} \longrightarrow C(p)_{MS}]_L$.

Variables used for conditional specifications in the enabling arcs must be also used as conditionals, "=", in the causal arcs (similar for non conditional, "−", specifications).

Checking arcs indicate which data must mark each place in order to enable a transition but no data is removed. It can be represented as an enabling and causal arc together.

It is also possible for a transition to request information from the user interface. This is done whenever an arc function requires data that has not been provided by the variables in the enabling or checking arcs. Symbol "?" in the arc functions also indicates an information request.

5.2.3.3 Predicates

Each transition can have a predicate, $PRED$, associated to it. This predicate impose additional constraints in the enabling of the transition. This predicate can only contain those variables which are already in the immediate surrounding arc functions. The predicate is restricted to have one of the following logical operators: $\neg(not), \vee(or)$ and $\wedge(and)$. To avoid degenerate transitions, the predicate must differ from the constant predicate $false$. The predicate is supposed to be $true$ by default.

5.2.3.4 Transition data sets

Now, we can describe more specifically the data sets associated to the transitions, $C(t)$, as a function of the data sets associated to the transition variables:

$$C(t) = \{(d_1, d_2, \cdots, d_n) \in D_1 \times D_2 \times \cdots \times D_n \mid (PRED(v_1, v_2, \cdots, v_n))(d_1, d_2, \cdots, d_n)\}$$

where $PRED$ is the predicate attached to t and $V(t) = \{v_1 : D_1, v_2 : D_2, \cdots, v_n : D_n\}$ is the set of all variables appearing free in the immediate surrounding arc functions.

Let $V_Ic(t)$ and $V_{In}(t)$ be the set of variables associated to $Ic\_(t)$ and $In\_(t)$ respectively, then it must be verified that $V_{Ic}(t) \bigcap V_{In}(t) = 0$ and $V_{Ic}(t) \bigcup V_{In}(t) = V(t)$

5.2.3.5 Transition enabling and firing

Let $UPN =< P, T, C, Ic\_, Ic_+, In\_, In_+, M >$ be a UPN net where P is the set of places and T is the set of transitions in UPN, and $C, Ic\_, Ic_+, In\_, In_+, M$ are as defined before.

A transition $t \in T$ is said to be enabled with respect to a data $c \in C(t)$ if the current marking M is such that $M(p) \geq Ic\_(p,t)(c), \forall p \in P$.

An enabled transition may be chosen to fire. The firing of a transition t with respect to a data c consists of removing $Ic\_(p,t)(c)$ data from each of its input places and adding $Ic_+(p,t)(c) + In_+(p,t)(c)$ data to each of its output places.

We may think of a firing as an event which may take place if certain conditions are satisfied. Each firing will create a new set of conditions, and the total number of data in the net may change after each firing.

5.2.3.6 Example of rule specification

In order to clarify the concepts introduced above we will model a rule from the "creation of work center from MRP II" scenario described in appendix A. Let us consider the following user-provided rule specification for "release work center from MRP II":

```
   MRP II user enters additional information (Capacity, Resource Code, and
Effectivity Start Date) through a modify transaction.  MRP II user then
releases the WC. Otherwise, if the additional information was not entered,
the system would prompt for it during releasing of the WC in MRP II.
check :
     WC ID exists in MRP II
     WC with h status exists in MRP II
     All the necessary data fields are filled
     WC record does not exist in CAPP
update :
     WC record status changed from h to r in MRP II
     Skeletal WC Record automatically created in CAPP with w status
```

Figure 9a shows a graphical representation for this rule, expressed as a UPN net.

The rule, works with work center records belonging to two different databases: MRP II and CAPP. The contents of these records is described in appendices B and A.1, and they are represented by the following data structures :

$$Mwc(wcid, sts, ste, des, dep, cap, res, rat, dh, esd, eed, lp)$$
$$Pwc(wcid, sts, des, dep, esd, eed, hp, sr, fr, we, acc, tct, fct, sct, trt, tat, rtt)$$

Places $NMwc$ and $NPwc$ represent the non existence of the record in MRP II and CAPP databases respectively. On the other hand, places $EMwc$ and $EPwc$ represent the existence. The data sets are:

$$C(NMwc) = WCID, C(NPwc) = WCID, C(EMwc) = MWC, C(EPwc) = PWC$$

The rule is represented by transition t_2 and the set of variables of t_2, $V(t_2)$, is:

$$\{< wcid\# >: WCID, < cap\# > CAP, < res\# >: RES, < esd\# >: ESD\}$$

In order to explain the transition enabling and firing procedure, let us consider the following marking: $M_0(EMwc) = Mwc(wcid = A123, sts = w)$ and $M_0(NPwc) = A123$. It is clear that transition t_2 is enabled with respect to the data: $c = \{< wcid\# > \leftarrow A123, Mwcsts \leftarrow w\}$. The data structure in place $EMwc$, $Mwc(wcid = A123, sts = w)$, satisfy condition $Mwcsts = w$ in arc function, so variable $< wcid\# >$ can be substituted by value $A123$. If we look now to the arc connecting place $NPwc$ with t_2, we can find a substitution $< wcid\# > \leftarrow A123$ that match with the previous substitution. This

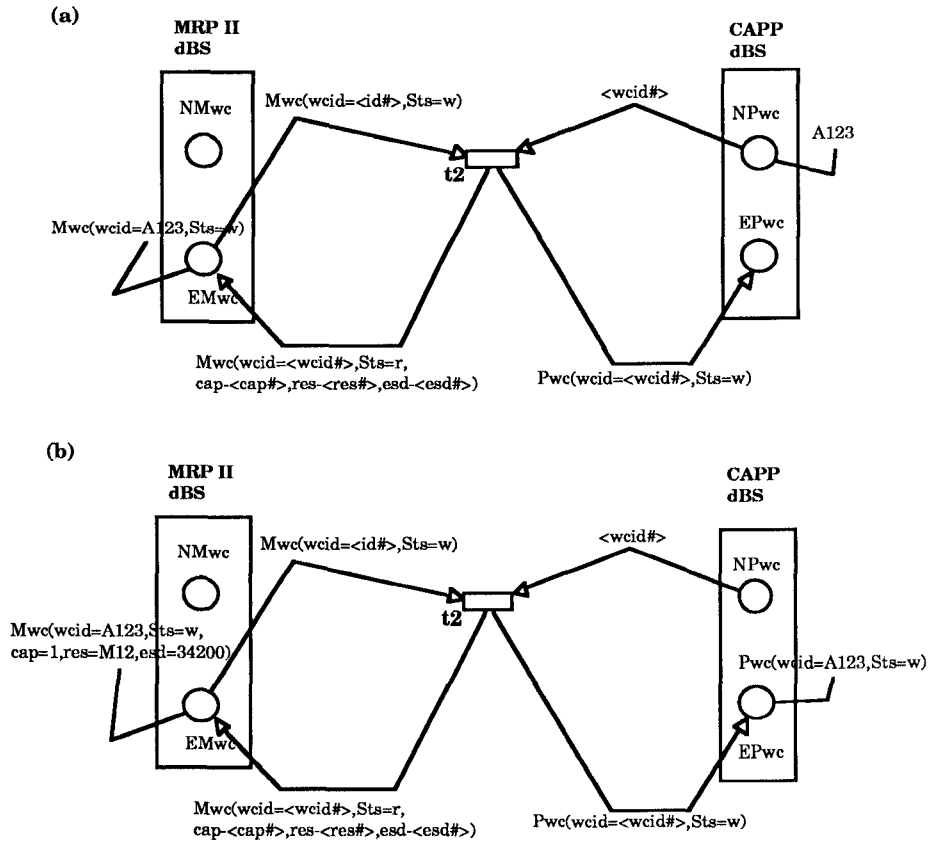


Figure 9: UPN representation for "release work center from MRP II" rule.

substitutions verify the enabling conditions:

$$M(EMwc) \geq Ic\_(EMwc, t_2)(< wcid\# > \leftarrow A123, Mwcsts \leftarrow w, < cap\# >, < res\# >, < esd\# >)$$
$$M(NPwc) \geq Ic\_(NPwc, t_2)(< wcid\# > \leftarrow A123, Mwcsts \leftarrow w, < cap\# >, < res\# >, < esd\# >)$$

where:

$$M(EMwc) = Mwc(wcid = A123, sts = w)$$
$$Ic\_(EMwc, t_2)(< wcid\# > \leftarrow A123, Mwcsts \leftarrow w, < cap\# >, < res\# >, < esd\# >) = Mwc(wcid =$$
$$M(NPwc) = A123$$
$$Ic\_(NPwc, t_2)(< wcid\# > \leftarrow A123, Mwcsts \leftarrow w, < cap\# >, < res\# >, < esd\# >) = A123$$

If the transition t_2 is enabled, it can be fired, in this case with respect to data:

$$(< wcid\# > \leftarrow A123, Mwcsts \leftarrow w, < cap\# > \leftarrow ?, < res\# > \leftarrow ?, < esd\# > \leftarrow ?)$$

Variables $< cap\# >, < res\# >, < esd\# >$ that have no substitutions must be requested to the user interface.

In the firing:

- $Mwc(wcid = A123, sts = w)$ is removed from $M(EMwc)$ and
- $A123$ is removed from $M(NPwc)$

and then:

- $Mwc(wcid = A123, sts = r, cap =< cap\# >, res =< res\# >, < esd\# >= esd)$ is added to $M(EMwc)$ and
- $Pwc(wcid = A123, sts = w)$ is added to $M(EPwc)$.

The net state after firing is shown in figure 9b.

5.2.4 Metarule specification

Metarules provide a higher level representation mechanism. Metarules are rules which express knowledge about other rules providing a mechanism to establish relations between rules. Metarules are sometimes used to increase the efficiency in the rule control process, a metarule can focus on a restricted set of rules that are relevant to the decision process being made, these avoid the effort of considering rules that has no relation with that decision process.

Metarules are mainly used in UPN as a mechanism to define sub-nets. They are used in two different directions to allow a structural and hierarchical composition of the knowledge:

- **Horizontal** to define sub-nets for different scenarios.
- **Vertical** to define relations between nets at different levels of abstraction.

5.2.4.1 Horizontal composition of rules: hmrules

Rules at the same level of abstraction can be related to form sub-nets. This horizontal composition allow the aggregation of rules under specific criteria. Horizontal relations are established by means of what we call "hmrules". A hmrule hm_a, specify a relation in a set of transitions $hm_a^t = \{t_1, t_2, \ldots, t_m\}$ where $m \geq 1$ and t is defined at the level of abstraction $a, \forall t \in hm_a^t$. The sub-net, defined by metarule hm_a is composed by the set of transitions hm_a^t and the places that are connected to the surrounding arcs of transitions in hm_a^t.

Let $UPN =< P, T, C, Ic^-, Ic^+, In^-, In^+, M >$ be a UPN net, $HM = \{hm_1, hm_2, \ldots, hm_n\}$ the set of sub-nets in UPN defined by the hmrules, it must be verified that $hm_i \bigcap hm_j = \emptyset, \forall i, j \in \{1, \ldots, n\}$ with $i \neq j$, what means that one transition can not belong to more than one sub-net.

Hmrules are generally used to identify scenarios or sub-nets used refine a transition at a lower level of abstraction. An example for the interest of horizontal composition can be seen in figure 11. In that example, an scenario to "create a work center via MRP II" is defined by aggregatin the rules: "insert a work center in MRP II", "release a work center in MRP II", "release a work center in CAPP", and "release a work center in SFC".

5.2.4.1 Vertical composition of rules: vmrules

The vertical composition of rules is used to establish relations between one rule and other rules which define knowledge at a lower level of abstraction. Vertical composition in UPN allows the composition of rules forming an abstraction hierarchy. This abstraction method makes easier the design and verification process by allowing the designer to follow a stepwise refinement process working at different levels of detail (section 6).

Given a rule, at (high level abstraction transition), and a level of abstraction, $i - 1$, a vertical metarule (called "vmrule"), vm_{i-1}^{at}, is defined by a three tuple $< hm_i, Ivm, Ovm >$, where:

- hm_i is a hmrule that identify a sub-net: $UPN' =< P', T', C', I'c^-, I'c^+, I'n^-, I'n^+, M' >$, which refine t at a higher level of detail i,
- $Ivm \neq \emptyset, Ivm \subset T'$ is the set of input transitions to hm_i, and
- $Ovm \neq \emptyset, Ovm \subset T'$ is the set of output transitions from hm_i (eventually Ovm can be equal to Ivm).

Let UPN_{i-1} and UPN_i be the UPN nets representing the same information system at the levels of abstraction $i-1$ and i, respectively. The new UPN net at the level of abstraction i, UPN_i, is defined as follows:

1. $P_i = P_{i-1} \bigcup P'_i$
2. $T_i = T_{i-1} - \{at\} \bigcup T'_i$
3. All the incomming arcs of UPN_i and UPN'_{i-1} are preserved but the ones related with transition at which are added to the transitions in Ivm:
   $Ic_i^-(t,p) = Ic_{i-1}^-(at,p)$ and $In_i^-(t,p) = In_{i-1}^-(at,p), \forall p \in P_{i-1}, \forall t \in Ivm$
4. All the outgoing arcs from UPN_i and UPN'_{i-1} are preserved but the ones related with transition at which are added to the transitions in Ivm:
   $Ic_i^+(t,p) = Ic_{i-1}^+(at,p)$ and $In_i^+(t,p) = In_{i-1}^+(at,p), \forall p \in P_{i-1}, \forall t \in Ovm$
5. $M_i(p) = M_{i-1}(p), \forall p \in P$ and $M_i(p) = M'_i(p), \forall p \in P'$

Vertical composition in UPN is performed in such a way that behavior at the higher level of abstraction is also preserved when working at lower level of abstraction.

An example of this vertical composition is the relation between the rule "insert a work center in MRP II" and the metarule of "create a work center via MRP II" shown in figure 11 which will be explained in the next section.

5.2.5 Database domain constrains

So far, the user is only constrained by the restrictions imposed by the UPN primitives and is free to use them in order to model any structure.

Generally, when a person is working in a specific domain of application, its specifications assume as valid certain features that he does not have to explicitly express. For example, in the context of manufacturing database systems, features like the way in which the database management system stores the data are considered as a basis and they are not specified by the user.

6 Hierarchical Modeling Methodology

6.1 Introduction

Generally speaking, any "company policy", starts from the specification of general rules which describe the aggregate operations for a given entity within the system. These rules will then be further refined into more detailed specifications on a step by step basis until no aggregate operations are left. Following similar concept, a hierarchical modeling method using UPN has been developed to model the control of information flow among different manufacturing applications. However, it can also be applied to model the interaction between any computerized application modules in a company. The model of the "company policy" is then constructed following this approach: an abstract Petri net representing the overall company policy modeled using compound transitions, corresponding to the vertical meta rules described in section ?? for operations performed in the various application systems will be drawn first, followed by refining each of the compound transitions into a set of operations using primitive and /or compound transitions until no compound transitions are left. This technique will be applied on each of the scenarios involved in our CAD/CAPP/MRP II/SFC integrated system. Once the separate scenarios are modeled and analyzed, they need to be synthesized in order to form a coherant model representing the integrated system. It is very difficult not only to identify the common structure of separate nets but also to decide whether to combine them or not in a systematic way. The breakthrough in our modeling approach is to intoduce the modeling of the Databases of those manufacturing applications using UPN by defining the database states as global variables (see 5.2.2), and synthesizing nets through them.

In this section, a top-down refinement technique was first developed, and detailed with its basic methodology and examples of modeling rules in the CAD/CAPP/MRP II/SFC integrated system. Follwoing the same approach, a net synthesis technique was also developed. Their detailed mathematical formalisms are included in appendix D.

6.2 Top-Down Refinement Technique

There have been some efforts in refinement and reduction methods for Petri Nets modeling and analysis. A stepwise refinement method has been proposed by Vallette [VALE 79] to deal with the noninstantaneous transition firing. It demonstrates the substitution of a transition by a *well-formed* block. Some extended work has also been done by Suzuki and Murata in [?].Their method allows to proceed with the description and the analysis of a control structure by a stepwise refinement, and this concept is adopted into the refinement procedure of our approach. However, our modeling approach does not restrict in the *well-formed* block, but also allows new constraints to be introduced during the refinement procedure. A review of synthesis techniques for Petri Nets, which includes both

Figure 10: Top-down refinement schema.

the refinement and the sysnthesis techniques developed in recent years, was presented in [JENG 90].

Our approach also reflects the management hierarchy in a typical company. An abstract or aggregate company policy will first be defined by the top level management. Then, it will be refined to many detailed rules by managers and experts from lower levels of authority. The top-down refinement technique described below also aggrees with the structure of general rule specification language.

The following sections present first the basic methodology, then the system specification of a sample scenario in our CAD/CAPP/MRP II/SFC integrated system, the UPN model of it, and finally the refinement procedure and formation of the projected low level net.

6.2.1 Basic Methodology

An abstract company policy, which is represented by one scenario of the integrated system as shown in figure 10, is first modeled with a set of compound transitions using the horizontal composition of rules (see section 5.2.4), and with the interaction to the databases involved. Each of these compound transitions represents a metarule at the

highest abstract level which is then refined, based on domain expert knowledge, into a lower level sub-net with a set of primitive transitions and compound transitions using the horizontal and vertical composition of rules (see section 5.2.4). The refinement includes the construction of horizontal relationships within the sub-net and vertical relationships with the higher level net based on the detailed domain knowledge. The horizontal relationship is built upon the precedence relations of operations within the sub-net and it allows extra constrains be introduced. The vertical relationship is based on the identification of the input and output transitions which are mainly the links between the sub-net with the input places and output places of its parent compound transition. This refinement procedure continues until there is no compound transition left in the net.

The scope of the variable is the metarule (sub-net) in which variable hold the same value. For example, Wcid value remains the same within each metarule, but not between different metarules unless it was transferred through calling one metarule from the other.

The result is a set of inter-connected sub-nets, which can then be projected into one single net with all primitive transitions, representing primitive operations within application systems. The projected single net will be used to verify the underlying company policy and expert knowledge.

6.2.2 System Specification of a Sample Scenario

A set of scenarios of the CAD/CAPP/MRP II/SFC integrated system, which form a knowledge base to control information flow among manufacturing applications, have been developed as described before. In this section, one scenario, Establishing New Work Centers in the system (Add): ADD from MRP II, is detailed in plain English like most company policies. Both the common data fields of work centers among manufacturing applications and precedence constraints of operations within those applications on work centers, are described by the company policy and detailed domain expert knowledge.

Sample scenario - Establishing New Work Centers in the system (Add): ADD from MRP II

MRP II users, have the sole authority to establish new work centers. This is because MRP II is the execution function in most manufacturing organizations, and is used for all types of purchasing and procurement activities. The detail specification of this scenario has been developed as follows.

> To establish a new workcenter in the system, MRP II users must provide the following basic information, in addition to assigning the identification number.
>
> - I.D. Number
> - Description
> - Department

Subject to the condition that this I.D. Number does not already exist in the system, this work center record is established in MRP II with its status set to hold. MRP II users then finalize all the other work center details needed in MRP II module as follows:

- Capacity
- Resource Code
- Rate Code
- Dispatch Horizon
- Effectivity Start Date

Then the MRP II user releases the work center.

This data may be entered separately as each data item becomes known, by using the modify transaction provided in the system model. Otherwise, if the additional information was not entered, the system will prompt for them during releasing of the WC in MRP II as described below. In addition to the data fields mentioned, effectivity end date, status code, work center state, and work center load profile are also part of the work center record in MRP II. The status code is not a user input, but is automatically updated from "h" for hold to "r" for released by the release transactions on the work center. The work center state is maintained by SFC users and is not provided at this stage. The work center load profile will be entered and maintained by SFC and updated automatically in MRP II after the work center is allocated for job scheduling.

Invoking the work center release transaction in MRP II triggers a set of consistency checks, which are as follows: the I.D. Number provided must exist in MRP II with hold status; all the required data fields should have been filled, and any data fields left out by users are requested at this stage with the help of system generated prompts. If all these checks are satisfied, the system changes the work center status code from 'hold' to 'released', and a skeletal work center record is automatically created in the work center file in CAPP, with its status set to 'working' as well as a work center record in the work center file in SFC with its status set to 'hold'. These work center records contain all the common information between MRP II and CAPP, and between MRP II and SFC.

CAPP users then input the detailed technical information regarding that work center. The following fields are required to be completed in CAPP, before the work center can be given a released status, and made effective and ready to be used in process plans. If a particular data field does not apply to the specific work center, then 'inapplicable' will be entered automatically. However no field can be left blank.

- Horse Power
- Speed Range
- Feed Range
- Work Envelope
- Accuracy
- Tool Change Time
- Feed Change Time
- Speed Change Time
- Table Rotation Time
- Tool Adjusting Time
- Rapid Traverse Time

Similar to the MRP II, invoking the work center release transaction in CAPP triggers a set of consistency checks, which are as follows: the I.D. Number is checked to ensure that it exists in CAPP a work center file with status set to 'working'; all the required data fields should have been filled, and any data fields left out by users are requested at this stage with the help of system generated prompts. Upon the satisfaction of these checks, the work center gets a released status in CAPP and the common data fields between CAPP and SFC are automatically copied from CAPP to SFC.

This scenario ends by releasing the work center in the SFC module. Invoking the work center release transaction in SFC triggers a consistency check: the I.D. Number is checked to ensure that it exists in the SFC work center file with status set to 'hold'; the WC status in both MRP II and CAPP are set as "r" to ensure all the necessary information have been provided; the current date is past the effectivity start date. The work center state is then automatically set to "AV" in SFC and MRP II for being available. Upon the satisfaction of these checks, the workcenter gets a released status in SFC and the work center state is updated in MRP II.

1. MRP II user enters the basic data (WC ID, Description, Department) to create the Work Center Record with *h* status in MRP II.

   checks WC ID not exist in MRP II

2. MRP II user enters additional information (Capacity, Resource Code, Rate Code, Dispatch Horizon, and Effectivity Start Date) through modify transaction. MRP II user then releases the WC. Otherwise, if the additional information was not entered, the system will prompt for them during releasing of the WC in MRP II.

   checks WC ID exists in MRP II
   WC with *h* status exists in MRP II
   All the necessary data fields are filled
   WC record does not exist in CAPP

   updates WC record status changed from *h* to *r* in MRP II
   Skeletal WC Record automatically created in CAPP with
   w status

3. CAPP user enters additional information (Horse Power, Speed Range, Feed Range, Work Envelope, Accuracy, Tool Change Time, Feed Change Time, Speed Change Time, Table Rotation Time, Tool Adjusting Time, and Rapid Traverse Rate) through modify transaction. CAPP user then releases the WC. Otherwise, if the additional information was not entered, the system will prompt for them during releasing of the WC in CAPP.

   checks WC ID exists in CAPP
   WC with *w* status exists in CAPP
   All the necessary data fields are filled
   WC record does not exist in SFC

   updates WC record status changed from *w* to *r* in CAPP
   WC Record automatically created in SFC with *h* status

4. SFC user releases the WC with the work center state as *av* for being available

   checks WC ID exists
   WC with *r* status exists in MRP II and CAPP
   WC with *h* status exists in SFC

   updates WC record status changed from *h* to *r* in SFC,
   ans state changed from *na* to *av* in SFC and MRP II

6.2.3 UPN model of a Sample Scenario

Once system specification is written down, a model representing it can then be developed to simulate its logic as described in this section, and furthermore to detect the inconsistency and incompleteness in it which will be described in section 8. A formalism for modeling metarules using UPN described in section 5.2.4, is applied in the refinement procedure. The UPN model is presented in two forms, graphically (UPN graph) and mathematically (UPN incidence matrices).

Sample scenario - Establishing New Work Centers in the system (Add): ADD from MRP II

This scenario is one metarule itself, which in turn involves four metarules - "insert a work center in MRP II", "release a work center in MRP II", "release a work center in CAPP", "release a work center in SFC" - connected to each other based on the horizontal composition formalism and the system specification. Each of these metarules is then refined to one sub-net and connected to the higher level net based on the vertical composition formalism.

UPN graph

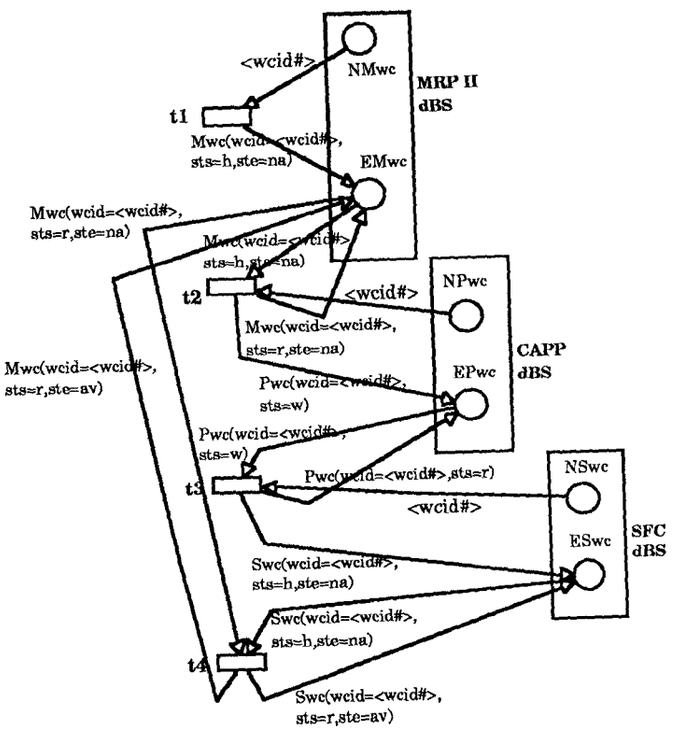
The above example modeled in UPN is shown in figure 11. An abstract company policy, which represents the scenario "create a work center from MRP II", is modeled with compound transitions (blank bar), representing (t1) "insert a work center in MRP II", (t2) "release a work center in MRP II", (t3) "release a work center in CAPP", (t4) "release a work center in SFC", and the sequence and constraints among these compound transitions.

One of these compound transitions (metarules), "insert a work center in MRP II" of figure 12 is refined into lower level sub-net. This sub-net contains four primitive transitions representing (t1-1) "request Wcid", (t1-2) "output error message", (t1-3) "request Des and Dep", (t1-4) "add work center record into the MRP II DB", which can not be further refined. It also defines the horizontal composition among them and the interaction with the MRP II Database.

UPN incidence matrix

An incidence matrix ($C = C^+ - C^-$) is an algebraic representation of one Petri Net with columns representing transitions, rows representing places, and the matrix elements representing the weights of arcs from transitions to places (C^+) and from places to transitions(C^-). It enable us to model and analyze the dynamic behavior of Petri Nets mathematically. The matrix elements in GPN are interger values representing simple connection between places and transitions. However, by using UPN, the elements in the incidence matrix become functions.

A State equation to transform one marking of Petri Nets into the other is shown below.



t1: add a new work center record in MRP II
t2: release the work center in MRP II
t3: release the work center in CAPP
t4: release the work center in SFC

Figure 11: CPN of the Scenario to create a work center from MRP II at abstract level

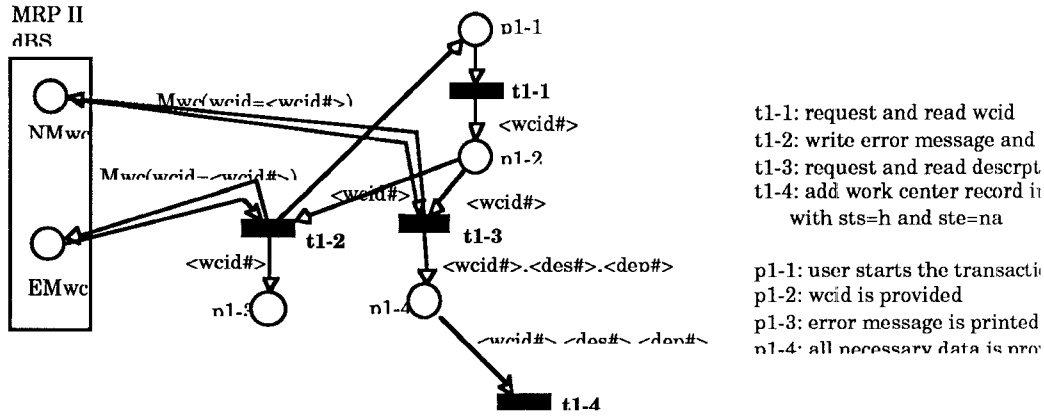


Figure 12: Sub net of the work center creation scenario to insert a work center from MRP II

$M_t = M_{t-1} + Cx$

where M_t and M_{t-1} are the markings at time t and $t-1$. and x is a vector containing the numbers of transitions firing.

The following matrix described the UPN model in figure 11.

$C^- =$

	t_1	t_2	t_3	t_4
$NMwc$	$< wcid\# >$	0	0	0
$EMwc$	0	$(< wcid\# >, h, na)$	0	$(< wcid\# >, r, na)$
$NPwc$	0	$< wcid\# >$	0	0
$EPwc$	0		$(< wcid\# >, w)$	0
$NSwc$	0	0	$< wcid\# >$	0
$ESwc$	0	0		$(< wcid\# >, h, na)$

$C^+ =$

	t_1	t_2	t_3	t_4
$NMwc$	0	0	0	0
$EMwc$	$(< wcid\# >, h, na)$	$(< wcid\# >, r, na)$	0	$(< wcid\# >, r, na)$
$NPwc$	0	0	0	0
$EPwc$	0	$(< wcid\# >, w)$	$(< wcid\# >, r)$	0
$NSwc$	0	0	0	0
$ESwc$	0	0	$(< wcid\# >, h, na)$	$(< wcid\# >, r, av)$

The following matrix described the UPN model in figure 12.

$C^- =$

	$t_{1.1}$	$t_{1.2}$	$t_{1.3}$	$t_{1.4}$
$NMwc$	0	0	$< wcid\# >$	0
$EMwc$	0	$(< wcid\# >)$	0	0
$p_{1.1}$	1	0	0	0
$p_{1.2}$	0	$< wcid\# >$	$< wcid\# >$	0
$p_{1.3}$	0	0	0	0
$p_{1.4}$	0	0	0	$< wcid\# >, < des\# >, < dep\# >$

$C^+ =$

	$t_{1.1}$	$t_{1.2}$	$t_{1.3}$	$t_{1.4}$
$NMwc$	0	0	$< wcid\# >$	0
$EMwc$	0	$(< wcid\# >)$	0	0
$p_{1.1}$	0	1	0	0
$p_{1.2}$	$< wcid\# >$	0	0	0
$p_{1.3}$	0	$< wcid\# >$	0	0
$p_{1.4}$	0	0	$< wcid\# >, < des\# >, < dep\# >$	0

6.2.4 Procedure for the Formation of Complete Lowest Abstraction Net

Following the same technique in refining the Petri Net from the highest abstract level step by step to the lowest abstract level until no compound transition left, the result is a set of sub-nets inter-connected together through horizontal and vertical composition. These sub-nets can then be projected into one single net with only the primitive transitions. The projection of previous example is shown below.

The abstract net and the refined net are then connected based on the vertical composition formalism. The t1-4 was identified as both the input and output transition, and was connected from the input places of t1 and to the output places of t1. The Database places are merged together for consistency. Following the basic methodology of the refinement technique described in section 6.2.1, input transitions will be connected to "all" the input places of the refined compound transition and output transitions will be connected with "all" the output places of the refined compound transition. The result of refining t_1 of figure 11 is shown in Figure **??**.

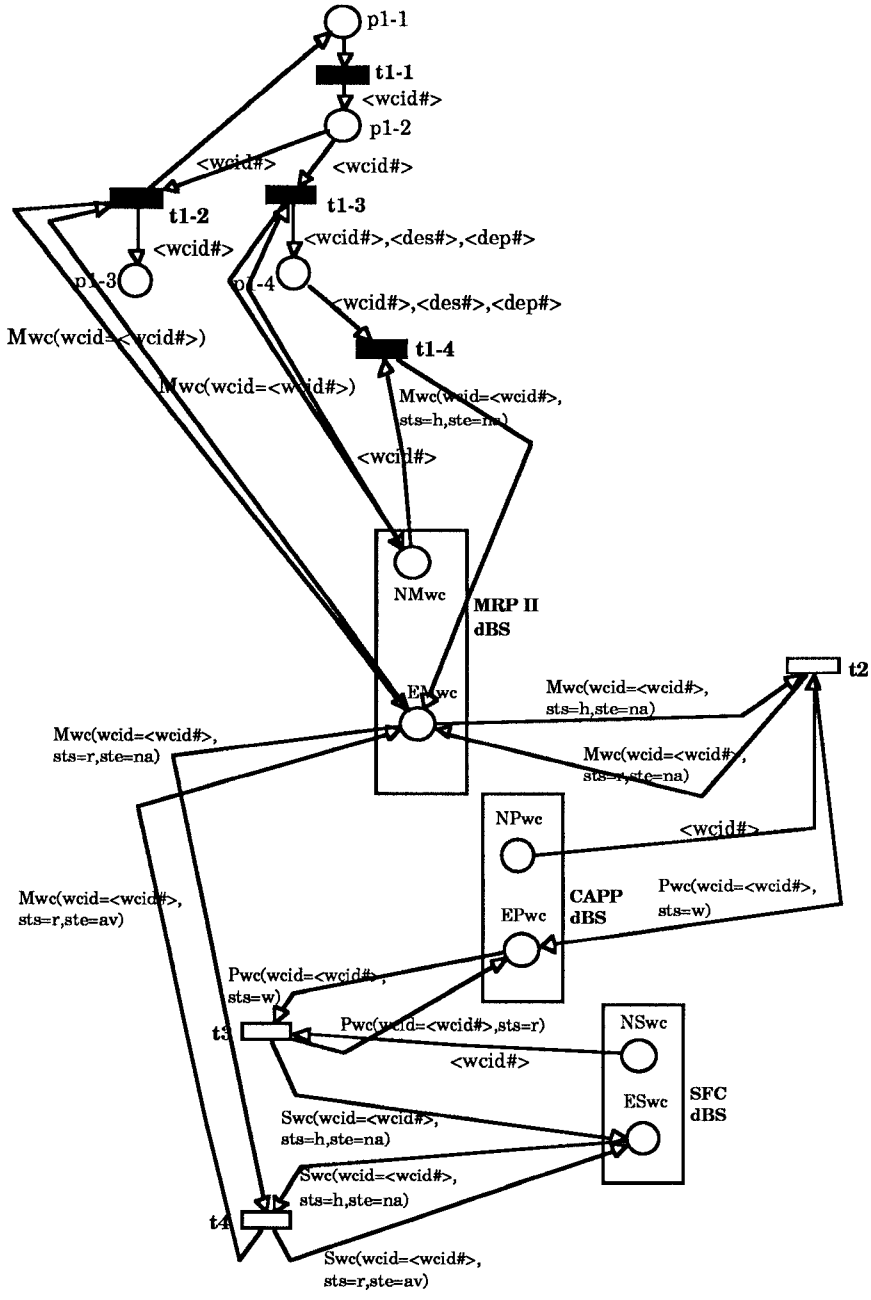


Figure 13: Partially refined UPN of the scenario to create a work center from MRP II

The resulted incidence matrix of the refined net is shown below and the formalism of creating this incidence matrix is described in appendix D.

6.3 Synthesis Technique

— follow the same structure in the previous section ————

Synthesis of different scenarios' models to form a coherent model, which represents an integrated company policy, is the second step of designing a knowledge based system if the user has followed the right methodology to define rules. Generally speaking, Synthesis is made by merging one place into another place if its interpretation is the same as the other one, or if its interpretation is included into the other place. An example of the first case is places with the same interpretation of "work center status is r", and an example for the second case is places with interpretations of "work center status is r" and "work center status is r and state is av" respectively. It is necessary to define a specific syntax for the interpretation of places, which is used to represent facts and provides a straightforward basis to merge common places. However, as the integrated system become very complex with massive data entities involved, it becomes very difficult to identify the common places according to the interpretation of them. As mentioned before, a breakthrough in our modeling approach is to intoduce the modeling of the Databases of the manufacturing applications using UPN by defining the database states as global variables (see 5.2.2), and synthesizing nets through them systematically .

There have been some efforts in synthesis methods for Petri Nets modeling and analysis. [NARA 85] [KROG 86] [JENG 90]. Narahai and Viswanadham [NARA 85] have presented the concept of merging common entities of separate sub-nets and also developed a theorem that will facilitates the computation of invariants of the union of sub-nets in terms of the invariants of the individual sub-net. The result is that the invariants can be computed concurrently with the combining construction of the model. Beck and Krogh [KROG 86] consider the synthesis of common simple elementary paths, that is, they share a common sequence of connected places and transitions. Their results also include the proof of safeness and liveness of the combined nets.

Based on the survey of various synthesis techniques and our modeling of the data entities' states, we started developing a set of rules for the synthesis of separate scenarios modeled with Petri nets. Theories on the properties of the synthesized net deduced from the properties of the sub-nets also need to be derived because of the increased complexity of the large net.

The following sections present first the basic methodology, then the system specification of a sample scenario in our CAD/CAPP/MRP II/SFC integrated system, the UPN model of it, and finally the synthesis procedure.

—— New Meta rule for the synthesis ————

6.3.1 Basic Methodology

The entire company policy is first decomposed into a set of scenarios representing individual operation procedeure, eg. creation, deletion, and modification of work center in our CAD/CAPP/MRP II/SFC integrated system. Each of these scenarios will then be modeled using the top-down refinement methodology presented in section 6.2. This synthesis procedure continues until all the scenarios are connected into one single coherent net representing the entire system. The synthesized net will then be used to reveal conflicts between scenarios and to verify the overall underlying company policy and expert knowledge.

The rule for synthesizing the Petri net submodels of the proposed CIM system are currently under development

The result of each step of the synthesis is a single net with the following notions. Suppose PN is the result of synthesizing PN_1 with PN_2. Then, the resulted $PN = (P, T, I^-, I^+)$ is defined below.

1. $P = P_1 \cup P_2$
2. $T = T_1 \cup T_2$
3. $I^-(P,T) = I^-(P_1,T_1) + I(P_2,T_2)$

Let $UPN_1(P_1, T_1, I_1^-, I_1^+, M_1)$ and $UPN_2(P_2, T_2, I_2^-, I_2^+, M_2)$ be the UPN nets representing two scenarios in the knowledge based system. The synthesized UPN net $UPN(P, T, I^-, I^+, M)$ is defined as follows:

1. $P = P_1 \bigcup P_2$
2. $T = T_1 \bigcup T_2$
3. $I^-(p,t) = I^-(p_1,t_1) + I^-(p_2,t_2)$
   $\forall p \in P, \forall t \in T$
4. $I^+(p,t) = I^+(p_1,t_1) + I^+(p_2,t_2)$
   $\forall p \in P, \forall t \in T$
5. $M(p) = M(p_1) \bigcup M(p_2), \forall p \in P$

6.3.2 System Specification of a Sample Scenario

Two scenarios will be used to demonstrate the synthesis of separate nets, Creation and Deletion of Work Centers in MRP II. The systems specification of creating work center in MRP II is descibed in section 6.2.2, and that of deleting work center in MRP II is described below.

Sample scenario - Deleting Work Centers from the system (Delete): Delete from MRP II

Work centers can only be deleted from MRP II. As explained earlier, MRP II is the execution function in most companies, being in charge of maintaining static data regarding parts and work centers in the system. It is the sole center for purchasing and procurement of resources, and in turn, is the function through which equipment is phased out, or deleted from the system. As in the case of inducting new work centers and other resources into the system, where MRP II users act based on the recommendations of manufacturing and process planning personnel, similarly, while deleting equipment, MRP II users take into account the suggestions of these personnel. The decision is not that of MRP II users alone.

When the delete operation is invoked in MRP II, the following system checks are initiated. A check is made to see that the work center being deleted, exists in MRP II. The status of the work center is not relevant to the operation. In addition all the routings maintained by the MRP II routings module are checked. If any routings utilizing this work center exist, and are in the 'hold' or 'released' status in CAPP, the operation fails, and a message to this effect is displayed. This is because work centers which are utilized by active routings, cannot be deleted. If any manufacturing order in MRP II and SFC utilizing this work center exist, the operation fails, and a message to this effect is again displayed. This is because work centers which are utilized by active orders, cannot be deleted. If the above checks are satisfied, the work center is deleted from the routings module of MRP II, CAPP and SFC.

1. MRP II user starts deleting the Work Center

checks WC ID exists in MRP II
All the routing using this work center are not on h or r status in CAPP
All the manufacturing orders using this work center are not on h or r status in MRP II
All the manufacturing orders using this work center are not on h or r status in SFC

updates WC record removed from MRP II
WC record removed from CAPP DBS if WC exists in CAPP
WC record removed from SFC DBS if WC exists in SFC

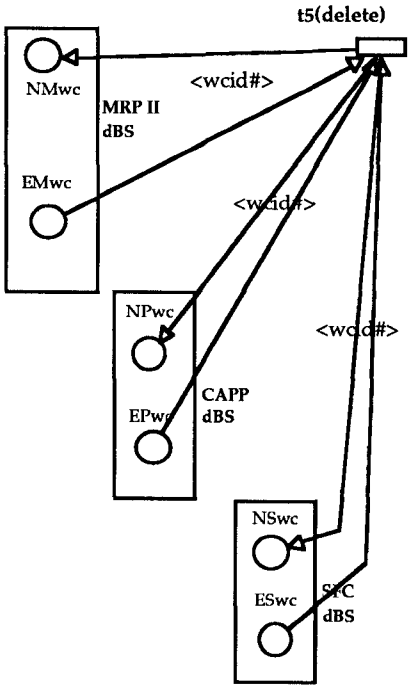


Figure 14: CPN of the Scenario to delete a work center from MRP II at abstract level

6.3.3 UPN model of a Sample Scenario

The two scenarios, Creation and Deletion of Work Centers in MRP II. modeled in UPN is shown in figure 11 and 14. Each of the figures has its incidence matrix ($C = C^+ - C^-$)associated with it.

UPN graph

This example modeled in UPN is shown in figure 11 in the previous section and 14 below.

UPN incidence matrix

The following incidence matrices described the UPN model in figure 14.

$$c^- = \begin{array}{c|c} & t_1 \\ NMwc & 0 \\ EMwc & < wcid\# > \\ NPwc & 0 \\ EPwc & < wcid\# > \\ NSwc & 0 \\ ESwc & < wcid\# > \end{array} \qquad c^+ = \begin{array}{c|c} & t_1 \\ NMwc & < wcid\# > \\ EMwc & 0 \\ NPwc & 0 \\ EPwc & < wcid\# > \\ NSwc & < wcid\# > \\ ESwc & 0 \end{array}$$

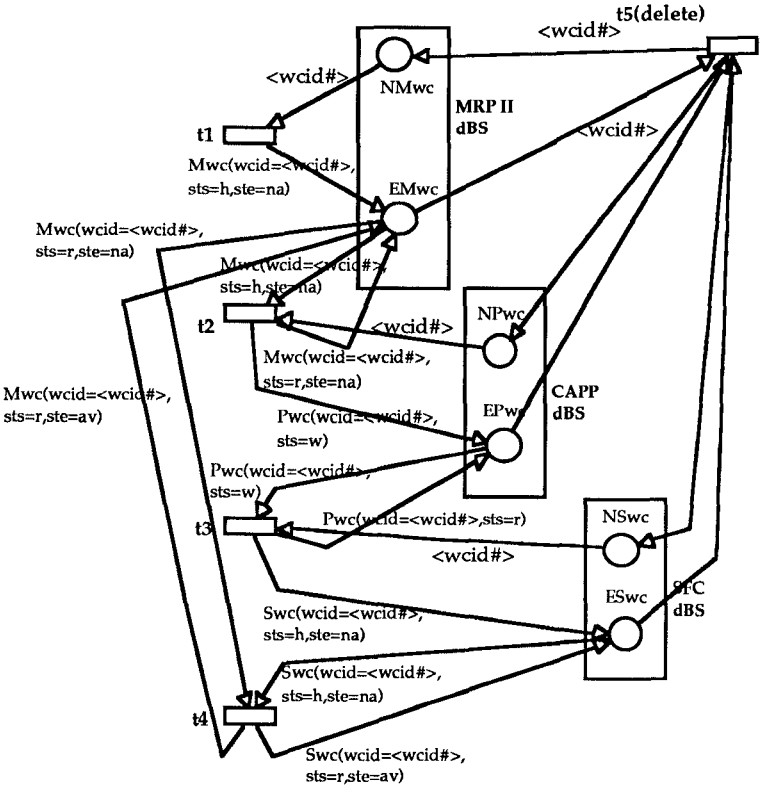


Figure 15: CPN of synthesized Scenario for creation and deletion of a work center from MRP II at abstract level

6.3.4 Procedure for the Formation of Complete Lowest Abstraction Net

6.3.5 Synthesis Formalism for Incidence Matrices

Following the same synthesis technique one by one for all the scenarios , the result is a coherent net representing the behavior of entire system.

In above example, synthesized company policy, which involves the scenario "create a work center from MRP II" and "delete a work center from MRP II", is represented by a single net. This net has the dynamic behavior of both scenarios as well as the relationship between them.

The resulted Petri Net of synthesizing figure 11 and figure 14 is shown in Figure 15. The formalism of creating the resulted incidence matrix is described in appendix E.

7 Conversion of UPN to Generalized Petri Nets

7.1 Introduction

To be able to verify model properties it is useful to have a mathematical representation of the model. By using Colored Petri Net, the representation is mostly symbolic with simple or complicated functions as the elements of the incidence matrix. As has been mentioned before, this complexity is reduced if the formalism for the knowledge base can be described in terms of integers numbers instead of linear functions. For this purpose, a necessary step must be done to convert the high level net of the UPN model into a low level net expressed in terms of Generalized Petri Nets (GPN). We will call this conversion process: "abstraction and unfolding".

As the first step of the conversion process, we will make abstraction of the UPN to reduce the complexity of duplicated net due to the non-fixed data sets. The unfolding consists in taking the information that is aggregated in data structures, data sets and arc functions, and create a new net where this information is splitted in more places and transitions. The arcs of this net have associated integer numbers (but no linear functions) and there exists just tokens.

In order to describe the technical issues for abstracting and unfolding UPN nets, we first briefly introduce the GPN formalism and then we will propose an abstarction and an unfolding procedure to make the conversion.

7.2 Petri net formalism

Formally, a Petri Net can be defined as a 5-tuple $PN \equiv< P, T, I^+, I^-, M_0 >$ in which P and T are disjoint and non-empty sets (whose elements are said *places and transitions* respectively), I^+ e I^- are mappings from $P \times T$ into $\mathcal{N}$ called *pre-incidence and post-incidence* functions, and M_0 is a (initial) marking: $M_0 : P \rightarrow \mathcal{N}$

For each $t \in T$, we define the pre-set and post-set as ${}^\bullet t = \{p \in P \mid I^-(p,t) \neq 0\}$, $t^\bullet = \{p \in P \mid I^+(p,t) \neq 0\}$ (in the same way, for each p we define ${}^\bullet p = \{t \in T \mid I^+(p,t) \neq 0\}, p^\bullet = \{t \in T \mid I^-(p,t) \neq 0\}$).

We say that $t \in T$ is enabled when $\forall p \in P$ it occurs that $M(p) \geq I^-(p,t)$. In other words, in each place of ${}^\bullet t$ there is a number of tokens bigger or equal than the weight of the correspondent arc. An enabled transition can be fired, changing from the present marking M to a new marking M' given by $M'(p) = M(p) + I^+(p,t) - I^-(p,t), \forall p \in P$ (takes tokens from places of ${}^\bullet t$ and puts them in $t^\bullet$ places).

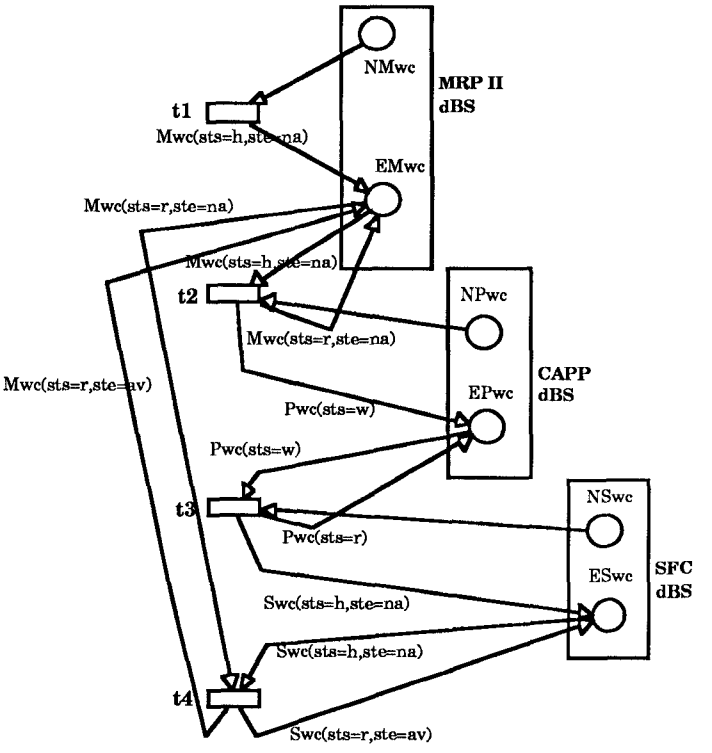


Figure 16: Abstraction of work center identification in the scenario to create a work center via MRP II.

7.3 Abstractions for analysis

Most of the specifications made by the user refer to generic data. In the manufacturing database domain, the user make generic specifications for general work center, part, routing, etc.

A typical example of this can be seen in the net shown in figure 11. The UPN net represents an scenario to create a work center via MRP II at abstract level. All the rules described in that scenario make specifications about a generic work center identification (which is the reference to a work center record). If we consider a particular work center going through the net, the data of its identification will be successively associated to the variables named $< wcid\# >$, and the same will happen with any other work center. That fact means that, in this case, the behavior of the subnet does not depend on a specific work center.

We can reduce the complexity of the net by making an abstraction that considers the behavior for a generic work cell. We will create an abstract net by eliminating all the references to variables of some specific data set. No information is lost in this abstraction process if the variables that are eliminated affect in the same way at all the arcs in the

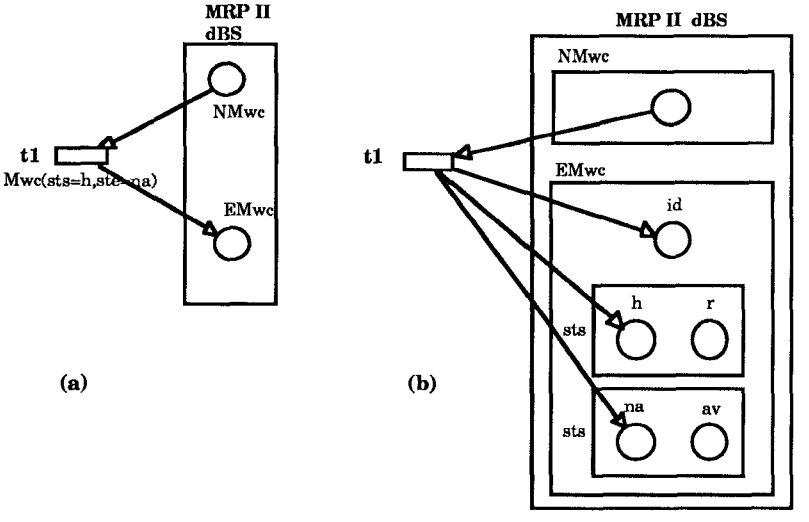


Figure 17: Unfolding subnet with transition $t1$.

subnet been considered. In general, this king of abstraction is very interesting in the analysis of subnets for specific scenarios, but it has bigger implications during the related subnets are considered.

Figure 16 shows the resulting net of abstracting the work cell in the net of the previous example (figure 11). All references to data set $WCID$ has been eliminated.

7.4 Intuitive presentation of the unfolding process

In order to illustrate the unfolding process of a UPN net, let us consider as an example, the unfolding of the abstract net presented in the previous section which is shown in figure 16. To construct the new unfolded net, we will follow an incremental process where places will be created first, then transitions and finally arcs. Let us focus first in the subnet defined by transition $t1$ and places $NMwc$ and $EMwc$ which is shown in figure 17a.

Places $NMwc$ and $EMwc$ represent the nonexistence and existence of the work center record in MRP II database. The unfolding of place $NMwc$ is trivial because it can only contain tokens (it is already an unfolded place). On the other hand, place $EMwc$ contain work center data structures. After abstraction, we can think in a data structure with two fields: Mwc(sts,ste). If we want to differentiate the values of that fields in a net with indifferentiable tokens, we need to create one place for each possible value of that fields which are defined in its data sets:

$$MwcSTS = \{h, r\} \;\; \& \;\; MwcSTE = \{na, av\}$$

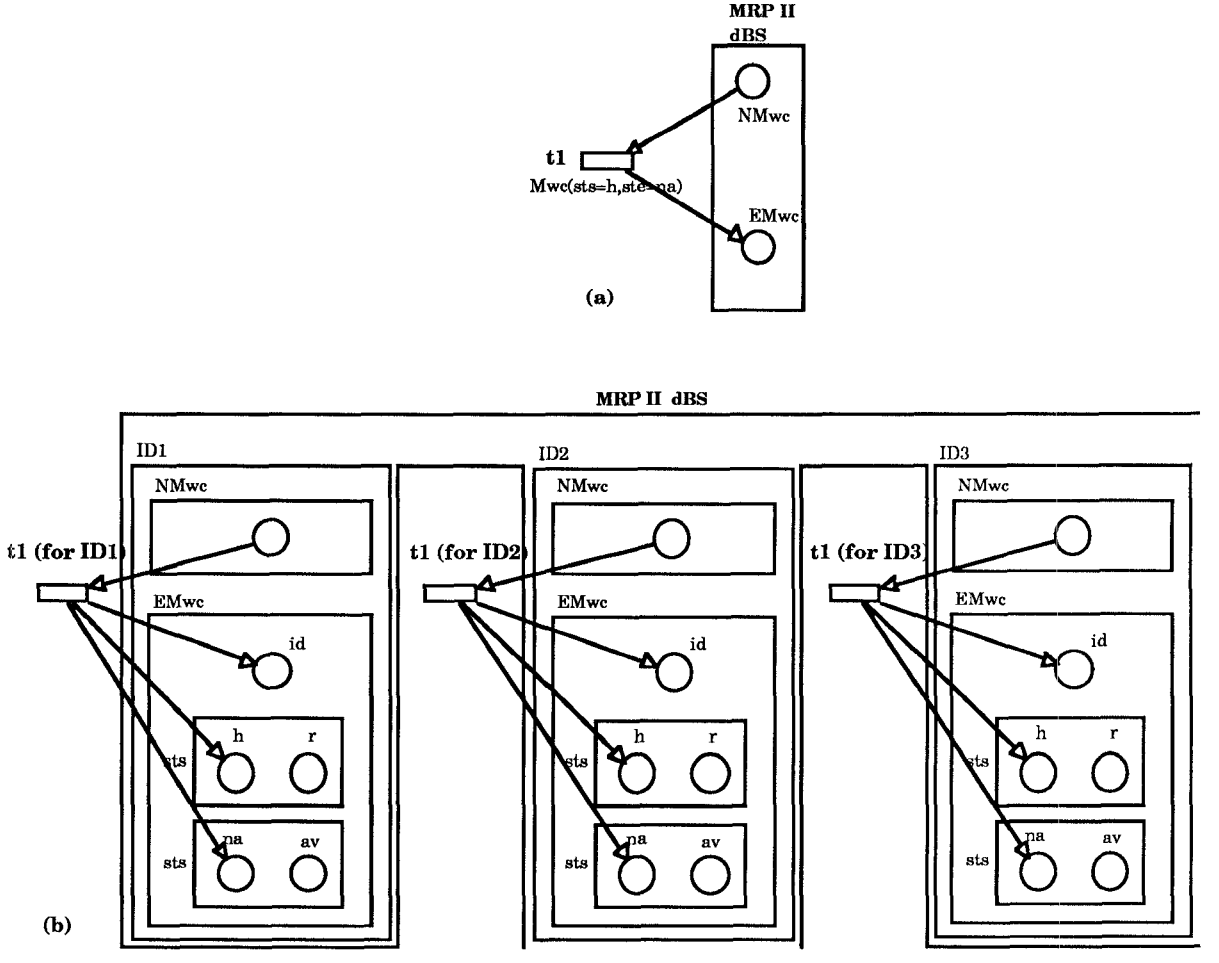


Figure 18: Unfolded subnet previously to abstraction.

Figure 17b shows these unfolded places which have been identified by field and value names.

In this case there is just one unfolded transition because there is just one data for which the transition can be enabled (just one data is associated to the transition). The arcs of the unfolded net describe the arc functions of the original net. In this case arc function $Mwc(sts = h, ste = na)$ connect the transition with the places representing data h from sts and na from ste.

Figure 18 shows the unfolding of rule $t1$ previously to the abstraction of the work center. Let us suppose that there can have three different work center records, $(ID1, ID2, ID3)$. Each work center record can have different field values and need different places (a duplication of the places in figure 17 for each possible work center record). We also need one transition for each possible data in $t1$ (element of the data set associated to $t1$), in this case three. The resulting net can be seen in figure 18b.

Following an analogous unfolding procedure for the rest of places and rules from figure

Figure 19: Unfolded net for the scenario to insert a work center via MRP II.

16 we get the generalized Petri net of the figure 19.

7.5 Automatic procedure for complete net unfolding

Let $N =< P, T, C, Ic^-, Ic^+, In^-, In^+, M_0 >$ be the UPN subnet to be unfolded where at least the non conditional information has been abstracted (that means that $In^-(t,p) = 0, In^+(t,p) = 0, \forall t \in T, \forall p \in P$). For the N net to be unfolded it is necesary step to know all the data sets used in its definition. That assertion implies that the user must specify which are the specific instances of the non fixed data sets (see section 5.2.1). In the previous example the data set for the work center identification was specified to be: $\{ID1, ID2, ID3\}$.

And let $N' =< P', T', C', I^{-\prime}, I^{+\prime}, M_0' >$ be the generalized Petri net resulting from the unfolding of UPN. The components of this new net are created following the next algoritm:

places : Let p be a place from UPN, ($p \in P$), and let $C(p) = C_1 \times C_2 \times \ldots \times C_n$ be its associated data set. Two cases are possible:

1. The associated data set is not a record data structure: One place, denoted $p'_{<c_1,c_2,\ldots,c_n>}$, of the new PN' is created for each possible element $< c_1, c_2, \ldots, c_n > \in C(p)$. We denote by $Uplaces(p)$ the set of places resulting from the unfolding of place p.

2. The associated data set is a record data structure r: The record structure, r, can be decomposed in the following tuple, $r =< c_2, c_3, \ldots, c_n >$, where the record identification field has not been included. One place $p'_{c_{i,j}}$ of the new PN' is created for each element $c_{i,j} \in C_i, \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, size_i\}$, where $size_i$ is the number of elements in C_i. An additional place, id, is added to the previous set of unfolded places which identify the existence of the record in the database.

   We denote by $RUplaces(r)$ the set of places which result from the unfolding of one generic record r. If the net considers a set of different record instances, which are identified by a set of different identification names, $RID =$

$\{id_1, \ldots, id_l\}$, a set $RUplaces(r)$ is needed for each $id \in RID$. We denote the set of places for a particular record identification id as $RUplaces(r)_{id}$.

$$Uplaces(p) = \bigcup_{id \in RID} RUplaces(r)_{id}$$

In general, the new set of places is given by:

$$P' = \bigcup_{p \in P} Uplaces(p)$$

transitions : Let t be a transition from $N, t \in T$, and let $C(t)$ be its associated data set[2]. For each data $< c_1, c_2, \ldots, c_m > \in C(t)$ one transition $t'_{<c_1,c_2,\ldots,c_m>}$ is created in N'. We denote $Utrans(t) = \{t'_c / c \in C(t)\}$ the set of all transitions result of the unfolding of transition t.

In general, the new set of places is given by:

$$T' = \bigcup_{t \in T} Utrans(t)$$

arcs : $I'_-(p', t') = Ic'_-(p, t)$

$I'_+(p, t) = Ic'_+(p, t)$

marking :

[2]In section 5.2.3 was pointed out that $C(t)$ is a function of the data sets associated to the transition variables:

$$C(t) = \{(d_1, d_2, \cdots, d_n) \in D_1 \times D_2 \times \cdots \times D_n \mid (PRED(v_1, v_2, \cdots, v_n))(d_1, d_2, \cdots, d_n)\}$$

where $PRED$ is the predicate attached to t and $V(t) = \{v_1 : D_1, v_2 : D_2, \cdots, v_n : D_n\}$ is the set of all variables appearing free in the immediate surrounding arc functions.

8 Knowledge Base Validation

The first main step within the design methodology proposed above is to create a model to describe the specifications that has been provided by the user. However, the major objective of creating a KBS using Petri Nets is to be able to validate the KBS mathematically and systemmatically.

In this section we will speak about the methodology of analyzing the Petri Nets model in order to validate the model that has been created.

8.1 Introduction

Knowledge base validation is the key issue in the KBS building life cycle. It provides the designers an insight view, both structural and functional, of complex KBS before the KBS was implemented and tested. A knowledge base checking system, called CHECK, was developed in [NGUY 87] and implemented in Lockheed Expert System. Through a five-step procedure, the CHECK system can find simple redundant, conflicting, subsumed, circlar rules by comparing the IF conditions and THEN conditions of each rule. However, it is based on a specific rule syntax in order to find the relationship between every two conditions and it can not handles complicated situations, such as the redundancy between one simple rule and one compound rules (sets of rules). Most of the other KBS validation system are based on similar methodology. A review of some work in Knowledge base validation can be found in [LOPE 90].

It is dificult to provide a general definition of rule based information system validation. As has been pointed out in [LOPE 90], knowledge based systems validation does not constitute a well structured and developed field, and most of the existing validation systems are ad-hoc isolated approaches.

In the approach followed here, validation is concerned with a correct structure and with an adequate behaviour of the knowledge based information system.

UPN provides a formalism for the structured representation of user specifications expressed in terms of if/then rules. The definition of UPN is deliberately based in the classical formalism of Colored Petri Nets (CPN) [JENS 81]. It can be easily recognized from the definition presented before that, once all the data are known, UPN nets form a subset of Colored Petri Nets.

Several analysis techniques have already been developed, in the area of CPN, which can be used for the validation of properties in the systems. In general, this techniques consider that a great diversity of linear functions can be associated to the arcs. Unlikely analysis algorithms for Generalized Petri nets (GPN) that use integer matrices, analysis algorithms for CPN need to manipulate matrices composed by linear functions. This fact introduces a high complexity in the development and execution of that algorithms which

has produced just a limited success.

On the other hand, complete analysis of the model is not always very useful, many times it is sufficient to validate properties using an abstraction of the knowledge structure. Let us consider, for example, that we have created a UPN subnet to represent the user specification to "add a workcell record in CAD", all the transitions in the subnet refer to the workcell record as a variable. The subnet represent the procedure for a generic workcell. It is a useful and much simpler method to make an abstraction of that portion of knowledge and validate the properties of that generic subnet.

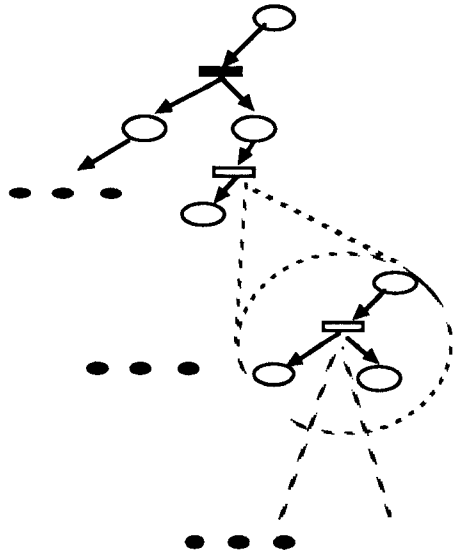


Figure 20: Abstraction schema for the analysis methodology.

The analysis methodology adopted here makes use of this two features to facilitate the analysis process. This is illustrated by figure 20. Starting with the original model that is going to be analyzed, the designer can restrict the portion of knowledge to be analyzed by focusing in a specific subnet. Then the designer can follow a bottom up method by which he selectively make abstractions of the knowledge described in the net. These abstractions avoid the complexity due the manipulation of non relevant details, and focus in more general features.

Perpendicular to this strategy, there exists other direction to facilitate the net manipulation. This is the unfolding of the UPN net, which is expressed in terms of a high level Petri net, into a simpler net, which is expressed in terms of a generalized Petri net (GNP). In this step, no abstraction is made and so, no information is lost in the process. This GNP makes the knowledge structure easier to analyze. A much larger amount of algorithms exists for GPN and, additionally, commercially available software

can be used for that purpose.

Acording to that two previous issues, the following aspects are considered:

Structural verification Structural validation focuses on the correctness of the knowledge base structure, which mainly depends on the KBS representation formalism used. In our case, it is the structure of Petri Nets. With a formal representation of the KBS, it is possible to validate the KBS structure mathematically. The structural validation does not depend on the domain which the KBS runs or the rule specification language used for implementation, thus it is generic for all the KBS using the same representation formalism. The following properties can be tested using structural methods.

[NGUY 87]

Completeness verification the goal is to detect possible gaps in the rule base that have been overlooked in the specification process.

Consistency verification the goal is to perform a static analysis of the logical semantics of the rule structure.

Domain knowledge verification Domain knowledge validation focuses on assuring a proper behavior with respect to the domain problem it is trying to solve. It depends on the functionalities of the company policy and can be very differnt from one application to the other. Therefore, it can not be done fully automatic but needs designer to determine the correctness of the functionalities of the system.

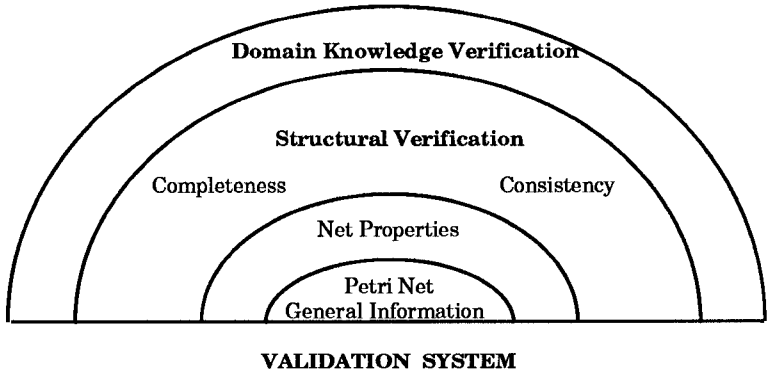


Figure 21: Architecture of the validation system.

Petri Nets has provided mathematical background to do both structural and domain knowledge validation. Generic properties and information of Petri Nets are shown in the following section, which provides the basis of verifying the KBS.

The architecture of our KBS, represented by Petri Nets, validation system, shown in figure 21, consists of three major parts:

1. Generic information and properties of Petri Nets
2. Structural Validation
3. Domain Knowledge Validation

The following sections describe first the generic information and properties of Petri nets, which can be used for validation, second the problems for validation, then the validation methodology, and last example for demonstration.

8.2 Generic Net Information and Properties

8.2.1 Net Information

Information, which is useful for the structural validation of the Petri Nets and can be extracted form Petri Nets [PETE 81] [REIS 85] [MURA 89], are listed as follows.

- Incidence Matrices

  As described in section 6.2.3 an incidence matrix ($C = C^+ - C^-$) is an algebraic representation of one Petri Net with columns representing transitions, rows representing places, and the matrix elements representing the weights of arcs from transitions to places (C^+) and from places to transitions(C^-). It is the state space representation of the Petri Net model, which can be used to reconstruct the Petri Net graph of the system modeled. The matrix elements in GPN are interger values representing the number of connection between places and transitions. For the modeling of information system, the elements of the incidence matrices are either 1, 0, or -1, which represents the initiation or cancellation of facts in the computer system.

  A State equation to transform one marking of Petri Nets into the other is shown below.

  $M_t = M_{t-1} + Cx$

  where M_t and M_{t-1} are the markings at time t and $t-1$. and x is a vector containing the numbers of firing of each transition.

- Reachability trees

  The reachability tree represents the reachability set of a Petri Net. The root node represents the initial marking (or initial state). An arc, representing an enabled

transition t_i under the initial marking, is drawn from the root node to a new node which represents a new marking after firing the transition t_i. Each branch of the tree is then growing until either no transition is firable (*terminal node*). However, the reachability might well be infinite due to an infinite loop in the net and might also produce infinite number of tokens in the net. Therefore, two other situations has to be considered during the construction of the reachability tree in order to prevent these problems.

1. The reachability tree will be terminated if a new node x is the same as an existing node y in the tree (*duplicate node*).
2. The number of tokens in a place of a new node x will be represented by the symbol ω, if the marking of the new node x is a super set of the marking of an existing node y on the path form the root node to the its immediate upper level node and the marking of the place in the new node x is strictly greater than in the existing node y. This special symbol ω represents a number of tokens which can be made arbitrarily large and remains the same through out the rest of the tree.

The reachability tree can provide information of the net, such as deadlock and unwanted final states. However, the complexity of the reachability tree is exponentially increased with respect of the size of the Petri Net. This is still a potential research area in Petri Net theory.

- Invariants

  The S-invariant represents a place vector of weighting factors, which results a constant by vector-product with any marking in the reachability set. If the constant equal to one, the conditions having positive weighting factors in the S-invariants are then mutually exclusive. Solving the S-invariants can be done by multiplying y^T on both sides of the state equation described above,

  $y^T \cdot M_t = y^T \cdot M_{t-1} + y^T \cdot Cx$

  combined with the following equation representing the numbers of tokens between different markings are the same.

  $y^T \cdot M_t = y^T \cdot M_{t-1}$

  The S-invariants are then all the y^T by solving

  $y^T \cdot C = 0$

  The solution is usually infinite and a fast algorithm in solving the minimal set of S-invariants (meaning linearly independent) was proposed in [MART 84].

8.2.2 Net Properties

Utilizing the existing analysis methods of the Petri net theory and the theories of linear algebra, we can validate properties in the knowledge base as described in the following sections. Basic properties of the Petri nets have been discussed in different literature [PETE 81] [REIS 85] [MURA 89] and are listed below. There have been extensive research on how to find these properties and their applications. [JANT 84] [MEMM 84] [POME 87]

- Boundedness

  A Petri net is bounded if the number of tokens in any place in any reachable state is bounded. Boundedness implies that the number of states (reachability set) of the model is finite, and can be applied to manufacturing as finite resources.

- Liveness

  A Petri net is live if the net is deadlock-free and also that from any state reachable by the net, there exist evolutions which can lead to the firing of any transition. This is especially important for an automated manufacturing system, which will leave jobs waiting indefinitely for needed resources if its net is not live.

- Reversibility

  A Petri net is reversible if it is always possible to return the system to its initial state from any reachable state. This type of net enables the user to trace and identify any design errors in the system, and the can then refine the model.

- Mutual Exclusiveness

  A set of places, which can never be marked simultaneously, is composed of mutually exclusive places. This refers to the correct utilization of a shared resource, or the correct modeling of two actually exclusive logical conditions.

The following sections will describe the problems which can be found from Knowledge Based Systems. Based on the logical structure of the Petri Nets and its mathematical background, many of these problems can be detected from generic Petri Nets information and properties, and simple manipulation of the incidence matrices.

8.3 Completeness validation

Completeness validationinvolves the revelation of unreachable results or unfireable rules in the rule base.

- Unreachable facts

- Unfireable rules
- Dead-end conditions

8.3.1 Unreachable facts

A fact, which will never be true through out the life cycle of the whole system, is an unreachable fact. This is a modeling error unless it is an input place, meaning interface of the system.

In Petri Nets, an unreachable fact corresponds to a place which does not have input arcs and is not an input place. This can be easily detecetd from the incidence matrices of the Petri Nets representing the KBS.

8.3.2 Unfireable rules

A rule, which will never fire from any possible state of the system, is an unfirable rule.

In Petri Nets, an unfirable rule corresponds to a transition which will never fire from all the possible initial states. It happens usually because of over constrained preconditions. This can be found by checking the reachability trees of the Petri Nets representing the KBS with all possible initial markings.

8.3.3 Dead-end conditions

A fact, which will never become false once it became true, is a dead-end condition. This is a modeling error unless it is an output place, meaning interface with the user.

In Petri Nets, a dead-end condition corresponds to a place which does not have output arcs and is not an output place. This can easily be detecetd from the incidence matrices of the Petri Nets representing the KBS.

8.4 Consistency validation

One of the most critical issues in the design of a knowledge base is the consistency check, which will eliminate conflicts and redundancies in the knowledge base. Some of the properties related to the consistency of rules is listed and discussed below.

- Redundant rules
- Under-constrained rules
- Subsumed rules

Some simple cases of these problems can be detected by manipulating incidence matrices of the Petri Nets representing the KBS based on similar idea of the CHECK system [NGUY 87]. However, the Petri Nets structurs are validated without the interpretation of the underlying meaning of places and transitions. Because the logic structure of the Petri Nets, some more complicated cases can also be detected, such as the redundancy between one simple rule (primitive transition) and one compound rule (compound trnsition - sub-net).

8.4.1 Redundant rules

Rules are redundant if they succeed from the same preconditions and have the same results. e.g.

$$\begin{cases} If\ \ a\ \ then\ \ b,\ \ and\ \ If\ \ b\ \ then\ \ c \\ bandcarethesame \end{cases}$$

Example of the redundant rules by Petri net is shown in figure 22:

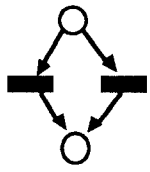


Figure 22: Simple redundant rules.

If two transitions have the same set of input places and they conclude in the same results without connection to any other places, they are then redundant.

If ${}^\bullet t_1 =^\bullet t_2$, and $t_1^\bullet = t_2^\bullet$, then t_1 and t_2 are redundant rules.

These redundant rules can be found by manipulating the incidence matrices for transitions having identical input place set and output place set.

8.4.2 Under-constrained rules

Rules succeed from the same preconditions but their results are not contradictory. e.g.

$$\begin{cases} If\ \ a\ \ then\ \ b,\ \ and\ \ If\ \ a\ \ then\ \ c \\ b\ \ and\ \ c\ \ arenotcontradictory \end{cases}$$

It can be represented by the Petri net graph in figure 23:

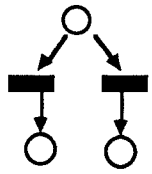


Figure 23: Free choice rules.

Two transition, with the same input places, result non-contradictory output places before more constraints introduced, are choice free in nature. This is not allowed within our current domain knowledge which does not involve any decision making. The decision making is made within the application systems (CAD, CAPP, MRP II, and SFC). If there is some ACTION required from the user, then wee use the **Input** places.

8.4.3 Subsumed rules

Two rules are subsumed if they have the same results but one contains additional constraints than the other. e.g.

$$\begin{cases} If\ \ a\ \ then\ \ c \\ If\ \ a\ \ and\ \ b\ \ then\ \ c \end{cases}$$

Two transitions, having the same output places, and the input places of one transition is a sub set of the input places of the other.
If ${}^\bullet t_1 \in {}^\bullet t_2$, and $t_1^\bullet = t_2^\bullet$, then t_1 is a subsumed rule of t_2.

It can be represented by the Petri net graph in figure 24:

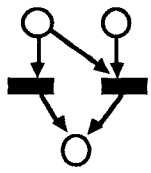


Figure 24: Subsumed Rules.

These subsumed rules can be found by simple incidence matrices manipulations, which can find two transitions having identical ouput place set and input set of one is included in the other's.

8.5 Domain knowledge validation

Assuming the correctness of the rule based information system structure, the system validation amounts to assure that the model of the domain embodied by the rule base contents complies with the semantics of the real world [LOPE 90].

8.5.1 Data Constraints

Each of the data records in the data base, such as work center records, part records, and routing records, could not beat different states at the same time; the states of one member of the same entity (e.g. part) are mutually exclusive.

These mutually exclusive properties can be partially detected by the place invariants (s-invariants) if the total number of the tokens in the invariant is 1. An S-invariants is a set of places in which the total number of tokens remains constant regardless of transition firings. A fast algorithm [MART 82] was adopted to calculate the set of minimal S-invariants, meaning they are linearly independent. From this set of S-invariants, some mutually exclusive conditions can be checked against the user specification of the CIM system.

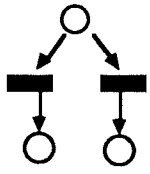


Figure 25: Conflicting rules.

8.5.2 Conflicting rules

Rules are conflicting if they succeed from the same preconditions but have contradictory results. e.g.

$$\begin{cases} If\ a\ then\ b,\ and\ If\ a\ then\ c \\ b\ and\ c\ arecontradictory \end{cases}$$

It can be represented by the Petri net graph in figure 25:

Two transitions, with the same input places, result in a contradictory output (mutually exclusive places) before introducing more constraints, are conflicting in nature.
If 1. ${}^\bullet t_1 =^\bullet t_2$, 2. $\exists p \in t_1^\bullet$, $\exists p' \in t_2^\bullet$ and p and p' are mutually exclusive, then t_1 and t_2 are conflicting rules.

Some of these conflict rules can be found by manipulating the incidence matrices for transitions having identical input place set and output place set and comparing the output places against the S-invariants of the Petri Nets.

8.5.3 If-then Questions

OPEN ISSUES

8.6 Complicated Structural Validation of Petri Nets

The knowledge base validation cases discussed above, which have also been studied by various research projects [NGUY 87] [LOPE 90] using different techniques, only handle simple rules validation but not complex cases as examples described below.

A more complicated case of redundant rules involves one primitive transition and one compond transition, or even two compound transitions. Two sub-nets N_1 and N_2 have the same input places and the same output places. In addition, the set of **Output** places of all the transitions in N_1 is the same as the set of **Output** places of all the transitions in N_2, and there is no **Input** places within both nets. Then these two nets are redundant. (**Input** and **Output**places are defined in section 5.2.2).

Example of the complicated redundant rules by Petri net is shown in figure 26:

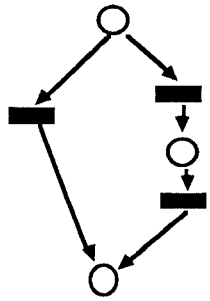


Figure 26: Complicated redundant rules.

If ${}^\bullet N_1 =^\bullet N_2, N_1^\bullet = N_2^\bullet$, and $OP_1 = OP_2$, where $OP_1 = \{p | p \in t^\bullet, \forall t \in N_1, and p^\bullet = \phi\}$

and $OP_2 = \{p|p \in t^\bullet, \forall t \in N_2, and p^\bullet = \phi\}$.

Similarly, a more complicated case of conflict rules involves one primitive transition and one compond transition, or even two compound transitions. Two sub-nets N_1 and N_2 have the same input places and but have contradictory results. e.g.

Example of the complicated conflict rules by Petri net is shown in figure 27:

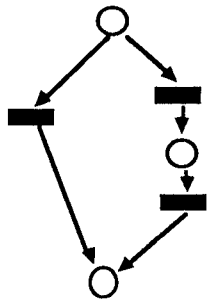


Figure 27: Complicated conflict rules.

By modeling the knowledge base using Petri Nets, we have found that it is possible to validate some complicated rules by applying restrcited reduction rules on the Petri Nets representing the knowledge base. There have been some work done in the Petri Nets reduction methods [LEE 85]. Some reduction methods can be applied on the Petri Nets in order to validate complicated rules within a feasible time frame.

Four net reduction methods which are used and three which can not be used, are shown in figure 28 and 29 in Petri Nets forms for the current knowledge validation procedure in our research.

The procedure to apply these reduction rules on the Petri Nets which representing our Knowledge Base System is described below.

1. Find Place which has only one output transition.

2. The output transition of this place has no other input place except this place.

3. Then this place can be eliminated and it's output transition will be merged into each of it's input transitions. – The ouput places of it's output transition will then be connected to all of it's input transitions.

Figure for the demonstration.

8.7 Conclusion

The procedure of knowledge base validation involves iterations of applying the analysis methods on the Petri net model, feeding back results to the user, and resolving

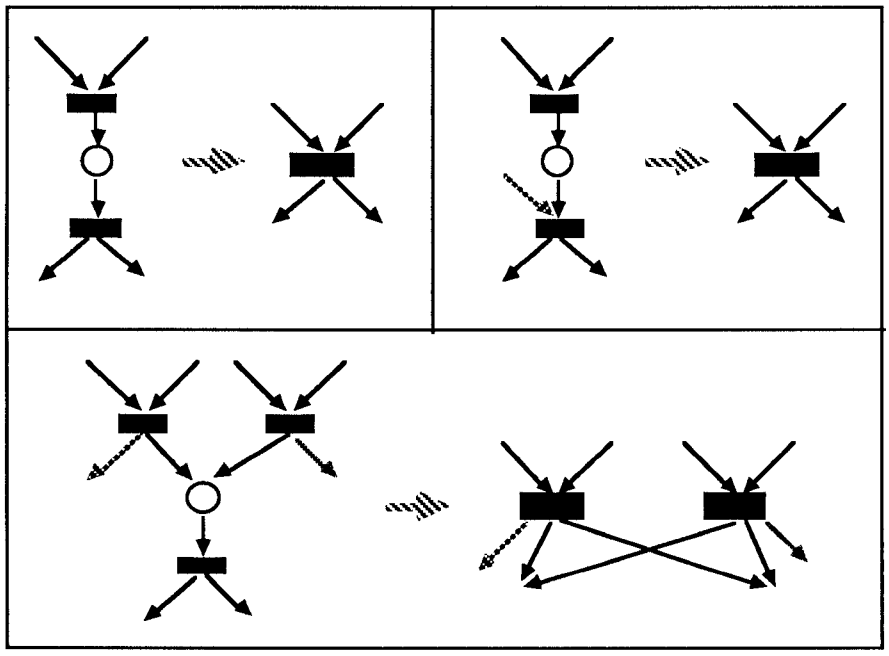


Figure 28: Applicable Petri Nets reduction rules.

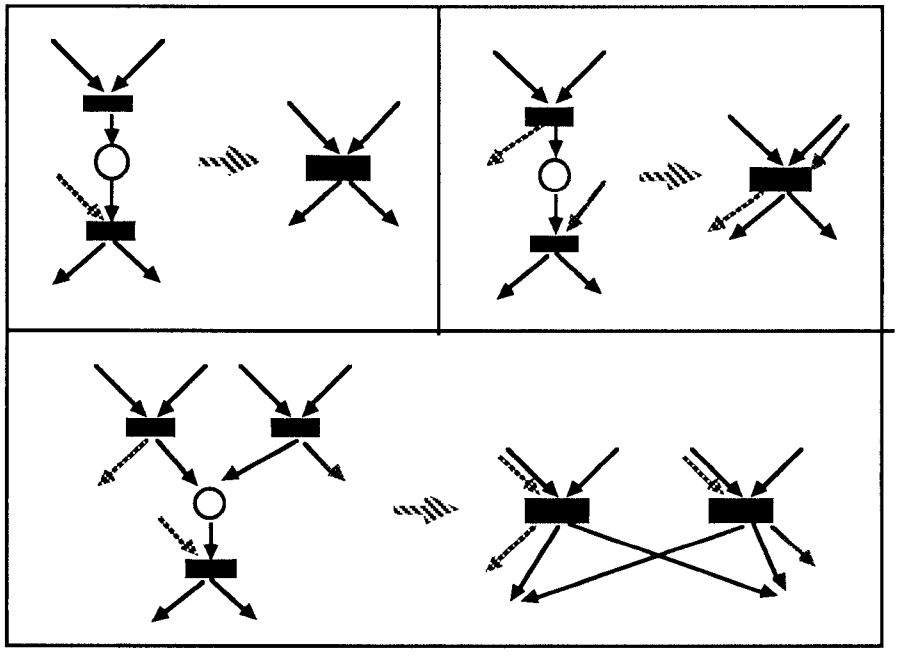


Figure 29: Non-applicable Petri Nets reduction rules.

9 Translation of UPN to Update Dependencies Language

9.1 The Update Dependencies Language

Update Dependencies is a means to specify and control the semantics of a database under update. A set of update dependencies give an operational declarative specification of an update on a relation in terms of a set of alternative sequences of implied updates on the relation and possibly on other relations, and specifies the conditions under which the implied updates must succeed for the original one to succeed. This is clarified by considering the following example.

workcell(R)
⟶ cond1,
    op2(S),
    op3(T),
    op4(R).
⟶ cond2,
    op5(S),
    op6(U),
    op7(R).

Here it is seen that the operation op1 on relation R can succeed only if in one of the two alternative conditions in its update dependency are satisfied, and each of the implied operations of that alternative succeed. This means that either cond1 is satisfied, and op2, op3, and op4 succeed on relations 'S', 'T', and 'R' respectively, or cond2 is satisfied, and op5, op6, and op7 succeed on relations 'S', 'U', and 'R' respectively. If neither of these two eventualities occur, then the operation op1 on relation 'R' fails. The implied operations op2, op3, etc. can in turn be specified either as primitive or compound operations.

The syntax and semantics of the language are formally presented in the appendix F.

9.2 Feature correspondence between UPN and UD

9.2.1 DB Data

The representation of data sets in Updated Dependencies language is by defining its name and its possible values as shown below:

$DataSetName(Value_1, Value_2, ..., Value_n)$

An example the data set of a work center status in MRP II is shown below, which represents the status of a work center in MRP II can have only two different values h for hold, and r for released.

$Sts(r, h)$

The representation of facts in Updated Dependencies language is through relations as shown below, each variable (attribute) corresponding to one data set.

$RelationName(Var_1Name, Var_2Name, ..., Var_nName)$

An example of a work center record in MRP II is shown below, which represents work center *lt101* is a *lathe*, located in the *machining* department, having *hold* status and *not avaiable* state. A common work center record in MRP II is represented as Mwc(Wcid,Sts,Ste,Des,Dep,Cap,Res,Rate,Dh,Esd,Eed,Lp).

$Mwc(lt101, h, na, lathe, machining, \_, \_, \_, \_, \_, \_, \_)$

9.2.2 DB checking

The DB checks corresponds to the facts in UPN, which are represented by the DB record and the values of its variables (attributes). For example: the DB check of work center *lt101* having *hold* status is shown below in UD form

$Mwc(lt101, h, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_)$

and the work center not having *hold* status is shown below in UD form

$Mwc(lt101, h, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_)$

In UPN form, the DB check is represented by a pair of input and output arcs, which have the same arc function, linked between the transition and the DB place.

9.2.3 Input and Output places

The **Input** and **Output** places represents the interface with the users. The **Input** place represents the initiation of an operation by an authorized user and the **Output** places represents message from the system to the user.

9.2.4 Rules

An UD rule will represent a subnet at the lowest level of abstraction. If a path finish in a place internal to the subnet, which means that the subnet can evolve later on, this fact is implemented by making a recursive call to the UD rule (subnet). The parameters to send in the recursive call correspond to the information that is transmitted to the postconditions of the last transition in the path.

Transfer database data related activities represented by transitions will be expressed in terms of UD primitives, such as $add, remove, write, read$.

The variables in the UD rule are defined by the union of all different data sets associated with variables in the transitions aggregated in the UD rule.

9.3 Translation Procedure

The translation to UD will be something like another "implementation" of Petri nets but specific for this application domain. The purpose is to generate efficient code in UD because this is the language in which the specifications will be executed. So we can avoid the restrictions of PN implementation that are not necessary to carry out the user specifications.

For the translation from UPN to UD we will follow what we call an information driven approach. Subnets are splitted in different paths defined by path of transitions that can be fired sequentialy. The enabling of one path is defined by the enabling of the first transition in the sequence. Each path will be implemented as a part of a UD rule. We will specify the enabling of the different paths by distinguishing the information required for the preconditions of the enabling transition.

Provided with the following inputs,

- The UPN
- The corresponding database record with its variables and interpretation

the procedure to translate one sub net into a piece of UD code is detailed as follows:

1. Generate an UD compound operation based on the UPN name and its corresponding database record as: OperationName DBRecord($Var_1, ..., Var_m$).
2. Transform each transition (primitive or compound) into an alternative in UD code.
   - Generate preconditions of each alternative as follows:
     - For input arcs, which carrying values of variables $(iv_1, ..., iv_n)$, generate: nonvar(iv_1) $\wedge$ ... $\wedge$ nonvar(iv_n)
     - Comparing variables between input arcs and ouput arcs to get the additional variables $(av_1, ..., av_n)$ which have not yet been initiated,.and generate: $\wedge$ var(av_1) $\wedge$ ... $\wedge$ var(av_n)
     - For input arc, which carrying information from database records $(RecordName(Value\ of\ Var_1, ..., Value\ of\ Var_m))$, and generate: $\wedge$ $RecordName(Value\ of\ Var_1, ..., Value\ of\ Var_m)$

- Generate operation of each alternative as follows:
  - Request values of additional variables $(av_1, ..., av_n)$ between input arcs and ouput arcs, and generate: ,write('Enter $< av_1 >$'), read(av_1), ..., write('Enter $< av_n >$'), read(av_n)
    where $< av_x >$ corresponding to the interpretation of av_x in the database record tables
  - Output message for each **Output** place by generating: ,write('$< interpretationofthe$**Output** $>$')
  - Remove data from the DB for single input arc from the data base places by generating: ,remove($< inputarcfunction >$) (except from the non-existing data base places)
  - Add data to the DB for single output arc to the data base places by generating: ,add($< outputarcfunction >$)
  - Copy the opeartion name from the interpretation of the transition if it is a compound transition: ,OperationNameX DBRecord($Var_1, ..., Var_{mX}$)
- Restart the operation if the **Input** place belongs to its output set, and generate: ,OperationName DatabaseRecord()
- Generate recursive call if anyone of its output places, which is not a database place, is an input place of any transition within the net, and generate: ,OperationName DatabaseRecord($Var_1, ..., Var_m$)

9.4 Translation Example

The following is an example of translating a sub net (figure 30) from its UPN representation to the respective UD code.

The following matrix described the UPN model in figure 12.

$c^- =$

	t_1	t_2	t_3	Bt_4
$NMwc$	0	0	$< wcid\# >$	0
$EMwc$	0	$(< wcid\# >)$	0	0
p_1	1	0	0	0
p_2	0	$< wcid\# >$	$< wcid\# >$	0
p_3	0	0	0	0
p_4	0	0	0	$< wcid\# >, < des\# >, < dep\# >$

and

$c^+ =$

	t_1	t_2	t_3	Bt_4
$NMwc$	0	0	$< wcid\# >$	0
$EMwc$	0	$(< wcid\# >)$	0	0
p_1	0	1	0	0
p_2	$< wcid\# >$	0	0	0
p_3	0	$< wcid\# >$	0	0
p_4	0	0	$< wcid\# >, < des\# >, < dep\# >$	0

The name of the UPN is 'insert Mwc' and the corresponding data base record - work center record in MRP II - is descrobed below.

Fields in the work center record in MRP II are:

Wcid:	identification number (no fixed)	
Sts:	work center status code	
	h (hold), r (release)	$STS = \{h, r\}$
Ste:	work center state	
	na (not available), av (available)	$STE = \{na, av\}$
Des:	description (no fixed)	
Dep:	department (no fixed)	
Cap:	capacity (no fixed)	
Res:	resource code (no fixed)	
Rat:	rate code (no fixed)	
Dh:	dispatch horizon (no fixed)	
Esd:	effective start date (no fixed)	
Eed:	effective end date (no fixed)	
Lp:	wc load profile (no fixed)	

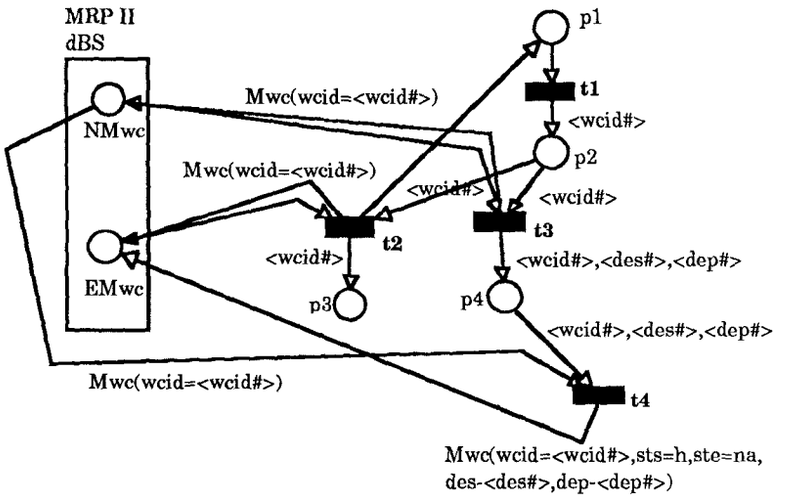
The complete work center record is represented by:

$$Mwc(Wcid, Sts, Ste, Des, Dep, Cap, Res, Rate, Dh, Esd, Eed, Lp)$$

Following the translation preocedure described in previous section, the ouput of the translator is a piece of UD code as shown below, which representing the logic of the desiognated UPN.

```
insert Mwc()                                                    ; Operation Name
⟶ insert Mwc(Wcid,Sts,Ste,Des,Dep,Cap,Res,Rate,Dh,Esd,Eed,Lp)   ; Operation with all its variables


insert Mwc(Wcid,Sts,Ste,Des,Dep,Cap,Res,Rate,Dh,Esd,Eed,Lp)     ; Name of the subnet
⟶ var(Wcid),                                                   ; Transition t1
   write('Enter identification number'),
   read(Wcid),
   insert Mwc(Wcid,Sts,Ste,Des,Dep,Cap,Res,Rate,Dh,Esd,Eed,Lp).  ; Recusive call
⟶ novar(Wcid) ∧ Mwc(Wcid,_,_,_,_,_,_,_,_,_,_,_),               ; Transition t2
   write('Wcid already exists, enter again'),                  ; Output message associated
                                                               to the interpretation of P3
   insert Mwc().                                               ; Restart call
⟶ novar(Wcid) ∧  Mwc(Wcid,_,_,_,_,_,_,_,_,_,_,_),              ; Transition t3
   write('Enter description'),
   read(Des),
   write('Enter department'),
   read(Dep),
   insert Mwc(Wcid,Sts,Ste,Des,Dep,Cap,Res,Rate,Dh,Esd,Eed,Lp).
⟶ novar(Wcid) ∧  Mwc(Wcid,_,_,_,_,_,_,_,_,_,_,_) ∧ novar(Des)   ; Transition t4
   ∧ novar(Dep),      add(Mwc(Wcid,h,na,Des,Dep,null,null,null,null,null,null,null)).
```



t1: user enter the work center number
t2: work center number check fail and rerequest
t3: work center number check succeed and
request more information
t4: B add work center record in MRP II

NMwc: work center does not exist in the database
EMwc: work center exists in the database
p1: I MRP II user starts the insert wc transaction
p2: work center id number is provided
p3: all the necessary dat are provided
p4: O Wcid already exists, enter again

Figure 30: UPN representation of "Insert a work center in MRP II"

References

[ALAN 84] Alanche, P., Benzakour, K, Dolle, F., Gillet, P., Rodrigues, P., and Vallette, R., "PSI: A Petri net based simulator for flexible manufacturing systems", *Advances in Petri Nets 1984, pp. 1-14*, 1984.

[ALLA 84] Alla, H, Ladet, P., Martinez, J., and Silva, M., "Modeling and validation of complex systems by coloured Petri nets: application to a flexible manufacturing system", *Advances in Petri Nets 1984, pp. 15-31*, 1984.

[ANDE 84] Anderson, D.C., Solberg, J.J., and Paul, R.P., "Factories of the future: How will Automation Research be Integrated", *Computer in Mechanical Engineering, vol. 2, no. 4, pp. 31-36*, January 1984.

[APPL 84] Appleton, Daniel S., "The State of CIM", *Datamation, vol. 30, no. 21, pp. 66-72*, December 1984.

[BECK 86] Beck C.L. and Krogh B.H.: "Models for Simulation and Discrete Control of Manufacturing Systems". *Proc. IEEE Int. Conf. on Robotics Automation., pp. 305-310*,1986.

[BONN 87] Bonnevie, A. and Krzesinski, P., "A Double Approach for Analysis and Design of Production Systems", *ESPRIT'86: Results and Achievements, pp. 469-478*, 1987.

[CHRY 87] Chryssolouris, G., "MADEMA: An Approach to Intelligent Manufacturing Systems", *CIM Review, pp. 11-17*, Spriing 1987.

[COUR 83] M. Courvoisier, R. Vallette, et al. "A Programmable Logic Controller Based on A High Level Specification Protocol", *Proceedings of IECON Conference on Industrial Electronics, pp. 174-179*, 1983.

[CROC 86] Crockett, D.H. and Desrochers, A.A., "Manufacturing Workstation Control Using Petri-Nets", *Technical Report 83, Robotics and Automation Laboratory, Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute*, August 1986.

[CROC 87] Crockett, D., Desrochers, A., Dicesare, F, and Ward, T., "Implementation of a Petri net controller for a machine workstation", *Proc. IEEE Int. Conf. on Robotics and Automation., pp. 1861-1867*, 1987.

[DATE 86] Date, C.J., An Introduction to Database System, Addison-Wesley Publication Company, 1986.

[DAVI 88] Davis, W.J. and Jones, A.T., "A Real-time Production Scheduler for a Stochastic Manufacturing Environment", *Int. J. Computer Integrated Manufacturing, vol. 1, no. 2, pp. 101-112*, 1988.

[DRID 85] Dridi, N., Leopoulos, V.I., and Proth, J.M., "Properties of FMS Regarding Optimal Control", *Advanced in Production Management Systems 85, pp. 325-327*, 1985.

[FISH 88] Fishwick P.A.: "The Role of Process Abstraction in Simulation". *IEEE Transactions on Systems, Man and Cybernetics, Vol. 18, No. 1, pp. 18-39*. January/February. 1988.

[FRAN 87] Franks, R.L., Holtman, J.P., Hsu, L.C., Raymer, L.G., and Snyder, B.E., "Productivity Improvement Systems for Manufacturing", *AT&T Technical Report, vol 66, issue 5*, 1987.

[HARH 88] Harhalakis, G., Mark, L., and Lin, C.P., "A Knowledge-Based Prototype of a Factory-Level CIM System", *Journal of Computer Integrated Manufacturing Systems, Vol. 2, No. 1, pp. 11-20*, September, 1989.

[HARH 90] Harhalakis, G., Lin, C., Hillion, H., and Moy, K., "Development of a Factory Level CIM Model", *Journal of Manufacturing Systems, vol. 9, no. 2, pp. 116-128*,1990.

[HSU 87] Hsu, C., Angulo, C., Perry, A., and Rattner, L., "A Design Method for Manufacturing Information Management", *Proceedings of Conference on Data and Knowledge Systems for Manufacturing and Engineering, Hartford, Connecticut, pp. 93-102*,1987.

[JABL 88] Jablonski, S., Ruf, T., and Wedekind, H., "Implementation of a distributed Data Management System for Manufacturing Applications", *Proceedings of International Conference on Computer Integrated Manufacturing, Troy, pp. 19-25*, 1988.

[JANT 84] Jantzen, M., and Valk, R., "Formal Properties of Place/Transition Nets", *Net Theory and Applications, pp. 165-212*, 1984.

[JENG 90] Jeng, M. D. and DiCesare F., "A Review of Synthesis Techniques for Petri Nets", *Proceedings of IEEE Computer Integrated Manufacturing Systems Conference, RPI*, May 1990.

[JENS 81] Jensen K.: "Colored Petri nets and the invariant method". *Theoretical Computer Science, vol. 14, pp. 317-336*, 1981.

[JENS 86] Jensen, K., "Computer Tools for Construction, Modification and Analysis of Petri Nets", *Advances in Petri Nets, Part II, pp. 4-19*, 1986.

[JONE 86] Jones, A.T. and Mclean, C., "A Proposed Hierarchical Control Model for Aotomated Manufacturing Systems", *Journal of Manufacturing Systems, vol. 5, no. 1, pp. 15-25*, 1986.

[JONE 89] Jones, A., Barkmeyer, E., and Davis, W., "Issues in the design and implementation of a system architecture for computer integrated Manufacturing", *Int. J. of Computer Integrated Manufacturing, vol. 2, no. 2, pp 65-76*, 1989.

[KAMA 86] Kamath, M. and Viswanadham, N., "Applications of Petri net based models in the modeling and analysis of flexible manufacturing systems". *Proc. IEEE Int. Conf. on Robotics and Automation., pp. 312-316*, 1986.

[KROG 86] Krogh, B.H. and Beck, C.L., "Synthesis of place/transition nets for simulation and control of manufacturing systems", *Proceedings of 4th IFAC/IFORS Symposium Large Scale Systems*, August 1986.

[MURA 89] Murata, T., "Petri Nets: Properties, Analysis and Applications", *Proceedings of The IEEE, vol. 77, no. 4, pp. 541-580*, April 1989.

[LEE 85] Lee, K.H. and Favrel, J., "Hierarchical Reduction Method for Analysis and Decomposition of Petri Nets", *IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, no. 2*, 1985.

[LOPE 90] Lopez B., Meseguer P. and Plaza E.: "Knowledge Based Systems Validation: A State of the Art". *AI Communications, Vol.3, No. 2, pp. 58-72*. June, 1990.

[LU 86] Lu, S.C.Y., "Knowledge-Based Expert System: A New Horizon of Manufacturing Automation", *Proceedings of Knowledge-Based Expert Systems for Manufacturing in the Winter Annual Meeting of ASME, Anaheim, California, pp. 11-23*,1986.

[MARK 87] Mark, L. and Roussopoulos, N., "Operational Specification of Update Dependencies", *Systems Research Center Technical Reprot No. SRC TR-87-37, University of Maryland*, 1987.

[MART 82] Martinez, J. and Silva, M." A Simple and Fast Algorithm to Obtain All Invariants of A Generalised Petri Net",. *Second European Workshop on Application and Theory of Petri Nets, pp. 301-310*, 1982.

[MELN 85] Melnyk, S.A., Carter, P.L., Dilts, D.M., and Lyth, D.M., SHOP FLOOR CONTROL, American Production and Inventory Control Society, Dow Jones-Irwin, 1985.

[MEMM 84] Memmi, G., and Roucairol, G., "Linear Algebra in Net Theory", *Net Theory and Applications, pp. 213-224*, 1984.

[MERA 87] Merabet, A.A., "Synchronization of Operations in a Flexible Manufacturing Cell: The Petri-Net Approach", *Journal of Manufacturing Systems, vol. 5, no.3, pp. 161-169*, 1987.

[MEYE 87] Meyer, W., "Knowledge-Based Realtime Supervision in CIM - The Workcell Controller", *ESPRIT'86: Results and Achievements, pp. 33-52*, 1987.

[NAGI 88] Nagi, R., "Selection and Layout of Facilities for Cellular Manufacturing", M.S. Thesis, University of Maryland, December 1988.

[NARA 85] Narahari, Y. and Viswanadham, N., "A Petri Net Approach to the Modeling and Analysis of Flexible Manufacturing Systems", *Annala of Operations Research, vol. 3, pp. 381-391*, 1985.

[NGUY 87] Nguyen T.A., Perkins W.A., Laffey T.J. and Pecora D.: "Knowledge Base Validation". *AI Magazine, summer, pp. 67-75*. 1987.

[PAN 89] Pan, Y.C., Tenenbaum J.M., and Glicksman, J., "A Framework for Knowledge-Based Computer-Integrated Manufacturing", *IEEE Transactions on Semiconductor Manufacturing, vol. SM(2)-2*, May 1989.

[PETE 81] Peterson, J.L., "Petri Net Theory and the Modeling of Systems", Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[POME 87] Pomello, L., "Observing Net Behaviour", *Concurrency and Nets, pp. 403-421*, 1987.

[RAVI 87] Ravichandran, R. and Chakravarty, A.K., "Decision Support in Flexible Manufacturing Systems Using Timed Petri-Nets", *Journal of Manufacturing Systems, vol. 5, no.2, pp 89-100*, 1987.

[REIS 85] Reisig, W., "Petri Net", Springer-Verlag, 1985.

[ROSS 85] Ross, D., "Applications and Extensions of SADT", *Computer, vol. 18*, 1985.

[SEPE 87] Sepehri, M., "Integrated Data Base for Computer Integrated Manufacturing", *IEEE Circuits and Devices Magazine, pp. 48-54*, March 1987.

[STAF 83] Staff Report, "ICAM Program Overview", Materials Laboratory, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, Ohio, 1983.

[SU 86] Su, S.Y.W., "Modeling Integrated Manufacturing Data With SAM*", *Computer, vol. 19, no. 1, pp. 34-49*, 1986.

[SUZU 83] Suzuki, I. and Murata, T., "A method for stepwise refinement and abstraction of Petri nets", *Journal of Computer System Science, vol. 27, pp. 51-76*, 1983.

[VALE 79] Valette, R., "Analysis of Petri Nets by Stepwise Refinements", *Journal of Computer and System Sciences 18, pp 35-46*, 1979.

APPENDICES

A Policy Specification Expressed in Natural Language

A.1 Data specification

The basic work center data fields supported by the majority of MRP II, CAPP and SFC systems are as follows. Initially the data fields established by MRP II are described, as the majority of these are supported by all three application systems. Subsequently, the remaining CAPP instantiated data fields and SFC instantiated data fields will be described.

I.D. Number This is a unique identifying number assigned to each work center in the system. MRP II is the sole center for assigning the I.D. Numbers to new work centers, and these numbers identify the work center in CAPP and SFC as well.

Description This field is entered by MRP II users, and contains information as to the type of work center identified by that particular I.D. Number. This record is also maintained in CAPP and SFC.

Department This represents the department to which the work center is assigned. MRP II assigns the work centers to various departments, such as the machine shop, assembly line, quality inspection etc. CAPP and SFC include this field in their records.

Capacity This field represents the number of hours a particular work center is available for use in a given day. This information is established and maintained in MRP II, and transferred to SFC for scheduling purposes.

Resource Code This is a code which is inserted into the work center record, allocating a particular measure to the resource being utilized. The measure could be man hours, machine hours etc. This is done for costing purposes.

Rate Code The Rate Code associates a particular dollar rate per hour, with the specified Resource Code.

Dispatch Horizon : The Dispatch Horizon defines how many working days into the future are to be scanned, so as to provide workload to a work center.

Effectivity Start Date It represents the date on which a particular work center becomes active in the system. This date is determined by MRP II users, based on the introduction of the work center in the system, and the time required to set it up, so that it becomes fully operational. CAPP and SFC users maintain this information as well, in order to determine if the new work center will be ready in time to be utilized by a particular routing. (In fact, upon release of a W.O. to SFC, SFC should check the effectivity start date of work centers involved to make sure they are currently active.

Effectivity End Date This represents the date at which the work center is expected to become inactive in the system, due to obsolescence, or any other reason. This information is not required to be entered when establishing a new work center. It can be entered at a later time.

Status Code Status codes are used as before, to control the state of a work center in the system at any given time. MRP II, CAPP and SFC users can change the status of a particular work center, meaning that they can release, place hold, or delete a work center subject to the rules of the knowledge base.

Work Center Load Profile This field shows the capacity loads on a specific work center scheduled by SFC within the dispatch horizon of both MRP II and SFC.

Work Center State This field shows the physical state, either "operational" or "broken", of a work center used at the shop floor. Both SFC and MRP II use this field to monitor the availability of all the work centers, and resolve problems of machine break down .

In addition to some of the fields described above, CAPP systems normally maintain additional technical performance data in their work center files, which are essential for the generation of process plans, and the selection of work centers for specific operations by CAPP, and are accessable by SFC for routing selection, work order scheduling, and resource allocation. These data fields are as follows.

Horse Power This represents the rated horse power of a particular work center. It is usually given in HP.

Speed Range This field indicates the range of spindle speeds available in the work center. These speeds are represented in RPM.

Feed Range The range of feeds available in a work center are identified by this field. These are in mm./rev or mm./tooth.

Work Envelope This is a measure of the maximum volume of a workpiece that can be accommodated and worked upon by a given work center. This is represented in cm. cube.

Accuracy The rated accuracy of a work center is entered here. It is represented in microns.

Tool Change Time This is the time required to change tools in a particular work center.

Feed Change Time This is the time required to change feed in a given work center.

Speed Change Time This is the time required to change the speed in a particular work center.

Table Rotation Time This is the maximum time required to rotate the machining table, if applicable.

Tool Adjustment Time The time required to adjust the tool to the desired setting in the work center is represented here.

Rapid Traverse Rate This represents the rate of travel of the tool from one position of the workpiece to another, when in the raised position, ie; when not cutting. It is expressed in mm/min.

A.2 Policy specification

Work centers are originated in the system primarily through the MRP II module. MRP II users are responsible for establishing as well as phasing out work centers in the system, and maintaining work center data in MRP II. Because CAPP requires detailed work center information for generating process plans, its work center files incorporate both the work center information maintained in MRP II and other detail technical information. Similarly, SFC also needs detailed work center information as in MRP II and CAPP. Additional

work center information in SFC includes the state of a work center, to be able to schedule operations in the work orders generated by MRP II. Once again there is a great deal of similarity between the sets of data maintained by each application system.

A.2.1 Establishing New Work Centers in the system (Add): ADD via MRP II

1. MRP II user enters the basic data (WC ID, Description, Department) to create the Work Center Record with *h* status in MRP II

   checks WC ID not exist in MRP II

2. MRP II user enters additional information (Capacity, Resource Code, Rate Code, Dispatch Horizon, and Effectivity Start Date) through modify transaction. MRP II user then releases the WC. Otherwise, if the additional information was not entered, the system will prompt for them during releasing of the WC in MRP II.

   checks WC ID exists in MRP II
   WC with *h* status exists in MRP II
   All the necessary data fields are filled
   WC record does not exist in CAPP

   updates WC record status changed from *h* to *r* in MRP II
   Skeletal WC Record automatically created in CAPP with
   w status

3. CAPP user enters additional information (Horse Power, Speed Range, Feed Range, Work Envelope, Accuracy, Tool Change Time, Feed Change Time, Speed Change Time, Table Rotation Time, Tool Adjusting Time, and Rapid Traverse Rate) through modify transaction. CAPP user then releases the WC. Otherwise, if the additional information was not entered, the system will prompt for them during releasing of the WC in CAPP.

   checks WC ID exists in CAPP
   WC with *w* status exists in CAPP
   All the necessary data fields are filled
   WC record does not exist in SFC

   updates WC record status changed from *w* to *r* in CAPP
   WC Record automatically created in SFC with *h* status

4. SFC user releases the WC with the work center state as *av* for being available

   checks WC ID exists
   WC with *r* status exists in MRP II and CAPP
   WC with *h* status exists in SFC

   updates WC record status changed from *h* to *r* in SFC,
   ans state changed from *na* to *av* in SFC and MRP II

A.2.2 Place Hold on a Work Centers via MRP II

1. MRP II user starts the Hold of the WC record

checks WC ID number exists
WC is not on Hold already

updates WC status changed to *h* in MRP II, CAPP and SFC
All Routings using this WC will be put on Hold in MRP II, CAPP, and SFC.
ALL Manufacturing using this WC will be put on Hold in SFC for reselecting routing and rescheduling

Either MRP II or CAPP users can place a hold on a work center, if required. A hold invoked in MRP II could be due to routine preventive maintenance, or serious machine breakdowns. A hold invoked in MRP II will trigger the checks of the existence of the work center ID number and the status of this work center, and will also result in a hold on that workcenter in CAPP and SFC. In this situation, the work center should not be used for the shop floor scheduling, and the shop floor scheduler will then look for alternative routings for those parts utilizing that work center. In addition, it is necessary for CAPP to stop using that work center in developing new process plans because the work center may be broken down for some serious problems and will not be available in a short time. All routings utilizing that work center are as a result automatically placed on hold in MRP II, CAPP, and SFC. All manufacturing orders utilizing that work center are as a result automatically placed on hold in SFC for reselecting routing and rescheduling. A work center can be rereleased at any time in MRP II, resulting in the rerelease of the work center record in CAPP and SFC, and of all the routings which utilized that work center and were placed on hold.

On the other hand, a work center placed on hold in CAPP is the result of complications arising out of manufacturing activities, on the shop floor, and the failure of the workcenter in question to carrying out the requirements of the prescribed operation. In this situation CAPP places the work center on hold, so as to temporarily suspend using that workcenter in its new process plans. In addition since this is a serious situation affecting quality, a hold invoked in CAPP automatically results in that workcenter being placed on hold in MRP II and SFC. As is the case with MRP II, all routings utilizing the workcenter are placed on hold. Therefore until further notice, alternate routings not utilizing the affected workcenter, are to be followed. All manufacturing orders utilizing that work center are as a result automatically placed on hold in SFC for reselecting routing and rescheduling. On review if it is decided to continue using the same workcenter, it can be rereleased. This results in the workcenter getting a released status automatically in MRP II. In addition all the routings utilizing the workcenter, which had been placed on hold, are released.

A.3 Detailed policy specification

MRP II users, have the sole authority to establish new work centers. This is because MRP II is the execution function in most manufacturing organizations, and is used for all types of purchasing and procurement activities. To establish a new work center in the system, MRP II users must provide the following basic information, in addition to assigning the identification number.

- WC I.D.
- Description
- Department

Subject to the condition that this WC I.D. does not already exist in the system, this work center record is established in MRP II with its status set to hold. MRP II users then finalize all the other work center details needed in MRP II module as follows:

- Capacity

- Resource Code
- Rate Code
- Dispatch Horizon
- Effectivity Start Date

Then the MRP II user releases the work center.

This data may be entered separately as each data item becomes known, by using the modify transaction provided in the system model. Otherwise, if the additional information was not entered, the system will prompt for them during releasing of the WC in MRP II as described below. In addition to the data fields mentioned, effectivity end date, status code, work center state, and work center load profile are also part of the work center record in MRP II. The status code is not a user input, but is automatically updated from h for hold to r for released by the release transactions on the work center. The work center state is maintained by SFC users and is not provided at this stage. The work center load profile will be entered and maintained by SFC and updated automatically in MRP II after the work center is allocated for job scheduling.

Invoking the work center release transaction in MRP II triggers a set of consistency checks, which are as follows: the WC I.D. provided must exist in MRP II with hold status; all the required data fields should have been filled, and any data fields left out by users are requested at this stage with the help of system generated prompts. If all these checks are satisfied, the system changes the work center status code from 'hold' to 'released', and a skeletal work center record is automatically created in the work center file in CAPP, with its status set to 'working' as well as a work center record in the work center file in SFC with its status set to 'hold'. These work center records contain all the common information between MRP II and CAPP, and between MRP II and SFC.

CAPP users then input the detailed technical information regarding that work center. The following fields are required to be completed in CAPP, before the work center can be given a released status, and made effective and ready to be used in process plans. If a particular data field does not apply to the specific work center, then 'inapplicable' will be entered automatically. However no field can be left blank.

- Horse Power
- Speed Range
- Feed Range
- Work Envelope
- Accuracy
- Tool Change Time
- Feed Change Time
- Speed Change Time
- Table Rotation Time
- Tool Adjusting Time
- Rapid Traverse Time

Similar to the MRP II, invoking the work center release transaction in CAPP triggers a set of consistency checks, which are as follows: the WC I.D. is checked to ensure that it exists in CAPP a work center file with status set to 'working'; all the required data fields should have been filled, and any data fields left out by users are requested at this stage

with the help of system generated prompts. Upon the satisfaction of these checks, the work center gets a released status in CAPP and the common data fields between CAPP and SFC are automatically copied from CAPP to SFC.

This scenario ends by releasing the work center in the SFC module. Invoking the work center release transaction in SFC triggers a consistency check: the WC I.D. is checked to ensure that it exists in the SFC work center file with status set to 'hold'; the WC status in both MRP II and CAPP are set as r to ensure all the necessary information have been provided; the current date is past the effectivity start date. The work center state is then automatically set to av in SFC and MRP II for being available. Upon the satisfaction of these checks, the work center gets a released status in SFC and the work center state is updated in MRP II.

B Data Representation

Fields in the work center record in MRP II are:

wcid:	identification number (no fixed)	
sts:	work center status code	
	h (hold), r (release)	$STS = \{h, r\}$
ste:	work center state	
	na (not available), av (available)	$STE = \{na, av\}$
des:	description (no fixed)	
dep:	department (no fixed)	
cap:	capacity (no fixed)	
res:	resource code (no fixed)	
rat:	rate code (no fixed)	
dh:	dispatch horizon (no fixed)	
esd:	effective start date (no fixed)	
eed:	effective end date (no fixed)	
lp:	wc load profile (no fixed)	

The complete work center record is represented by:

$$Mwc(wcid, sts, ste, des, dep, cap, res, rat, dh, esd, eed, lp)$$

Fields in the work center record in CAPP are:

wcid:	identification number (no fixed)	
sts:	work center status code	
	w (working), h (hold), r (release)	$STS = \{w, h, r\}$
esd:	effective start date (no fixed)	
eed:	effective end date (no fixed)	
hp:	horse power (no fixed)	
sr:	speed range (no fixed)	
fr:	feed range (no fixed)	
we:	work envelope (no fixed)	
acc:	accuracy (no fixed)	
tct:	tool change time (no fixed)	

fct: feed change time (no fixed)
sct: speed change time (no fixed)
trt: table rotation time (no fixed)
tat: tool adjusting time (no fixed)
rtt: rapid traverse time (no fixed)

The complete work center record is represented by:

$$Pwc(wcid, sts, des, dep, esd, eed, hp, sr, fr, we, acc, tct, fct, sct, trt, tat, rtt)$$

Fields in the work center record in Shop Floor Control (SFC) are:

wcid: identification number (no fixed)
sts: work center status code
h (hold), r (release) $STS = \{h, r\}$
ste: work center state
na (not available), av (available) $STE = \{na, av\}$
des: description (no fixed)
dep: department (no fixed)
esd: effective start date (no fixed)
eed: effective end date (no fixed)
lp: wc load profile (no fixed)

The complete work center record is represented by:

$$Swc(wcid, sts, ste, des, dep, esd, eed, lp)$$

C Deletion of Work Centers

C.1 Delete via MRP II

1. MRP II user starts deleting the Work Center

checks WC ID exists in MRP II
All the routing using this work center are not on h or r status in CAPP
All the manufacturing orders using this work center are not on h or r status in MRP II
All the manufacturing orders using this work center are not on h or r status in SFC

updates WC record removed from MRP II
WC record removed from CAPP DBS if WC exists in CAPP
WC record removed from SFC DBS if WC exists in SFC

Work centers can only be deleted via MRP II. As explained earlier, MRP II is the execution function in most companies, being in charge of maintaining static data regarding parts and work centers in the system. It is the sole center for purchasing and procurement of resources, and in turn, is the function through which equipment is phased out, or deleted from the system. As in the case of inducting new work centers and other resources into

the system, where MRP II users act based on the recommendations of manufacturing and process planning personnel, similarly, while deleting equipment, MRP II users take into account the suggestions of these personnel. The decision is not that of MRP II users alone.

When the delete operation is invoked in MRP II, the following system checks are initiated. A check is made to see that the work center being deleted, exists in MRP II. The status of the work center is not relevant to the operation. In addition all the routings maintained by the MRP II routings module are checked. If any routings utilizing this work center exist, and are in the 'hold' or 'released' status in CAPP, the operation fails, and a message to this effect is displayed. This is because work centers which are utilized by active routings, cannot be deleted. If any manufacturing order in MRP II and SFC utilizing this work center exist, the operation fails, and a message to this effect is again displayed. This is because work centers which are utilized by active orders, cannot be deleted. If the above checks are satisfied, the work center is deleted from the routings module of MRP II, CAPP and SFC.

D UPN Refinement Formalism

1. C is an $n \times m$ matrix representing the abstract net - $PN \equiv< P, T, I^+, I^-, M_0 >$
2. ${}^kC$ $(= {}^kC^+ - {}^kC^-)$ an $n_k \times m_k$ matrix representing the sub net of transition k - ${}^kPN \equiv< {}^kP, {}^kT, {}^kI^+, {}^kI^-, {}^kM_0 >$, in the abstract net.
3. The set of input transitions in ${}^kPN$ is defined as ${}^kT_{input}$ and the set of output transitions in ${}^kPN$ is defined as ${}^kT_{output}$
4. There are n_c common places between these two nets, and are defined as $P_c = \{p | p = p_i = p_{i_k}, and\ p_i \in P, p_{i_k} \in {}^kP\}$
5. ${}^{k'}C^+$ and ${}^{k'}C^-$, being $(n_k - n_c) \times m_k$ matrices,represent ${}^kC^+$ and ${}^kC^-$ respectively by eliminating the rows of P_c.

The refined net can then be represented by ${}^{rf}C$ $(= {}^{rf}C^+ - {}^{rf}C^-)$, and ${}^{rf}C$ is an $(n + n_k - n_c) \times (m + m_k - 1)$ matrix as described below:

- For $i = 1 \sim n, j = 1 \sim (k-1)$
  ${}^{rf}C^-_{ij} = C^-_{ij}$
  ${}^{rf}C^+_{ij} = C^+_{ij}$
- For $i = 1 \sim n, j = k \sim (m-1)$
  ${}^{rf}C^-_{ij} = C^-_{i(j+1)}$
  ${}^{rf}C^+_{ij} = C^+_{i(j+1)}$
- For $i = (n+1) \sim (n + n_k - n_c), j = 1 \sim (m-1)$
  ${}^{rf}C^-_{ij} = 0$
  ${}^{rf}C^+_{ij} = 0$

- For $i = 1 \sim n, j = m \sim (m + m_k - 1)$,
  ${}^{k}t_{j-m+2} \notin {}^{k}T_{input}$ and $p_i \notin P_c$
  ${}^{rf}C^{-}_{ij} = 0$
  ${}^{k}t_{j-m+2} \notin {}^{k}T_{output}$ and $p_i \notin P_c$
  ${}^{rf}C^{+}_{ij} = 0$

- For $i = 1 \sim n, j = m \sim (m + m_k - 1)$,
  ${}^{k}t_{j-m+2} \in {}^{k}T_{input}$ and $p_i \notin P_c$
  ${}^{rf}C^{-}_{ij} = C^{-}_{ik}$
  ${}^{k}t_{j-m+2} \in {}^{k}T_{output}$ and $p_i \notin P_c$
  ${}^{rf}C^{+}_{ij} = C^{+}_{ik}$

- For $i = 1 \sim n, j = m \sim (m + m_k - 1)$,
  ${}^{k}t_{j-m+1} \notin {}^{k}T_{input}$ and $p_i \in P_c$
  ${}^{rf}C^{-}_{ij} = C^{-}_{i_k(j-m+1)}$
  ${}^{k}t_{j-m+1} \notin {}^{k}T_{output}$ and $p_i \in P_c$
  ${}^{rf}C^{+}_{ij} = C^{+}_{i_k(j-m+1)}$

- For $i = 1 \sim n, j = m \sim (m + m_k - 1)$,
  ${}^{k}t_{j-m+1} \in {}^{k}T_{input}$ and $p_i \in P_c$
  ${}^{rf}C^{-}_{ij} = C^{-}_{ik} + C^{-}_{i_k(j-m+1)}$
  ${}^{k}t_{j-m+1} \in {}^{k}T_{output}$ and $p_i \in P_c$
  ${}^{rf}C^{+}_{ij} = C^{+}_{ik} + C^{+}_{i_k(j-m+1)}$

- For $i = (n + 1) \sim (n + n_k - n_c), j = m \sim (m + m_k - 1)$
  ${}^{rf}C^{-}_{ij} = {}^{k'}C^{-}_{(i-n)(j-m+1)}$
  ${}^{rf}C^{+}_{ij} = {}^{k'}C^{+}_{(i-n)(j-m+1)}$

E UPN Synthesis Formalism

1. ${}^{1}C$ is an $n_1 \times m_1$ matrix representing one net - $PN \equiv <\ {}^{1}P,\ {}^{1}T,\ {}^{1}I^{+},\ {}^{1}I^{-}, M_0 >$

2. ${}^{2}C$ $(= {}^{2}C^{+} - {}^{2}C^{-})$ is an $n_2 \times m_2$ matrix representing the sub net of transition k - ${}^{2}PN \equiv <\ {}^{2}P,\ {}^{2}T,\ {}^{2}I^{+},\ {}^{2}I^{-},\ {}^{2}M_0 >$, in the abstract net.

3. There are n_c common places between these two nets, and are defined as $P_c = \{p | p = p_i = p_{i_k}, and\ p_i \in {}^{2}P, p_{i_k} \in {}^{2}P\}$

4. ${}^{2'}C^{+}$ and ${}^{2'}C^{-}$, $(n_2 - n_c) \times m_2$ matrices,represent ${}^{2}C^{+}$ and ${}^{2}C^{-}$ respectively by eliminating the rows of P_c.

The refined net can then be represented by ${}^{sy}C$ $(= {}^{sy}C^{+} - {}^{sy}C^{-})$, and ${}^{sy}C$ is an $(n_1 + n_2 - n_c) \times (m_1 + m_2)$ matrix as described below:

- For $i = 1 \sim n_1, j = 1 \sim m_1$
  ${}^{sy}C_{ij}^{-} = {}^{1}C_{ij}^{-}$
  ${}^{sy}C_{ij}^{+} = {}^{1}C_{ij}^{+}$

- For $i = (n_1 + 1) \sim (n_1 + n_2 - n_c), j = 1 \sim m$
  ${}^{sy}C_{ij}^{-} = 0$
  ${}^{sy}C_{ij}^{+} = 0$

- For $i = 1 \sim n_1, j = (m_1 + 1) \sim (m_1 + m_2)$, and $p_i \notin P_c$
  ${}^{sy}C_{ij}^{-} = 0$
  ${}^{sy}C_{ij}^{+} = 0$

- For $i = 1 \sim n_1, j = (m_1 + 1) \sim (m_1 + m_2)$, and $p_i \in P_c$
  ${}^{sy}C_{ij}^{-} = C_{i_k(j-m_1)}^{-}$
  ${}^{sy}C_{ij}^{+} = C_{i_k(j-m_1)}^{+}$

- For $483i = (n_1 + 1) \sim (n_1 + n_2 - n_c), j = (m_1 + 1) \sim (m_1 + m_2)$
  ${}^{sy}C_{ij}^{-} = {}^{2'}C_{(i-n_1)(j-m_1)}^{-}$
  ${}^{sy}C_{ij}^{+} = {}^{2'}C_{(i-n_1)(j-m_1)}^{+}$

F Update Dependencies Language

The syntax and semantics of the language are formally presented in the following subsections.

F.1 UD syntax

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

<op>
⟶ <c1>,
<op1,1>,
<op1,2>,
<op1,3>,
.
.
.
<op1,n1>.
⟶ <c2>,

<op2,1>,
<op2,2>,
<op2,3>,
.
.
.
<op2,n2>.
⟶ . . .

where <op> is the compound update operation being defined, <opi,j> is either an implied compound update operation of an implied primitive operation, and <ci> is a condition on the database state.

A compound update operation <opi> has the following form:

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

where the <tuple spec> in 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 <opi, 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 <opi,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' ($\uparrow$) 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 can also be used to retrieve data from the system.

F.2 UD 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. It fails otherwise.

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 are 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 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 operation are:

add(r(t)) ; its effect is $r := r \bigcup \{t\}$; 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.