Structured Representation of Rule-Based Specifications in CIM Using Updated Petri Nets

G. Harhalakis, *Member, IEEE*, C. P. Lin, *Member, IEEE*, L. Mark, and P. R. Muro-Medrano

Abstract—**A graphical representation tool—updated Petri nets (UPN)—has been developed to model rule-base specifications for CIM databases. UPN facilitates the modeling of relationships between operations of various manufacturing application systems and the database updates and retrievals among all the respective distributed databases. Based on this representation, a hierarchical modeling technique which includes refining and aggregating rules has also been developed. An application of the UPN is demonstrated in designing rule-based systems for controlling and integrating the information flow between manufacturing applications, including computer aided design, computer aided process planning, manufacturing resources planning, and shop floor control.**

I. Introduction

KNOWLEDGE-BASED systems have become a main stream for controlling the information flow and facilitating the decision process in a variety of industrial applications. A major application of it is manufacturing automation for the control of both data and physical machining processes. Our research emphasis is placed on the control and management of information flow related to production operations to achieve a computer-controlled factory management system. Most of the previous and current research projects emphasize on individual aspects of manufacturing, such as developing a generic CIM architecture, creating a global database framework, or interfacing shop floor activities. We have developed a control mechanism, in the form of a rule-based system, for managing the information flow among manufacturing application systems, and for filling the gap between the high-level production management and the low-level factory operation. A similar approach has been adopted by [1] to develop a framework for integrated CIM databases, using knowledge-based technology. The system architecture of integrated CIM databases involves both the distributed database management systems and knowledge-based systems for sharing information, and the communication between them is achieved through an integrated interface. Its knowledge base consists of several types of knowledge, including domain knowledge, a conceptual data model, a logical structure of the database, and data accessibility. Our approach, however, is emphasizing the knowledge which reflects a company policy; more specifically, the company rules that establish the functional relationships of procedures and operations of various manufacturing application systems.

The emphasis of this paper is to develop a powerful representation tool which can be analyzed in order to validate the underlying domain knowledge extracted from the company policy and can be implemented into rule production systems automatically. The ICAM definition (IDEF) developed by the United States Air Force in their integrated computer-aided manufacturing (ICAM) program [2] provided a primary tool of system design and modeling. However, it can not model adequately precedence constraints or sequences of events occurring in a manufacturing environment, because of its limited modeling structure. The GRAI laboratory of the University of Bordeaux in France has developed the GRAI method [3] especially for the study of decisional production systems. The GRAI method has been developed for analyzing, designing and specifying production management systems in a context of integration. Based on theory of complex systems, hierarchical systems, organization systems, and on the theory of discrete events, it is especially aimed at the study of decisional aspects in manufacturing systems. Therefore, it is not suitable for modeling the dynamic behavior of an information control system.

As a basis, we started with Petri nets for modeling and analyzing rule bases which reflect a company-specific policy and expert knowledge. Petri nets have been proven to be 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 [4], and flexible manufacturing systems [5]. Most recent applications of Petri nets in manufacturing systems are focusing again on the shop floor level, with a large number of workstations, robots, and transportation systems, to be handled by a central controller [6]–[8]. Research in modeling manufacturing systems has been quite extensive in recent years on system validation and performance evaluation using high-level Petri nets, such as timed Petri nets, stochastic timed Petri nets, predicate transition nets, and colored Petri nets [9]–[12]. We chose to start with colored Petri nets (CPN), which allow the model designer to work at different aggregation levels in modeling the flow of information. The main advantage of CPNs over GPNs is the capability of obtaining a compact representation of large and complex systems.

Manuscript received April 10, 1992; revised April 21, 1993.

G. Harhalakis was with the Systems, Research Center, University of Maryland, College Park, MD 20742 USA. He is now deceased.

C. P. Lin is with the Atomic Energy Council, Taiwan.

L. Mark is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA.

P. Muro-Medrano is with the Department of Electrical Engineering and Computer Science, University of Zaragoza, Spain.

IEEE Log Number 9214594.

This paper presents a formal representation of rule-based information systems with a special set of colored Petri nets (CPN) that we developed, which facilitate the modeling of complex systems and can still be analyzed mathematically in order to be validated against the domain knowledge. The next section outlines our model design approach. The third section details the features of updated Petri nets (UPN) with examples from our CAD/CAPP/MRP II/SFC integrated system. The last section presents our conclusions with recommendations for future work.

II. Model Design Approach

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 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 net models can be validated analytically. Readers may refer to [13] and [14] or the fundamentals of Petri net theory. 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 conducted. Worth mentioning are artificial intelligence in network systems [4], flexible manufacturing systems [5], scheduling and sequencing [6]–[8], and information integration for manufacturing applications [15]–[17].

The domain knowledge dealt with in this research is related to company policies for the management and control of information flow in a manufacturing environment. Our work aims at linking product and process design, manufacturing operations and production management; it focuses on the control of information flow 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 company policy basically reflects the procedures and restrictions of how the data should be transferred from one engineering application system of the company to another one; for example, the information regarding a new work center in our view should be created in MRP II system, transferred to CAPP for generating process plans, and landed at SFC for physically installing and operating the work center. Other companies may have a different view about the policy on work center creation; our aim is to present a methodology of policy implementation instead a working knowledge-based system. Similar work on describing the relationship between Petri nets and production rules or logic programs has been done in [18]–[22].

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 project. Our design methodology, as shown in Fig. 1, starts from user-defined rule specifications, reflecting a specific company policy, which is then modeled using UPN in a hierarchical

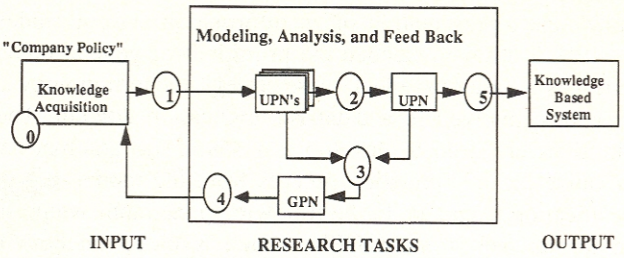


Fig. 1. Knowledge base design methodology.

modeling methodology. We use UPN in modeling not only the rule base, but also the database states in order to ensure the consistency in representing database status. The next step is to convert the UPN model into a set of general Petri nets (GPN) for validation purposes, and to feed the results back to the user to resolve i) conflicting company rules and ii) errors introduced during the modeling phase. After the model has been validated, a parser translates the UPN model into a rule specification language for implementation. The advantage of our methodology is the direct implementation from UPN models to rule-based systems which facilitates the design life cycle and any subsequent necessary updates of the rule bases. A translator has been implemented on a Sun workstation to translate automatically the UPN model to a rule-based system in the [35] Update Dependency Language.

III. Updated Petri Nets (UPN) for Information Systems

High-level nets have been introduced to improve the modeling power of Petri nets for large scale and complex systems. Predicate transition nets [23] have been developed in order to represent complex systems in a much simpler graphical form. Work has been done on modeling logic programs with predicate transition nets [20]. However, the formalism of predicate transition nets involves "token-color" sets which may include an infinite number of colors due to new colors created by transition firings. Therefore, it becomes difficult to interpret the place-invariants (which is one of the most important analysis methods for Petri nets), due to the free variables in those invariants. Colored Petri nets were then proposed by Jensen [24] in order to overcome this drawback, which define explicitly the possible token colors attached to each place and a set of possible occurrence colors to each transition so that only a finite number of token-colors can be created in the model. We have proposed a representation which extends the colored Petri nets formalism in order to utilize both the analytical power of colored Petri nets and the descriptive power of predicate transition nets. More specifically, the "updated Petri nets" (UPN) model, which includes both finite color sets for logic representation and infinite color sets for data entity representation, combines the power of both colored Petri nets and predicate transition nets. Another advantage of UPN is its ability to model data structures; on this issue some earlier work has been done by Silbertin-Blanc [28] and similar work on object-oriented Petri nets has been done by Garnousset [19] which has also included the concept of aggregation of nets

at different levels. This section describes the formalism for knowledge representation of an information system modeled by UPN. We have extended the primitives of general colored Petri nets in order to reflect, more closely, the terminology and semantics involved in a database application domain. These primitives are used to develop a procedure to automate and formalize the interpretation process from the model to a rule specification language. In the following paragraphs we present the formal definition of UPN, which is based on both the CP-graph definition and the CP-matrix definition given by [24].

An UPN is a directed graph with three types of nodes: *places*, which represent facts or predicates; *primitive* transitions, which represent rules or implications; and *compound* transitions, which represent sets of related rules (subnets). Enabling and causal conditions and information flow specifications are represented by arcs connecting places and transitions.

Formally, an UPN is represented as UPN $= \langle P, T, C, I^-, I^+, M_0, I_o, MT \rangle$, which can be decomposed in three different parts:

1) P, T, C, I^-, I^+, M_0 represent the classic color Petri net definition. They identify the part of the information system that provides the conditions for the control of information flow. Only this part of the UPN is used in the validation process. Its entities are defined as follows [24]:
   - $P = \{p_1, \cdots, p_n\}$ denotes the set of places (represented graphically as circles).
   - $T = \{t_1, \cdots, t_m\}$ denotes the set of *primitive* transitions (represented graphically as black bars).
   - $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.
   - C is the color function defined from $P \bigcup T$ into non-empty sets. It attaches to each place a set of possible token-data and to each transition a set of possible data occurrences.
   - I^- and I^+ are negative and positive incidence functions defined on $P \times T$, such that $I^-(p,t), I^+(p,t) \in [C(t)_{\mathrm{MS}} \rightarrow C(p)_{\mathrm{MS}}]_L \ \forall (p,t) \in P \times T$, where S_{MS} denotes the set of all finite multisets over the non-empty set S, $[C(t)_{\mathrm{MS}} \rightarrow C(p)_{\mathrm{MS}}]$ the multiset extension of $[C(t) \rightarrow C(p)_{\mathrm{MS}}]$, and $[\ldots]_L$ denotes a set of linear functions[1].
   - The net has no isolated places or transitions: $\forall p \in P, \exists t \in T : I^-(p,t) \neq 0 \vee I^+(p,t) \neq 0$ and $\forall t \in T, \exists p \in P : I^-(p,t) \neq 0 \vee I^+(p,t) \neq 0$
   - M_0 the initial marking, a function defined on P, such that: $M_0(p) \in C(p), \forall p \in P$.
2) I_o is an inhibitor function defined on $P \times T$, such that: $I_o(p,t) \in [C(t)_{MS} \rightarrow C(p)_{MS}]_L, \ \forall (p,t) \in P \times T$.
3) MT = $\{hm_1, \cdots, hm_l\}$ denotes the set of related transition sets. These are sets of transitions grouped into subnets.

[1] Although, any linear function is allowed in the general color Petri net, only projections, identities and decolorizing functions have been needed so far in our models.

From a rule-based perspective, we have divided the representation of knowledge base components in the following four groups: Data, Facts, Rules, Sets of Rules. In the following sections we define the syntax and semantics of the UPN elements representing those groups. To illustrate our approach, we focus on an example scenario, *creation of a work center via MRP II* and, particularly, on one of its subnets, the *release of a work center in MRP II* as shown in Fig. 2, which is used all along this section. The specification of the example is expressed in a natural language (in the appendix) and represented here using UPN primitives.

A. Data

In information systems, the user needs to refer to atomic data, and establish relations between different data, by structuring information into composed data objects, called database relations or records. In order to a) provide a more adequate representation to facilitate the translation of the user specifications, b) give a more clear graphical representation, and c) make the validation process easier, UPN allow for the specification of the following classes of information (in general we refer to any of these data classes by the generic name "data" or "color"):

Atomic data is individual information with a lexical representation. Each atomic data is represented by an identifier, a color, which is attached to a specific token, whose syntax is an alpha-numerical sequence. As an example, let us suppose that a part in CAD can be in four different statuses: w (working), r (released), 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 color set: DSTS $= \{w, r, h, o\}$.

Atomic data can belong to one of the following two classes: *fixed*, if the color set is completely known in advance, such as the set status which was described above DSTS = $\{w, r, h, o\}$, and *non-fixed*, if not all of the color set components are completely known in advance, such as the information about a part identification number in CAD, which can be an alphanumerical sequence (e.g., $c157635$) or the description of a part record which can be any string (e.g., "Fine-pitch involute spur gear").

This last example raises another interesting point related to non-fixed data. There is some data, such as a part identification number, that is normally used to guide the control of information flow; generally, preconditions of a rule can be related a specific part identification number, but normally, specifications do not care about the specific instances of its description; in other words, the description is associated to a specific part and it is "flowing" with it in the system, but no precondition check for a specific description is necessary. Thus, *non-conditional* data refer to information which is not important for the control of the system and their flow is controlled by *conditional* data.

Data structure is the classical mechanism to aggregate related data. Therefore, instances of data structures can be seen as ordered tuples $\langle s_1, s_2, \ldots, s_n \rangle$. Different tuples

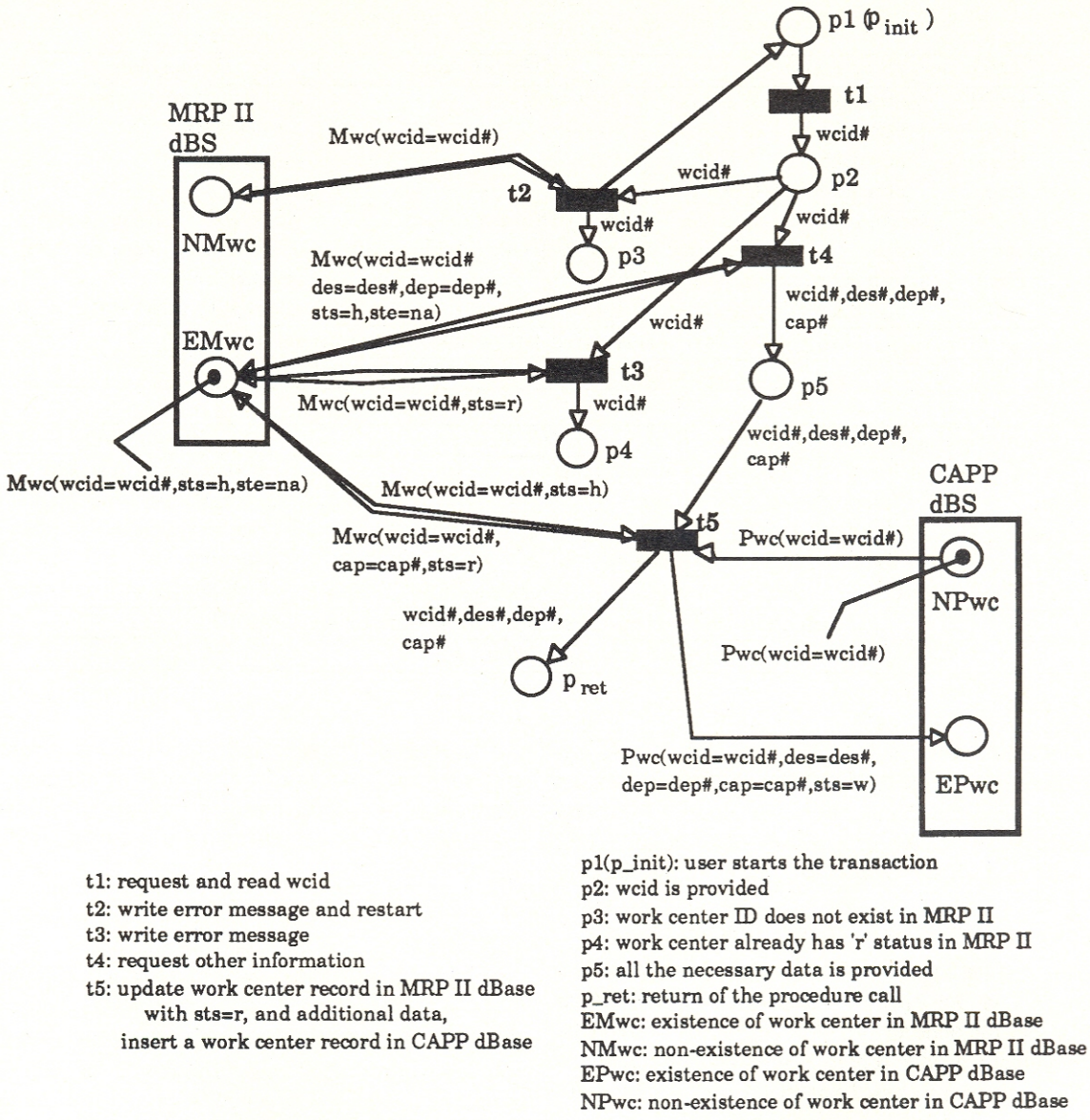


Fig. 2. Subnet of the work center creation scenario: "Release of a work center in MRP II" with initial marking.

can be identified by their relation name, $\langle R \rangle$, and different elements of a tuple can also have textual identifications $A_1, \ldots, A_m$. This means that every tuple is represented as $\langle R \rangle(A_1, \ldots, A_m)$, where $\langle R \rangle$ is the database relation name and $A_1, \ldots, A_m$ are textual identifications for its attributes. A specific element of a tuple can be accessed in two ways: either by referencing its specific position within that tuple ("$-, -, \ldots, v_i, \ldots, -$"), or by referencing its attribute name ("$A_i = v_i$").

To illustrate the model adopted for data representation, let us consider the specifications related to a work center record in MRP II. This record is represented in UPN by a data structure named Mwc (where M identifies the MRP II database and wc identifies the work center record). Its composition, characteristics of its attributes, and its representation are shown in Table I.

B. Facts

Each fact declares a piece of information about some data, or data structure, in the system. 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 subnets for different scenarios which can be linked together later to form an unified model.

Facts in UPN are represented by places. Places are one of three kind of nodes in a UPN and are graphically represented as circles. The fact asserted by one place is determined by the place name and its content. The content defines the marking of the place; we will refer to the marking of place p by $M(p)$. For example, the fact that a token of color A is in place *inprocess*, can be interpreted as: "*A is in process*" or *inprocess*(A) in logic syntax.

TABLE I
DATA STRUCTURES AND COLOR SETS OF WORK CENTERS

Attibute	*Data class*	*Color set*	*DB data type*	*Description*
wcid	non fixed	$WCID$	identification	identification number
des	non fixed	DES	text	description
dep	non fixed	DEP	text	department
cap	non fixed	CAP	integer	capacity
sts	fixed	$MSTS$	$\{h, r\}$ (hold, release)	work center status code
ste	fixed	$MSTE$	$\{na, av\}$ (not avail., avail.)	work center state code
res	non fixed	RES	text	resource code
esd	non fixed	ESD	date	effectivity start date
Complete data structure for work center in MRP II				
$Mwc(wcid, des, dep, cap, sts, ste, res, esd)$				

Attibute	*Data class*	*Color set*	*DB data type*	*Description*
wcid	non fixed	$WCID$	identification	identification number
des	non fixed	DES	text	description
dep	non fixed	DEP	text	department
cap	non fixed	CAP	integer	capacity
sts	fixed	$PSTS$	$\{w, h, r\}$ (working, hold, release)	work center status code
Complete data structure for work center in CAPP				
$Pwc(wcid, des, dep, cap, sts)$				

Tokens in places *NMwc* and *NPwc* from Fig. 2 represent the non existence of a record in MRP II and CAPP databases respectively. On the other hand, tokens in places *EMwc* and *EPwc* represent the existence of a record. The interpretation associated with each of the rest of the places is provided in the figure.

Regarding facts, places can be of two different types:

Local scope: when a place is used to represent a local fact (relevant only to one specific scenario or subnet) which will not affect the state of the other scenarios or subnets. They are typically used to represent intermediate state facts within the decision process. Examples of local places are p_1, p_2, p_3, p_4, p_5 in Fig. 2.

Global scope: when a place is used to represent a fact relevant to (accessible by) different scenarios or subnets. Global places must be referred to by the same name in all of their occurrences. Typical examples of global places are facts about database state information, such as places $EMwc$ an $NMwc$ (in the MRP II database) and $EPwc$ and $NPwc$ (in the CAPP database) shown in Fig. 2.

Regarding the user interface, there are two special types of places to enable the exchange of information with the user (these places belong to local scope):

Input places represent facts whose initialization is generated by an external action (i.e. by a request from the user through the interface). An example of an input place is place p_1 shown in Fig. 2, which needs the introduction of a token for the release process to start.

Output places are used to represent situations where some information must be passed to the user (i.e., some information to be displayed). An example of an output place is place p_4 shown in Fig. 2, the value provided in *wcid#* will be displayed.

The concept of *input* and *output* places is similar to that of *source* and *sink* places [26] and, in addition, provides a mechanism to specify terminal facts during consistency and completeness verifications of the knowledge model.

C. Rules

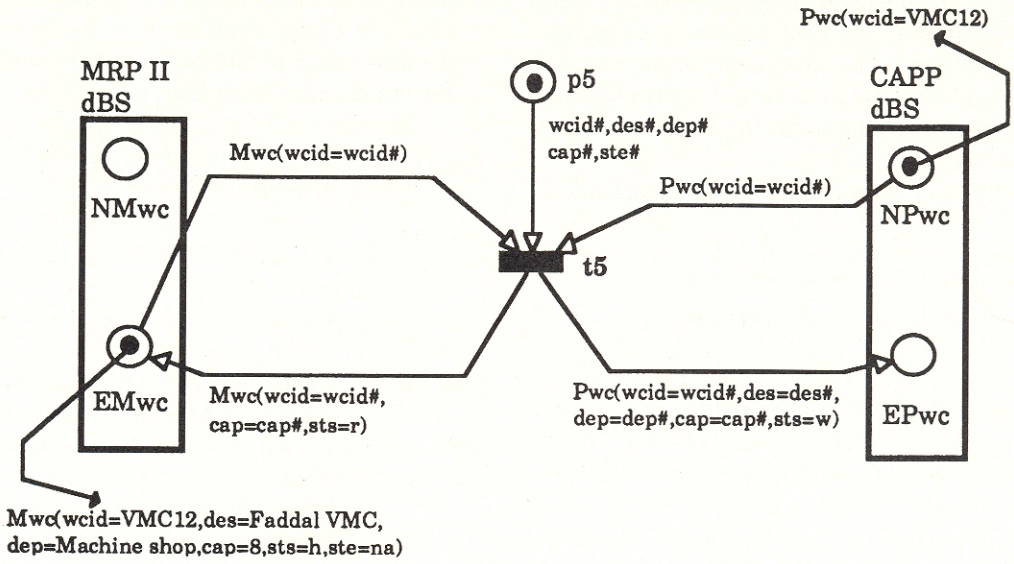
Another important feature in modeling information systems is the representation of information flow. Here, we are considering domains where the user specifies information flow policies using "if then" rules. These rules are expressed in UPN by means of *transitions*. Any transition t has a color set, $C(t)$, associated with it. The net in Fig. 2 shows five transitions (t_1, t_2, t_3, t_4, t_5) and their associated interpretations.

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

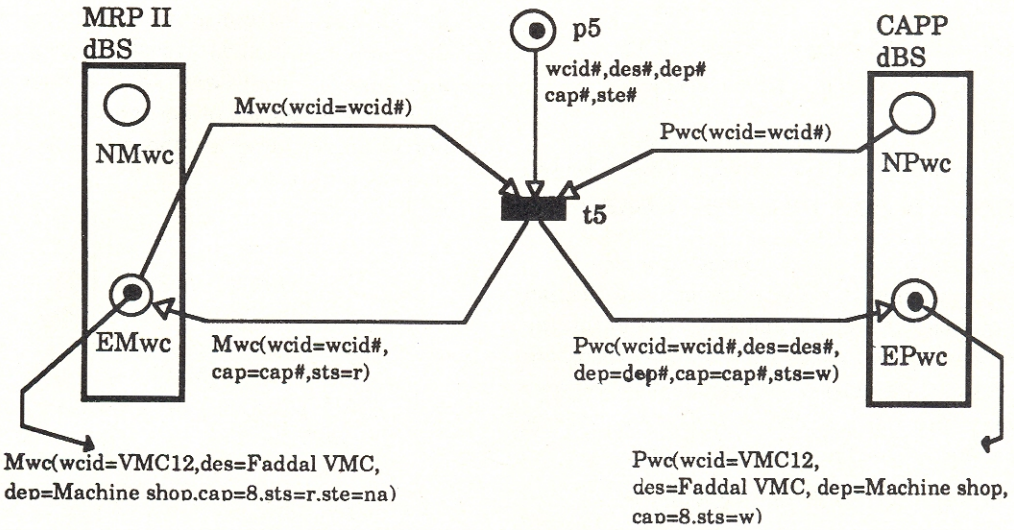
Variables: UPN characterize references to generic data-using variables (in the sense that a relation can hold for different data occurrences). There exists a set V of typed variables. Each variable $v : D$ has a name v and an associated color set D. They are represented graphically by names ending with the symbol "#". Examples of variables are *wcid#*, *des#*, *dep#*, and *cap#* as shown in Fig. 2.

Arcs: Arcs are relations that connect facts and actions to form rules, and constitute the rule preconditions and postconditions. Arcs also identify the flow of information. Formally, each arc has attached to it an arc expression, *exp*, 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 color set of the variable or constant v_i.

The lambda-expression, $\lambda(v_1, v_2, \cdots, v_n).exp$ is defined as a mapping from $D_1 \times D_2 \times \cdots \times D_n$ into $C(p)$, where the place p is source/destination of our arc.



(a)



(b)

Fig. 3. UPN representation of the rule "Release work center from MRP II."

UPN use variables as in typical rule-based systems, where they are used to specify data sets within the enabling arcs of a rule and as a mechanism to transfer data from the preconditions to the postconditions. For example, looking at the arcs between transition t_4 and place p_5 in Fig. 2, the information about variables *des#* and *dep#* is needed to go to place p_5, and is tranferred from place $EMwc$, as shown in the arc expression.

In general colored Petri nets, upon the firing of a transition, the enabled colors are taken out from the incoming places. We have introduced a special case in UPN, in situations where a place represents a database related fact and the color of its token represents a record. In case of firing, all the information of that record is removed from the place, while only a subset of its attributes may be involved for the required updates. The information about these unspecified attributes must also be included in the color set of the destination transition to transfer those information from one database to the other or back to itself. For example, looking at transition t_5 in Fig. 2, the attributes *des*, *dep*, and *ste* of the MRP II work center table are not shown in the arc expressions for the update of the work center record (pair of arrows from and to the place *EMwc*). However, these attributes have to be included in the colored set of transition t_5 as shown in Table I. One of these variables is used for each record attribute, which is not specified in the expressions of arcs connected to that transition. They are identified by names ending with the symbol "0".

Each transition can have an associated predicate, *pred*, to impose additional constraints for the enabling of the transition. This predicate is a boolean expression which can only contain

those variables which are already in the expressions of all arcs connected to that transition. To avoid degenerate transitions, the predicate must differ from the constant predicate *false*. The predicate is supposed to be *true* by default. Simple predicates can be grouped by means of the following logical operators: $\neg$(*not*), $\vee$(*or*) and $\wedge$(*and*).

The color set, $C(t)$, of a transition, t, $\forall t \in T$, is determined by [24]:

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

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 in the expressions of all arcs connected to the transition.

As an example, let us look at Fig. 3(a), which focuses on transition t_5 of the net shown in Fig. 2. The color sets for the associated places are as follows (the simple color sets are specified in Table I):

$$\begin{aligned} C(EMwc) &= \text{MWC} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP} \\ &\quad \times \text{MSTS} \times \text{MSTE} \times \text{RES} \times \text{ESD} \\ C(NPwc) &= \text{WCID} \\ C(EPwc) &= \text{PWC} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP} \times \text{PSTS} \\ C(p_5) &= \text{WDDC} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP} \end{aligned}$$

There are four variables in the arcs connected to t_5 (*wcid#, des#, dep#* and *cap#*). On the other hand, t_5 has an incoming arc from place *EMwc*, a database place for the record *Mwc (wcid,des,dep,cap,sts,ste,res,esd)*, and there are five attributes which do not need to be specified in the arcs to/from *EMwc: des, dep, ste, res*, and *esd*. The total set of variables and the color set for the transition t_5 are

$$\begin{aligned} V(t_5) &= (wcid\# : \text{WCID}, des\# : \text{DES}, dep\# : \\ &\quad \text{DEP}, cap\# : \text{CAP}, mste\# : \text{MSTE}, \\ &\quad res\# : \text{RES}, esd\# : \text{ESD}) \\ C(t_5) &= \text{MWCS} = \text{WCID} \times \text{DES} \times \text{DEP} \\ &\quad \times \text{CAP} \times \text{MSTE} \times \text{RES} \times ESD. \end{aligned}$$

Functions in I^- and I^+ are defined in terms of lambda expressions with the form $f(c) = \lambda(V)\exp(c)$, where $c \in C(t)$ and exp is the expression associated to the arc. For transition t_5, $V \in$ MWCS can be represented as follows:

$$V = (wcid\#, des\#, dep\#, cap\#, mste\#, res\#, esd\#)$$

UPN provides different types of arcs:

Enabling arcs are directed arcs which connect a place/fact with a transition/action and define a precondition for the transition. They indicate the data that must be in a place for a transition to be enabled and must be removed from that place on firing. Given $C(t)$ and $V(t)$ defined as above for some transition $t \in T$, if a single arc from a given $p \in P$ exists, with arc expression exp, $I^-(p,t)$ is then defined to be the function $\lambda(v_1, v_2, \cdots, v_n).\exp$ (mapping between $C(t)$ and $C(p)$). If no arc from p to t exists, $I^-(p,t)$ is the zero-function. For example, these are the enabling arcs for transition t_5 from Fig. 3(a):

- $I^-(p_5, t_5) : \exp = \begin{bmatrix} wcid\# \\ des\# \\ dep\# \\ cap\# \end{bmatrix}$,
  $\lambda(V).\exp \in [\text{MWCS}_{\text{MS}} \rightarrow \text{WDDC}_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = (wcid\#, des\#, dep\#, cap\#)$
- $I^-(EMwc, t_5) : \exp = [wcid = wcid\#]$,
  $\lambda(V).\exp \in [\text{MWCS}_{\text{MS}} \rightarrow \text{MWC}_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = EMwc(wcid = wcid\#)$
- $I^-(NPwc, t_5) : \exp = [wcid\#]$,
  $\lambda(V).\exp \in [\text{MWCS}_{MS} \rightarrow \text{WCID}_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = (wcid\#)$

Causal arcs are directed arcs which connect a transition with a place and define a postcondition for the rule. Causal arcs describe modifications to be performed in the state of the net when the transition is fired; and more concretely, they indicate which data must be added to a place after the transition has been fired. I^+ is defined in the same way as I^-, but by means of the arcs from transitions to places. For example, these are the causal arcs for transition t_5 from Fig. 3(a):

- $I^+(EMwc, t_5) : \exp = \begin{bmatrix} wcid & = wcid\# \\ des & = des\# \\ dep & = dep\# \\ cap & = cap\# \\ sts & = r \end{bmatrix}$,
  $\lambda(V).\exp \in [\text{MWCS}_{\text{MS}} \rightarrow \text{MWC}_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = EMwc(wcid = wcid\#, cap = cap\#, sts = r)$
- $I^+(EPwc, t_5) : \exp = \begin{bmatrix} wcid & = wcid\# \\ des & = des\# \\ dep & = dep\# \\ sts & = w \end{bmatrix}$,
  $\lambda(V).\exp \in [\text{MWCS}_{\text{MS}} \rightarrow PWC_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = EPwc(wcid = wcid\#, des = des\#, dep = dep\#, cap = cap\#, sts = w)$

Checking arcs indicate which data must be present in a place, in order to enable a transition, but no data are removed upon firing. They are represented by an enabling and a causal arc, with identical arc expressions. These are generally used to indicate database checkings. For example, transition t_2 and place *EMwc* in Fig. 2 are connected by a checking arc with the associated arc expression: *Mwc(wcid = wcid#)*. This is a precondition for t_2 to be enabled, but after the transition has been fired, no data are removed from or added to *EMwc*. Arcs between t_2 and *NMwc* and arcs between t_4 and *EMwc* are also checking arcs.

Inhibitor arcs indicate which data must not be present in a place in order to enable a transition. It is graphically represented as an enabling arc ending in a small circle.

It is also possible for a transition to request information from the user interface. This is done whenever an arc expression requires some information that has not been provided by the variables in the enabling or checking arcs.

Transition Enabling and Firing: The dynamic behavior of the net is provided by transition enabling and firing. A transition $t \in T$ is said to be enabled with respect to a color $c \in C(t)$ if the current marking M is such that

$$M(p) \geq I^-(p,t)(c) \quad \text{and } M(p) < I_o(p,t)(c), \forall p \in P.$$

An enabled transition may be chosen to fire. The firing of a transition t with respect to a color c consists of removing $I^-(p,t)(c)$ colors from each of its input places and adding $I^+(p,t)(c)$ colors to each of its output places. Each firing creates a new set of conditions, and the total number of colors in the net may change after each firing.

In order to illustrate the previous concepts, we focus on transition t_5 in Fig. 3(a). For this explanation, let us consider the following initial marking:

$$\begin{aligned}
M_0(NMwc) &= \emptyset \\
M_0(EMwc) &= (vmc12, \textit{"Fadal VMC"}, \textit{"machine shop"}, \\
&\qquad \textit{null, w, na, m12, null}) \\
&= Mwc(wcid = vmc12, des = \textit{"Fadal VMC"}, \\
&\qquad dep = \textit{"machine shop"}, cap = null, \\
&\qquad sts = w, ste = na, res = m12, esd = null) \\
M_0(NPwc) &= vmc12 \\
M_0(EPwc) &= \emptyset \\
M_0(p_5) &= (vmc12, \textit{"Fadal VMC"}, \textit{"machine shop"}, 8)
\end{aligned}$$

It is clear that transition t_5 is enabled with respect to the following specific color (note that the entire color set for transition t_5 is $C(t_5)$ = MWCS):

$$c = (vmc12, \textit{"Fadal VMC"}, \textit{"machine shop"}, \textit{null, na, m12, null})$$

Because c satisfies the following enabling rules, t_5 is then enabled.

$$\begin{aligned}
M_0(NMwc) &\geq \lambda(V)\exp(c) = \emptyset \\
M_0(EMwc) &\geq \lambda(V)\exp(c) = vmc12, \textit{"Fadal VMC"}, \\
&\qquad \textit{"machine shop", null, w na, m12, null} \\
M_0(NPwc) &\geq \lambda(V)\exp(c) = vmc12 \\
M_0(EPwc) &\geq \lambda(V)\exp(c) = \emptyset \\
M_0(p_5) &\geq \lambda(V)\exp(c) = (vmc12, \textit{"Fadal VMC"}, \\
&\qquad \textit{"machine shop"}, 8)
\end{aligned}$$

where exp represents the corresponding arc expression, and V is as defined in Section III-C

The firing of transition t_5, with respect to color c, implies the removal of the tokens corresponding to the arc expressions from the input places and the addition of the tokens defined by the causal arcs to the output places:

$$\begin{aligned}
M_0(NMwc) &= \lambda(V)\exp(c) = \emptyset \\
M_0(EMwc) &= \lambda(V)\exp(c) = (vmc12, \\
&\qquad \textit{"Fadal VMC", "machine shop" , 8, r,} \\
&\qquad na, m12, null), \\
M_0(NPwc) &= \lambda(V)\exp(c) = \emptyset \\
M_0(EPwc) &= \lambda(V)\exp(c) = (vmc12, \textit{"Fadal VMC"}, \\
&\qquad \textit{"machine shop"}, 8, w) \\
M_0(p_5) &= \lambda(V)\exp(c) = \emptyset
\end{aligned}$$

The final marking after transition t_5 fires is the following (Fig. 3(b)):

$$\begin{aligned}
M_0(NMwc) &= \emptyset \\
M_0(EMwc) &= (vmc12, \textit{"Fadal VMC", "machine shop",} \\
&\qquad \textit{8, r, na, m12, null}), \\
M_0(NPwc) &= \emptyset \\
M_0(EPwc) &= (vmc12, \textit{"Fadal VMC", "machine shop",} \\
&\qquad 8, w) \\
M_0(p_5) &= \emptyset
\end{aligned}$$

D. Sets of Rules

Sets of Rules or rule sets provide a higher level representation mechanism. They establish relations between rules. In UPN formalism, a ruleset is represented by a subnet. Rule sets increase the efficiency of the rule control process. Grouping related rules together saves substantial amount of time in searching appropriate sets of rules which are due to be executed together and be invoked many times. In other words, a rule set reflects a restricted set of rules that are bound to be called upon together and repeatedly. They are mainly used in UPN as a mechanism to define subnets which allow for the structural composition of the rule specification knowledge. They will be further discussed in the next section where our modeling approach is presented.

IV. Modeling Approach

Generally speaking, any "company policy" starts from the specification of general global rules which describe aggregate operations for a given entity within the system. These rules are then further refined into more detailed specifications, in a stepwise manner, until no aggregate operations are left. Following a similar concept, a hierarchical modeling method using UPN has been developed which allows the system designer to start from abstract global nets and continue with successive refinements until the desired degree of detail has been reached. In addition to the refinement of rules within each scenario, it is necessary to synthesize all scenarios for all entities to form a coherent net representing the company-wide policy in the system.

Some work in hierarchical representation using Petri nets has been done for various applications [27], [29]–[31]. We have adopted the ideas from Suzuki and Murata and the hierarchical modeling methodology used is discussed in detail in the following sections.

those variables which are already in the expressions of all arcs connected to that transition. To avoid degenerate transitions, the predicate must differ from the constant predicate *false*. The predicate is supposed to be *true* by default. Simple predicates can be grouped by means of the following logical operators: $\neg$(*not*), $\vee$(*or*) and $\wedge$(*and*).

The color set, $C(t)$, of a transition, t, $\forall t \in T$, is determined by [24]:

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

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 in the expressions of all arcs connected to the transition.

As an example, let us look at Fig. 3(a), which focuses on transition t_5 of the net shown in Fig. 2. The color sets for the associated places are as follows (the simple color sets are specified in Table I):

$$\begin{aligned} C(EMwc) &= \text{MWC} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP} \\ &\quad \times \text{MSTS} \times \text{MSTE} \times \text{RES} \times \text{ESD} \\ C(NPwc) &= \text{WCID} \\ C(EPwc) &= \text{PWC} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP} \times \text{PSTS} \\ C(p_5) &= \text{WDDC} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP} \end{aligned}$$

There are four variables in the arcs connected to t_5 (*wcid#, des#, dep#* and *cap#*). On the other hand, t_5 has an incoming arc from place *EMwc*, a database place for the record *Mwc (wcid,des,dep,cap,sts,ste,res,esd)*, and there are five attributes which do not need to be specified in the arcs to/from *EMwc: des, dep, ste, res*, and *esd*. The total set of variables and the color set for the transition t_5 are

$$\begin{aligned} V(t_5) &= (wcid\# : \text{WCID}, des\# : \text{DES}, dep\# : \\ &\quad \text{DEP}, cap\# : \text{CAP}, mste\# : \text{MSTE}, \\ &\quad res\# : \text{RES}, esd\# : \text{ESD}) \\ C(t_5) &= \text{MWCS} = \text{WCID} \times \text{DES} \times \text{DEP} \\ &\quad \times \text{CAP} \times \text{MSTE} \times \text{RES} \times ESD. \end{aligned}$$

Functions in I^- and I^+ are defined in terms of lambda expressions with the form $f(c) = \lambda(V)\exp(c)$, where $c \in C(t)$ and exp is the expression associated to the arc. For transition t_5, $V \in$ MWCS can be represented as follows:

$$V = (wcid\#, des\#, dep\#, cap\#, mste\#, res\#, esd\#)$$

UPN provides different types of arcs:

Enabling arcs are directed arcs which connect a place/fact with a transition/action and define a precondition for the transition. They indicate the data that must be in a place for a transition to be enabled and must be removed from that place on firing. Given $C(t)$ and $V(t)$ defined as above for some transition $t \in T$, if a single arc from a given $p \in P$ exists, with arc expression exp, $I^-(p,t)$ is then defined to be the function $\lambda(v_1, v_2, \cdots, v_n).\exp$ (mapping between $C(t)$ and $C(p)$). If no arc from p to t exists, $I^-(p,t)$ is the zero-function. For example, these are the enabling arcs for transition t_5 from Fig. 3(a):

- $I^-(p_5, t_5) : \exp = \begin{bmatrix} wcid\# \\ des\# \\ dep\# \\ cap\# \end{bmatrix}$,
  $\lambda(V).\exp \in [\text{MWCS}_{\text{MS}} \rightarrow \text{WDDC}_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = (wcid\#, des\#, dep\#, cap\#)$

- $I^-(EMwc, t_5) : \exp = [wcid = wcid\#]$,
  $\lambda(V).\exp \in [\text{MWCS}_{\text{MS}} \rightarrow \text{MWC}_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = EMwc(wcid = wcid\#)$
- $I^-(NPwc, t_5) : \exp = [wcid\#]$,
  $\lambda(V).\exp \in [\text{MWCS}_{MS} \rightarrow \text{WCID}_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = (wcid\#)$

Causal arcs are directed arcs which connect a transition with a place and define a postcondition for the rule. Causal arcs describe modifications to be performed in the state of the net when the transition is fired; and more concretely, they indicate which data must be added to a place after the transition has been fired. I^+ is defined in the same way as I^-, but by means of the arcs from transitions to places. For example, these are the causal arcs for transition t_5 from Fig. 3(a):

- $I^+(EMwc, t_5) : \exp = \begin{bmatrix} wcid & = wcid\# \\ des & = des\# \\ dep & = dep\# \\ cap & = cap\# \\ sts & = r \end{bmatrix}$,
  $\lambda(V).\exp \in [\text{MWCS}_{\text{MS}} \rightarrow \text{MWC}_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = EMwc(wcid = wcid\#, cap = cap\#, sts = r)$
- $I^+(EPwc, t_5) : \exp = \begin{bmatrix} wcid & = wcid\# \\ des & = des\# \\ dep & = dep\# \\ sts & = w \end{bmatrix}$,
  $\lambda(V).\exp \in [\text{MWCS}_{\text{MS}} \rightarrow PWC_{\text{MS}}]_L$ such that
  $\lambda(V).\exp(c) = EPwc(wcid = wcid\#, des = des\#, dep = dep\#, cap = cap\#, sts = w)$

Checking arcs indicate which data must be present in a place, in order to enable a transition, but no data are removed upon firing. They are represented by an enabling and a causal arc, with identical arc expressions. These are generally used to indicate database checkings. For example, transition t_2 and place *EMwc* in Fig. 2 are connected by a checking arc with the associated arc expression: *Mwc(wcid = wcid#)*. This is a precondition for t_2 to be enabled, but after the transition has been fired, no data are removed from or added to *EMwc*. Arcs between t_2 and *NMwc* and arcs between t_4 and *EMwc* are also checking arcs.

Inhibitor arcs indicate which data must not be present in a place in order to enable a transition. It is graphically represented as an enabling arc ending in a small circle.

It is also possible for a transition to request information from the user interface. This is done whenever an arc expression requires some information that has not been provided by the variables in the enabling or checking arcs.

Transition Enabling and Firing: The dynamic behavior of the net is provided by transition enabling and firing. A transition $t \in T$ is said to be enabled with respect to a color $c \in C(t)$ if the current marking M is such that

$$M(p) \geq I^-(p,t)(c) \quad \text{and } M(p) < I_o(p,t)(c), \forall p \in P.$$

An enabled transition may be chosen to fire. The firing of a transition t with respect to a color c consists of removing $I^-(p,t)(c)$ colors from each of its input places and adding $I^+(p,t)(c)$ colors to each of its output places. Each firing creates a new set of conditions, and the total number of colors in the net may change after each firing.

In order to illustrate the previous concepts, we focus on transition t_5 in Fig. 3(a). For this explanation, let us consider the following initial marking:

$$\begin{aligned}
M_0(NMwc) &= \emptyset \\
M_0(EMwc) &= (vmc12, \textit{"Fadal VMC"}, \textit{"machine shop"}, \\
&\qquad \textit{null, w, na, m12, null}) \\
&= \textit{Mwc(wcid} = vmc12, \textit{des} = \textit{"Fadal VMC"}, \\
&\qquad \textit{dep} = \textit{"machine shop"}, \textit{cap} = \textit{null}, \\
&\qquad \textit{sts} = w, \textit{ste} = na, \textit{res} = m12, \textit{esd} = \textit{null}) \\
M_0(NPwc) &= vmc12 \\
M_0(EPwc) &= \emptyset \\
M_0(p_5) &= (vmc12, \textit{"Fadal VMC"}, \textit{"machine shop"}, 8)
\end{aligned}$$

It is clear that transition t_5 is enabled with respect to the following specific color (note that the entire color set for transition t_5 is $C(t_5) =$ MWCS):

$$\begin{aligned}
c = (&vmc12, \textit{"Fadal VMC"}, \textit{"machine shop"}, \\
&\textit{null, na, m12, null})
\end{aligned}$$

Because c satisfies the following enabling rules, t_5 is then enabled.

$$\begin{aligned}
M_0(NMwc) &\geq \lambda(V)\exp(c) = \emptyset \\
M_0(EMwc) &\geq \lambda(V)\exp(c) = vmc12, \textit{"Fadal VMC"}, \\
&\qquad \textit{"machine shop", null, w na, m12, null} \\
M_0(NPwc) &\geq \lambda(V)\exp(c) = vmc12 \\
M_0(EPwc) &\geq \lambda(V)\exp(c) = \emptyset \\
M_0(p_5) &\geq \lambda(V)\exp(c) = (vmc12, \textit{"Fadal VMC"}, \\
&\qquad \textit{"machine shop"}, 8)
\end{aligned}$$

where exp represents the corresponding arc expression, and V is as defined in Section III-C

The firing of transition t_5, with respect to color c, implies the removal of the tokens corresponding to the arc expressions from the input places and the addition of the tokens defined by the causal arcs to the output places:

$$\begin{aligned}
M_0(NMwc) &= \lambda(V)\exp(c) = \emptyset \\
M_0(EMwc) &= \lambda(V)\exp(c) = (vmc12, \\
&\qquad \textit{"Fadal VMC"}, \textit{"machine shop"}, 8, r, \\
&\qquad na, m12, null), \\
M_0(NPwc) &= \lambda(V)\exp(c) = \emptyset \\
M_0(EPwc) &= \lambda(V)\exp(c) = (vmc12, \textit{"Fadal VMC"}, \\
&\qquad \textit{"machine shop"}, 8, w) \\
M_0(p_5) &= \lambda(V)\exp(c) = \emptyset
\end{aligned}$$

The final marking after transition t_5 fires is the following (Fig. 3(b)):

$$\begin{aligned}
M_0(NMwc) &= \emptyset \\
M_0(EMwc) &= (vmc12, \textit{"Fadal VMC"}, \textit{"machine shop"}, \\
&\qquad 8, r, na, m12, null), \\
M_0(NPwc) &= \emptyset \\
M_0(EPwc) &= (vmc12, \textit{"Fadal VMC"}, \textit{"machine shop"}, \\
&\qquad 8, w) \\
M_0(p_5) &= \emptyset
\end{aligned}$$

D. Sets of Rules

Sets of Rules or rule sets provide a higher level representation mechanism. They establish relations between rules. In UPN formalism, a ruleset is represented by a subnet. Rule sets increase the efficiency of the rule control process. Grouping related rules together saves substantial amount of time in searching appropriate sets of rules which are due to be executed together and be invoked many times. In other words, a rule set reflects a restricted set of rules that are bound to be called upon together and repeatedly. They are mainly used in UPN as a mechanism to define subnets which allow for the structural composition of the rule specification knowledge. They will be further discussed in the next section where our modeling approach is presented.

IV. Modeling Approach

Generally speaking, any "company policy" starts from the specification of general global rules which describe aggregate operations for a given entity within the system. These rules are then further refined into more detailed specifications, in a stepwise manner, until no aggregate operations are left. Following a similar concept, a hierarchical modeling method using UPN has been developed which allows the system designer to start from abstract global nets and continue with successive refinements until the desired degree of detail has been reached. In addition to the refinement of rules within each scenario, it is necessary to synthesize all scenarios for all entities to form a coherent net representing the company-wide policy in the system.

Some work in hierarchical representation using Petri nets has been done for various applications [27], [29]–[31]. We have adopted the ideas from Suzuki and Murata and the hierarchical modeling methodology used is discussed in detail in the following sections.

TABLE II
POSITIVE INCIDENCE FUNCTIONS

I^+		t_1	t_2	t_3	t_4	t_5
		WCID	WCID	WCID	MWCS	MWCS
NMwc	WCID	0	id	0	0	0
EMwc	MWC	0	0	$\mathbf{dbm}^{Mwc}\{wcid = p_1, sts = r\}$	$\mathbf{dbm}^{Mwc}\{wcid = p_1, des = p_2, dep = p_3, sts = h\}$	$\mathbf{dbm}^{Mwc}\{wcid = p_1, des = p_2, dep = p_3, cap = p_4, sts = r\}$
NPwc	WCID	0	0	0	0	0
EPwc	PWC	0	0	0	0	$\mathbf{dbm}^{Mwc}\{wcid = p_1, des = p_2, dep = p_3, sts = w\}$
p_1	E	0	abs	0	0	0
p_2	WCID	id	0	0	0	0
p_3	WCID	0	id	0	0	0
p_4	WCID	0	0	id	0	0
p_5	WDDC	0	0	0	$\{p_1, p_2, p_3, p_4\}$	0

TABLE III
NEGATIVE INCIDENCE FUNCTIONS

I^+		t_1	t_2	t_3	t_4	t_5
		WCID	WCID	WCID	MWCS	MWCS
NMwc	WCID	0	id	0	0	0
EMwc	MWC	0	0	$\mathbf{dbm}^{Mwc}\{wcid = p_1, sts = r\}$	$\mathbf{dbm}^{Mwc}\{wcid = p_1, des = p_2, dep = p_3, sts = h\}$	$\mathbf{dbm}^{Mwc}\{wcid = p_1, des = p_2, dep = p_3, cap = p_4, sts = r\}$
NPwc	WCID	0	0	0	0	0
EPwc	PWC	0	0	0	0	$\mathbf{dbm}^{Mwc}\{wcid = p_1, des = p_2, dep = p_3, sts = w\}$
p_1	E	0	abs	0	0	0
p_2	WCID	id	0	0	0	0
p_3	WCID	0	id	0	0	0
p_4	WCID	0	0	id	0	0
p_5	WDDC	0	0	0	$\{p_1, p_2, p_3, p_4\}$	0

A. Top-Down Stepwise Refinement Technique

The top-down stepwise refinement technique has been developed for the modeling of each scenario from an abstract and aggregate level to a detailed level. This approach necessitates the development of new Petri net modeling entities which include two types of transitions as mentioned in the previous section; one is *primitive* transitions representing primitive rules, and the other is *compound* transitions representing rule sets which can be further refined into subnets. The connections are represented by calls from one *compound* transition of the net at the abstract level to the subnets at the more detailed level. Two techniques for constructing *compound* transitions are presented below. However, the reader should note that this refinement procedure does not retain the properties at different levels of abstraction. It is not our intension to retain the same behavior from higher to lower level nets since we are only interested in analyzing the lowest level nets which represent the most detailed and implementable company policy.

Horizontal Composition of Rules: hmrules: Rules at the same level of abstraction can be connected to form subnets. This horizontal composition allows the aggregation of rules under specific criteria. Horizontal compositions are established by means of what we call "hmrules." A hmrule hm_a, specifies a relation in a subnet with a set of transitions $\{t_1, t_2, \ldots, t_m\}$, where $m \geq 1$ and t_i is defined at the level of abstraction $a, \forall t_i \in hm_a$. A subnet, defined by metarule hm_a, is composed by the set of transitions and the places that are interconnected together.

Hmrules are generally used to identify various scenarios used to refine a *compound* transition at a lower level of abstraction. This aggregation is very useful at the implementation stage, because a complete subnet is translated into one procedure, which avoids the use of local variables and makes the code more efficient. For example, if the complete subnet were implemented by several procedures, additional local variables would be required to pass the state of one procedure to the other, in order to complete the execution of the complete subnet with no user interruption.

An example of a horizontal composition is one of the subnets of the work center creation scenario, related to the release

of a work center in MRP II, shown in Fig. 2. The five rules related with that scenario are grouped together represented by transition t_2 in Fig. 5. The formal representation of that subnet is specified by its incidence functions as shown in Tables II and III. The color sets for the places are such that

$$E = \{\varepsilon\}$$
$$\text{MWC} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP} \times \text{MSTS} \times \text{MSTE} \times \text{RES} \times \text{ESD}$$
$$\text{MWCS} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP} \times \text{MSTE} \times \text{RES} \times \text{ESD}$$
$$\text{WDDC} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP}$$
$$\text{PWC} = \text{WCID} \times \text{DES} \times \text{DEP} \times \text{CAP} \times \text{PSTS}.$$

The incidence functions used in this application are listed as follows:

1) *id*: identity function ($id(V) = V$).
2) *abs*: decolorizing function ($abs(V) = \varepsilon$).
3) p_i: ith projection function ($p_i(V) = v_i$).
4)

$$dbm^R \left\{ \begin{array}{c} A_{x_1} = p_{y_1} \\ \cdots \\ A_{x_k} = p_{y_k} \\ A_{x_{k+1}} = c_{z_1} \\ \cdots \\ A_{x_l} = c_{z_{l-k}} \end{array} \right\}: \text{database mapping function.}$$

where

R represents a database relation
A_j represents jth attribute of R
p_{y_i} represents y_ith projection function
c_z's represent constant values
$x_i \not> $ no. of attributes in R, and $x_i \neq x_j$ for $i \neq j$

For example: $V \in C(t)$ and $V = \{v_1, v_2, \cdots, v_n\}$

$$dbm^R \left\{ \begin{array}{c} A_{x_1} = p_{y_1} \\ \cdots \\ A_{x_k} = p_{y_k} \\ A_{x_{k+1}} = c_{z_1} \\ \cdots \\ A_{x_l} = c_{z_{l-k}} \end{array} \right\} (V) = R(A_{x_1} = v_{y_1}, \cdots, A_{x_k} = v_{y_k}, A_{x_{k+1}} = c_{z_1}, \cdots, A_{x_l} = c_{z_{l-k}}).$$

Vertical Composition of Rules: vmrules: In order to facilitate the top-down stepwise refinement technique, UPN also allow for a vertical composition of rules. Vertical compositions in UPN are used as a mechanism to establish relations between one rule at a given level of abstraction and other rules at a lower level of abstraction. As a result, rules form an abstraction hierarchy. This abstraction facilitates the design of the model and it is in line with the natural way a company policy is defined: first with abstract and general rules, which are then refined with a higher degree of detail.

One rule can be refined and replaced by a set of rules representing more detailed specifications. This can be seen as a rule set where only part of its preconditions and postconditions are shown at the higher level of abstraction. Vertical composition is performed in such a way that rule preconditions and postconditions at a level of abstraction are preserved when working at the next lower level.

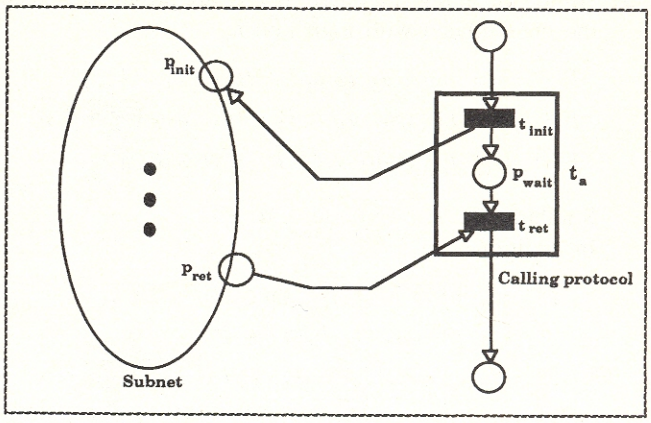


Fig. 4. Example of a subnet and a calling protocol.

In UPN terminology, a *compound* transition is refined and replaced by a subnet, where incoming and outgoing arcs from the transition are maintained in the resulting subnet.

Given a *compound* transition, t_a (high-level abstraction transition), and a level of abstraction $i-1$, a vertical metarule (called "vmrule") $vm_{i-1}^{t_a}$ is identified by the tuple $\langle t_a, hm_i \rangle$, where hm_i is a hmrule that identifies a subnet, UPN′, which refines t_a at a lower level of abstraction i.

The refinement of a *compound* transition of an abstract net produces a new UPN net which, in general, is the union of both nets minus the refined *compound* transition. A subnet being refined from a *compound* transition is formed with an attached calling protocol which establishes the link from the net at the abstract level with the subnet at the next lower level. The *compound* transition is replaced by the subnet with a calling net which contains one initiation transition (t_{init}), one waiting place (p_{wait}), and one returning transition (t_{ret}). In addition, p_{init} is the starting place of the subnet and p_{ret} is the ending place of it. One arc connecting t_{init} to p_{init} in the subnet and another arc connecting p_{ret} to t_{ret} in the subnet are added to link the calling protocol to the subnet. The calling protocol and the subnet are shown in Fig. 4.

It is also possible to explode a *compound* transition to several rule sets. Let UPN_1 be a net representing specifications at a high abstract level and let UPN_{2_1}, $\cdots$, UPN_{2_n} be the subnets representing the rule sets hm_{2_1}, $\cdots$, hm_{2_n}, from $\langle t_a, hm_{2_1}, \cdots, hm_{2_n} \rangle$, that refine the *compound* transition $t_a \in T_1$. The new UPN net is defined as follows:

1) $P = P_1 \cup P_{2_1} \cdots \cup P_{2_n} \cup P_{\text{call}}$, where P_1, P_{2_1}, $\cdots$, P_{2_n} denote the sets of places of UPN_1, UPN_{2_1}, $\cdots$, UPN_{2_n} respectively, and P_{call} denotes the set of places of the UPN for the calling protocol.
2) $T = \{T_1 - \{t_a\}\} \cup T_{2_1} \cdots \cup T_{2_n} \cup T_{\text{call}}$, where T_1, T_{2_1}, $\cdots$, T_{2_n} denote the sets of transitions of UPN_1, UPN_{2_1}, $\cdots$, UPN_{2_n} respectively, and T_{call} denotes the set of transitions of the UPN for the calling protocol.

3) All the incoming and outgoing arcs are preserved, except the ones related with transition t_a:

$$I^-(t,p) = I_1^-(t,p), \forall p \in P_1, \forall t \in \{T_1 - \{t_a\}\}$$
$$I^-(t,p) = I_{2_i}^-(t,p), \forall p \in P_{2_i}, \forall t \in T_{2_i}, 1 \le i \le n$$
$$I^-(t,p) = I_{\text{call}}^-(t,p), \forall p \in P_{\text{call}}, \forall t \in T_{\text{call}}$$

Similarly for $I^+(t,p)$ and $I_o(t,p)$.
In addition,

$$I^-(t_{\text{init}},p) = I_1^-(t_a,p), \forall p \in P_1$$
$$I^+(t_{\text{ret}},p) = I_1^+(t_a,p), \forall p \in P_1$$
$$I^+(t_{\text{init}},p_{\text{init}_i}) = \cup\{I_{2_i}^-(t,p_{\text{init}}) \mid t \in T_{2_i}, 1 \le i \le n\}$$
$$I^+(t_{\text{ret}},p_{\text{ret}_i}) = \cup\{I_{2_i}^+(t,p_{\text{ret}}) \mid t \in T_{2_i}, 1 \le i \le n\}$$
$$I^-(t,p) = \emptyset, \forall p \in P_{2_i}, \forall t \in T_{2_i},$$
where $I_{2_i}^-(t,p) \cap I_1^-(t_a,p)$,
and $I_{2_i}^-(t,p_{\text{ret}}) \neq \emptyset$
$$I^+(t,p) = \emptyset, \forall p \in P_{2_i}, \forall t \in T_{2_i},$$
where $I_{2_i}^+(t,p) \cap I_1^+(t_a,p)$,
and $I_{2_i}^+(t,p_{ret}) \neq \emptyset$

4)

$$M(p) = M_1(p) + M_{2_i}(p), \forall p \in P_1 \cap P_{2_i}, 1 \le i \le n$$
$$M(p) = M_1(p), \forall p \in \{P_1 - (P_1 \cap P_{2_1}) - \cdots - (P_1 \cap P_{2_n})\}$$
$$M(p) = M_{2_i}(p), \forall p \in \{P_{2_i} - (P_1 \cap P_{2_i})\}, 1 \le i \le n$$

An example of the top-down refinement technique is the refinement of the rule "Release of a work center in MRP II" (Fig. 2), which is the *compound* transition t_2 in the abstract scenario "Creation of a work center via MRP II" (Fig. 5). During the refinement procedure a calling protocol is used, and the refined UPN is shown in Fig. 6 with initial marking. Preconditions and postconditions represented by transition t_2 are preserved (arcs to/from *EMwc*, *NPwc*, and *EPwc*).

B. Synthesis Technique

It is necessary to synthesize related scenarios to build the company wide policy, represented by one single net. There have been some synthesis techniques presented in [29]–[31] based on their application domain. In our work we take advantage of the features of UPN, such as global places that reflect the state of the databases involved. In addition, we take advantage of standard modification procedures embedded in the database management systems associated with each application systems. The synthesis of UPN is achieved through the use of places of global scope (see Section III-B).

Due to the representation of database states in UPN, every scenario involves checkings, updates and retrievals in some of the databases of the system. Therefore, connections from and to global places, which represent database states, exist in every UPN. These global places provide the connectivities between all scenarios. For example, the scenario "Creation of a work center via MRP II" (Fig. 5) and "Removal of a work center via MRP II" (Fig. 7) are synthesized through the merging of the

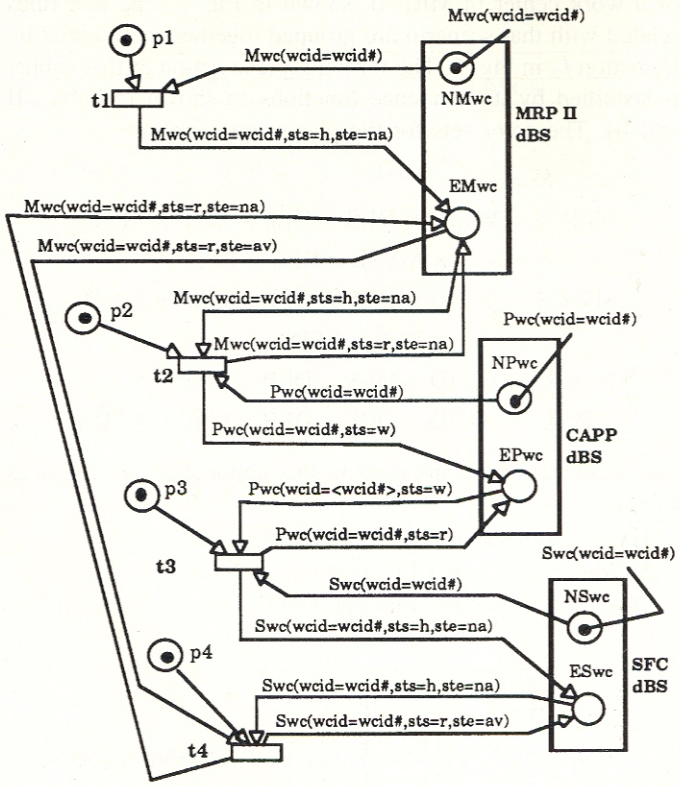


Fig. 5. UPN graph of the scenario: "Creation of a work center via MRP II" at an abstract level with initial marking.

work center global places of the databases in MRP II, CAPP, and SFC. The synthesized net is shown in Fig. 8.

Alternatively, standard modification procedures (representing the default procedures maintained in database management systems) can be used to synthesize subnets together. As shown in Fig. 9, there are three procedural levels in a typical database management system. The lowest level is the physical procedural level which represents the actual database changes. The procedures at this level are represented here by *ins, del*, and *upd*. The second level is the modification procedural level which represents the modification procedures with build-in rules in the DBMS. The procedures at this level are represented here by *insert, delete*, and *update*. The highest level is the application procedural level which represents the user-defined procedures (*create, release, remove*, etc.) based on company policy and expert rules, that are the focal points of this work. Note that the modification procedural level could also be modeled using the same methodology and techniques presented in this paper. In that case, instead of the global places defined in Section III-B, these modification procedures would become the links between application procedures which have been developed in various scenarios. The reasoning is that

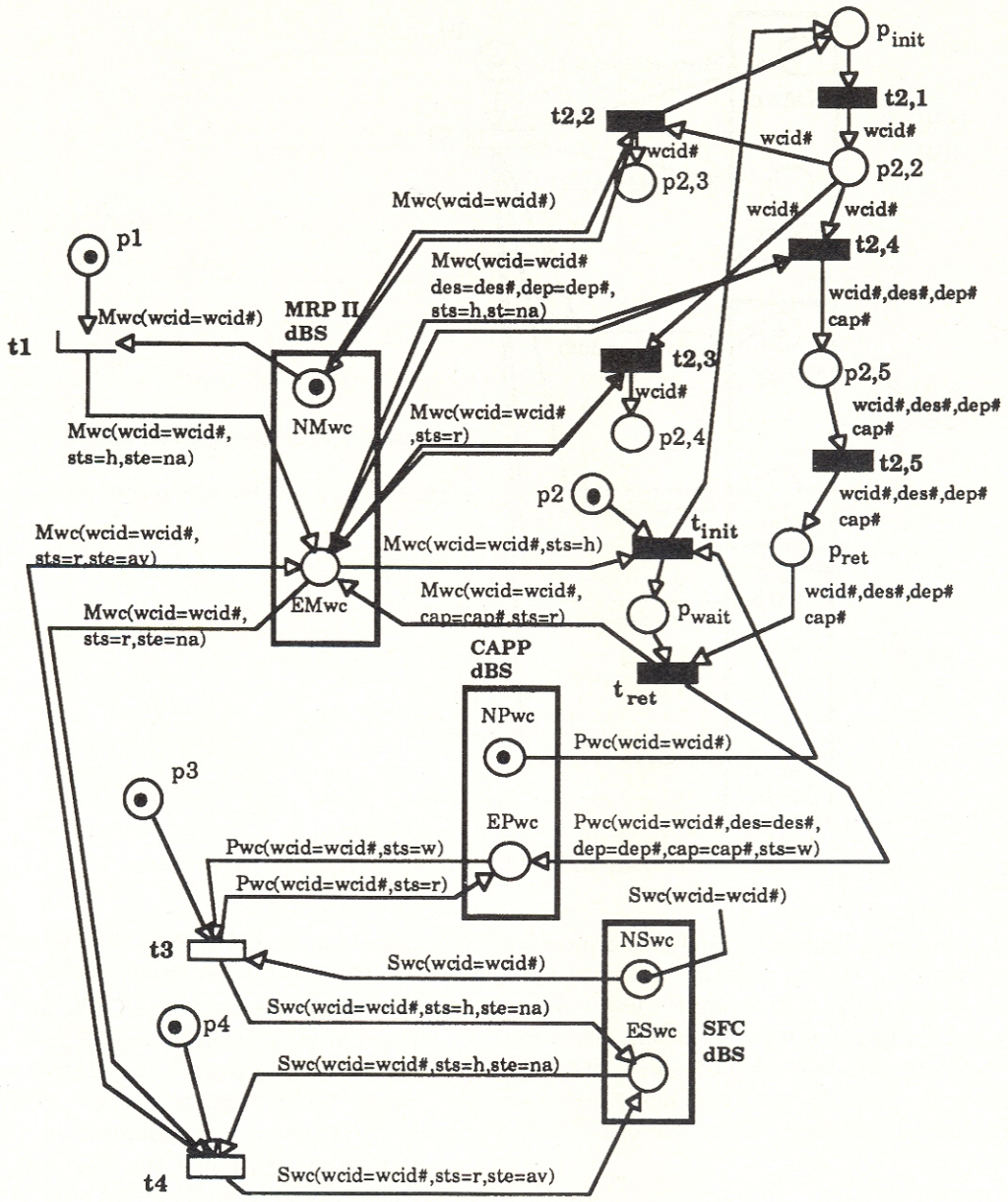


Fig. 6. Partially refined UPN of the scenario "Creation of a work center via MRP II" with initial marking.

all application procedures call for the standard modification procedures embedded in the DBMS in order to make changes in the databases involved.

In order to avoid "re-inventing the wheel," we decided not to model the database management systems of the manufacturing application systems involved using UPN, but to define the database states as global variables and to interface the application procedures (company policy) through those global places representing database states.

V. Conclusion

A formal structured representation schema for rule-based systems has been developed and applied for the management and control of information flow between manufacturing applications. The representation schema, called UPN, is an extension of the graphical and formal capabilities of colored Petri nets to express and validate if-then rules. UPN are capable of representing user specification rules as well as database updates and retrievals, which is necessary for controlling information flow within current and future distributed database systems. Related rules can be aggregated at the same level of abstraction and the relation between one rule at a given level of abstraction and a set of aggregated rules at the next lewer level of abstraction is also allowed. These facilities provide a mechanism for stepwise refinements in modeling and validation.

Future work includes the improvement of the representation schema to deal with more complex conditions and its extension to other application domains. Also, the joint use of UPN with other Petri net based knowledge representation schemas used

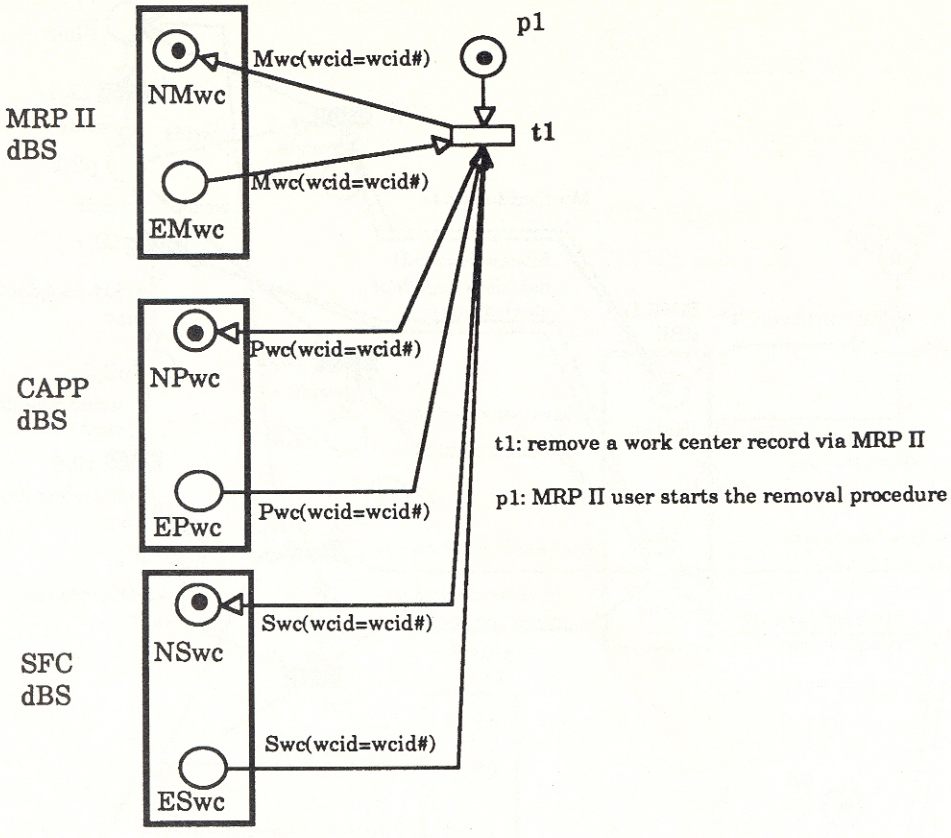


Fig. 7. UPN graph of the scenario: "Removal of a work center via MRP II" at an abstract level with initial marking.

to model factory layout and performance [32], [33] can provide an integrated framework for factory modeling and validation of design specifications. On the other hand, the use of Petri net structural features (incidence matrix, reachability tree and invariants) and structural properties (boundedness, liveness, mutual exclusiveness, etc.) that characterize completeness and consistency of the rule-based system is also one of our lines of interest.

APPENDIX

A. Creating New Work Centers in the System: Add via MRP II

Work centers are originated in the system primarily through the manufacturing resource planning (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 computer-aided process planning (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, shop floor control (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 typical scenario of adding a work center in the system is presented below:

1) MRP II user enters the basic data (WC ID, Description, Department) to create the Work Center Record with Hold (h) status in MRP II. The system then
   checks WC ID does not exist in MRP II
2) MRP II user enters additional information (Capacity, Resource Code, Rate Code, Dispatch Horizon, 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 prompts for it 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 from h to release (r) in MRP II
   Skeletal WC Record automatically created in CAPP with working (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 a modify transaction. CAPP user then releases the WC. Otherwise, if the additional information was not entered, the system prompts for it during releasing of the WC in CAPP.
   checks WC ID exists in CAPP

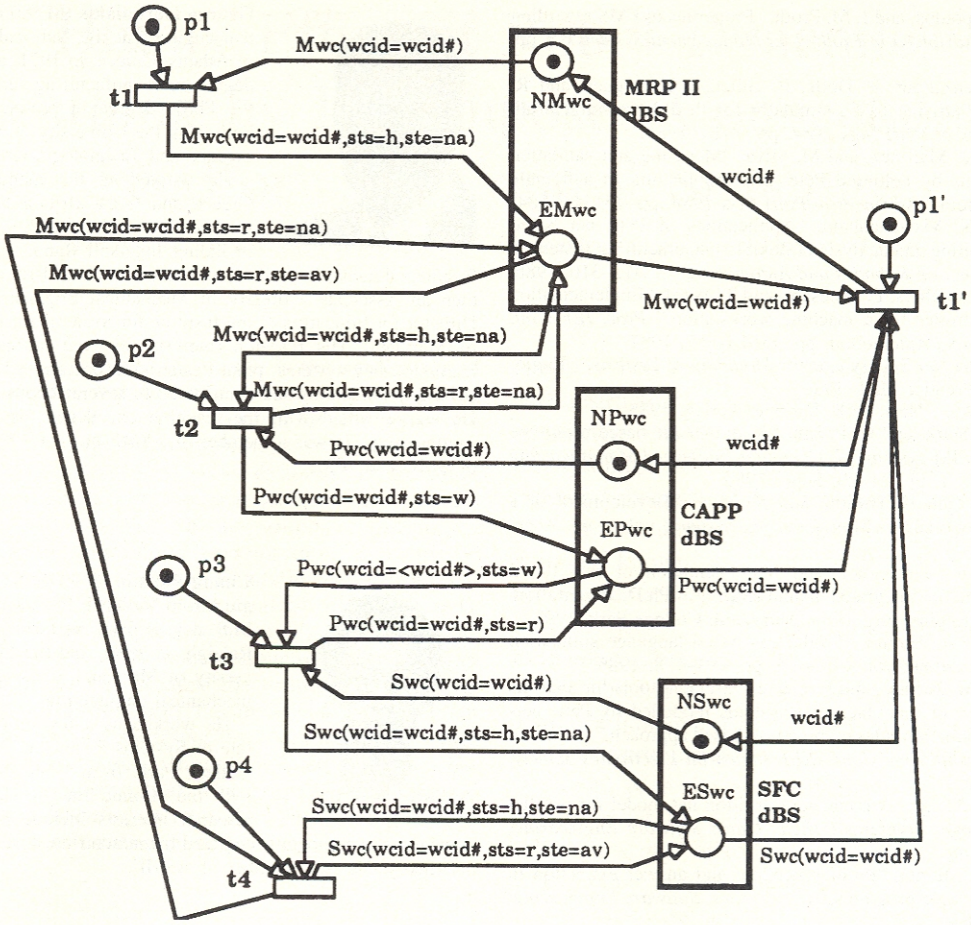


Fig. 8. UPN graph of the synthesized scenario: "Creation and removal of a work center via MRP II" at an abstract level with initial marking.

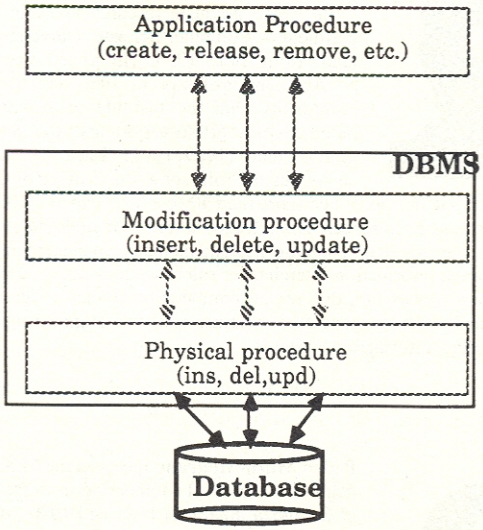


Fig. 9. Procedural levels in database management systems.

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 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 available (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 from h to r in SFC, and state changed from na to av in SFC and MRP II

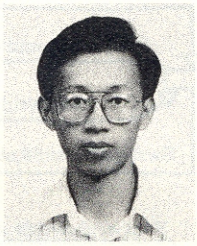
REFERENCES

[1] D. M. Dilts and W. Wu, "Using knowledge-based technology to integrate CIM databases," *IEEE Trans. Data and Knowledge Engineering*, vol. 3, no. 2, pp. 237–245,1991.
[2] Staff Report, "ICAM Program Overview," Materials Laboratory, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, OH, 1983.
[3] G. Doumeingts, B. Ballespir, D. Darricau, and M. Roboam, "Design methodology for advanced manufacturing systems," *Computers in Industry*, vol. 9, pp. 271–296, 1987.
[4] M. Courvoisier, R. Vallette, *et al.*, "a programmable logic controller based on a high level specification protocol," in *Proc. IECON Conf. on Industrial Electronics*, pp. 174–179, 1983.
[5] D. H. Crockett and A. A. Desrochers, "Manufacturing workstation control using Petri-nets," *Tech. Rep. 83, Robotics and Automation Laboratory*, Department of Electrical, Computer, and System Engineering, Rensselaer Polytechnic Institute, Aug. 1986.
[6] R. Ravichandran and A. K. Chakravarty, "Decision support in flexible manufacturing systems using timed Petri-nets," *J. Manufacturing Systems*, vol. 5, no. 2, pp 89–100, 1987.
[7] A. A. Merabet, "Synchronization of operations in a flexible manufacturing cell: The Petri-net approach," *J. Manufacturing Systems*, vol. 5, no. 3, pp. 161–169, 1987.

[8] N. Dridi, V. I. Leopoulos, and J. M. Proth, "Properties of FMS regarding optimal control," *Advanced in Production Management Systems 85*, pp. 325–327, 1985.
[9] P. Alanche, K. Benzakour, F. Dolle, P. Gillet, P. Rodrigues, and R. Vallette, "PSI: A Petri net based simulator for flexible manufacturing systems," *Advances in Petri Nets 1984*, pp. 1–14, 1984.
[10] H. Alla, P. Ladet, J. Martínez, and M. Silva, "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.
[11] M. Kamath and N. Viswanadham, "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.
[12] D. Crockett, A. Desrochers, F. Dicesare, and T. Ward, "Implementation of a Petri net controller for a machine workstation," *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 1861–1867, 1987.
[13] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice Hall, 1981.
[14] W. Reisig, *Petri Net*. New York: Springer-Verlag, 1985.
[15] G. Harhalakis, L. Mark, and C. P. Lin, "A knowledge-based prototype of a factory-level CIM system," *J. Computer Integrated Manufacturing Systems*, vol. 2, no. 1, pp. 11–20, Sept., 1989.
[16] G. Harhalakis, C. Lin, H. Hillion, and K. Moy, "Development of a factory level CIM model," *J. Manufacturing Systems*, vol. 9, no. 2, pp. 116–128,1990.
[17] C. P. Lin, "Design, verification and implementation of rule based information system for integrated manufacturing," Ph.D. dissertation, Dept. Mechanical Engineering, Univ. Maryland, College Park, 1991.
[18] R. Bauman and T. A. Turano, "Production based language simulation of Petri nets," *Simulation*, vol. 47, no. 5, pp. 191–198, 1986.
[19] H. Garnousset, J. M. Farines, and E. Cantu, "Efficient tools for analysis and implementation of manufacturing systems modelled by Petri nets with objects: A production rules compilation-based approach," presented at *IECON'89 Fifteenth Ann. Conf. IEEE Industrial Electronics Society*, Philadelphia, 1989.
[20] T. Murata and D. Zhang, "A predicate-transition net model for parallel interpretation of logic programs," *IEEE Trans. Software Engineering*, vol. 14, no. 1, 1988.
[21] G. Peterka and T. Murata, "Proof procedure and answer extraction in Petri net model of logic programs," *IEEE Trans. Software Engineering*, vol. 15, no. 2, 1989.
[22] M. D. Zisman, "Use of production systems for modelling asynchronous concurrent processes," in *Pattern Directed Inference Systems*, D. A. Watterman, and F. Hayes-Roth, eds. London: Academic Press, pp. 53–68, 1978.
[23] H. J. Genrich and K. Lautenbach, "System modelling with high-level Petri nets," *Theoreticl Computer Science*, vol. 13, pp. 109–136, 1981.
[24] K. Jensen, "Colored Petri Nets," *Petri Nets: Central Models and Their Properties. Advances in Petri Nets 1986, Part I. Proceedings of an Advanced Course, Bad Honnef, 8–19. Sept. 1986*, pp. 248–299, G. Goos and J. Hartmanis, eds. Heidelberg: Springer-Verlag, 1987.
[25] C. Sibertin-Blanc, "High-level Petri nets with data structure," presented at *6th European Workshop on Application and Theory of Petri nets*, Helsinki, Finland, June 1985.
[26] C. L. Beck and B. H. Krogh, "Models for simulation and discrete control of manufacturing systems," *Proc. IEEE Int. Conf. on Robotics Automation*, pp. 305-310,1986.
[27] R. Valette, "Analysis of Petri nets by stepwise refinements," *J. Computer and System Sciences 18*, pp. 35–46, 1979.
[28] C. Sibertin-Blanc, "High-level Petri nets with data structure," presented at *6th European Workshop on Application and Theory of Petri nets*, Helsinki, Finland, June 1985.
[29] I. Suzuki and T. Murata, "A method for stepwise refinement and abstraction of Petri nets," *J. Computer System Science*, vol. 27, pp. 51–76, 1983.
[30] Y. Narahari and N. Viswanadham, "A Petri Net Approach to the Modeling and Analysis of Flexible Manufacturing Sy
[31] stems," *Ann. Op. Res.*, vol. 3, pp. 381–391, 1985.
[32] M. D. Jeng and F. DiCesare, "A review of synthesis techniques for Petri nets," *Proc. IEEE Computer Integrated Manufacturing Systems Conf., RPI*, May 1990.
[33] P. R. Muro, J. L. Villarroel, J. Martínez and M. Silva, "A knowledge representation tool for manufacturing control systems design and prototyping," *INCOM'89, 6th IFAC/IFIC/IFORS/IMACS Symp. Information Control Problems in Manufacturing Technol.*, Madrid, Spain, Sept. 1989.
[34] P. R. Muro, J. Ezpeleta and J. L. Villarroel, "Knowledge based manufacturing modeling and analysis by integrating Petri nets," *Proc. the IMACS Symp. on Modelling and Control of Technological Systems*, Lille, France, May 7–10, 1991.
[35] L. Mark, "Self-Describing Database Systems—Formalization and Realization," (Ph.D. thesis) TR-1484 Comp. Sci. Dept., Univ. Maryland, 1985.

George Harhalakis (M'90) qualified as a mechanical engineer at the National Technical University of Athens, Greece, in 1971, and received the M.Sc. degree in manufacturing technology in 1981 and the Ph.D. degree in production control in 1984, both from the University of Manchester Institute of Science and Technology, Great Britain.

He worked in the manufacturing industry in Greece and Great Britain for 12 years, and also worked as consultant with a number of companies, including Ingersoll Rand, Staveley Vessels, Curtis Engine, Ohmeda, Bata Shoe, and others in Great Britain and the U.S. He was then an Associate Professor of Mechanical Engineering and the Associate Director for Education of the Institute for Systems Research of the University of Maryland. His research interests revolved around computer-integrated manufacturing systems, plant design, and operation. He authored more than seventy publications and contributed to several books as an author or editor. He was awarded with the teaching excellence award of the College of Engineering. He was a member of ASME and APICS, and a senior member of SME and IIE.

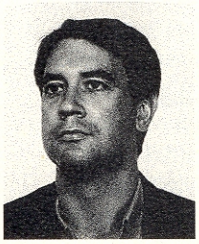


Chang-Pin Lin (S'90–M'91) received the B.S. degree from National Taiwan University in 1982, the M.S. degree from North Carolina State University, Raleigh, in 1986, and the Ph.D. degree from University of Maryland, College Park, in 1991, all in mechanical engineering.

He worked as a research engineer at the Institute of Systems Research, University of Maryland, College Park, from 1989 to 1992. He is currently with the Atomic Energy Council in Taiwan. His research interests include production control and management, computer-integrated manufacturing, system modeling, and Petri net theory. He is a member of ASME.

Leo Mark received the M.S. and Ph.D. degrees in computer science from Aarhus University, Denmark, in 1980 and 1985, respectively

He was a graduate research assistant at the Department of Computer Science, University of Maryland, from 1983–1985. He joined the faculty as an Assistant Professor in 1986 and had joint appointments with the Institute of Advanced Computer Studies (1986–1989) and the Systems Research Center (1989–1992). Since 1992 he has been an Associate Professor at the College of Computing, Georgia Tech. He has published more than 40 scientific papers in books, refereed journals, and conferences. His research interests include database system architecture, data models, database design; metadata management, data dictionary systems, information interchange; rulebases transaction dependencies; efficient query processing, differential computation models; transaction time databases; and database generator systems, software engineering databases, and engineering information systems.



Pedro Muro-Medrano received the M.S. and Ph.D. degrees in electrical engineering from the University of Zaragoza, Spain, in 1985 and 1990, respectively.

He was a graduate research assistant at the Department of Electrical Engineering and Computer Science, University of Zaragoza, 1985–1987. He was a visiting scholar at the Robotics Institute, Carnegie Mellon University, and a visiting research associate at the Systems Research Center, University of Maryland. He joined the faculty of the University of Zaragoza as an Assistant Professor in 1989, and since 1992 he has been an Associate Professor. His research interests include artificial intelligence applications, object oriented software engineering, and engineering information systems.

IEEE TRANSACTIONS ON

SYSTEMS, MAN, AND CYBERNETICS

A PUBLICATION OF THE IEEE SYSTEMS, MAN, AND CYBERNETICS SOCIETY

JANUARY 1995 VOLUME 25 NUMBER 1 ISYMAW (ISSN 0018–9472)

PAPERS

CORRESPONDENCE