Journal of the Korean Institute
of Industrial Engineers
Vol.23, N0.4, December, 1997

유연공정계획 표현을 위한 객체지향형 페트리네트 모델

Object-Oriented Petri Net Model for Representation of Flexible Process Plan*

이경휘**

Lee, Kyung Huy**

Abstract

In this research, an object-oriented Petri net model for representing a flexible process plan is proposed, which is hierarchically multi-faceted for supporting planning, scheduling, and shop floor control functions. The multi-faceted process plan model consists of the following: a) an object model which represents an object-oriented data model, b) a static model which represents a process flow model with process alternatives, and c) a dynamic model which represents a process activity model with resources alternatives, of a flexible process plan. Petri nets allow the static and the dynamic process plan models to be represented in a unified formalism with an ease of model transformation. The multi-faceted process plan model suggested in this paper, is illustrated with a prismatic part in comprehensive detail.

1. Introduction

Mechanical process plans describe the production process of a predefined workpiece and a specific set of resources which performs this production process [3]. Process plans provide all the information about production process sequence, machine routeing, and resources to shop floor control as well as scheduling. Recently, flexible process plans have been suggested for overcoming the rigidity problem of a single process plan in a sequential engineering approach [17].

Several methodologies for generating or representing flexible process plans are known in the related literature. First of all, a methodology for generating flexible process plans has been proposed in a conceptual manner [9] or in a formal manner based on Petri net formalism [15, 16]. In process plan representation, one important research is the ISO STEP-based representation model of a mechanical process plan, in which a flexible process plan was formally represented in an AND/OR graph for standardization [8]. In a similar manner, Mettala and Joshi [18] used an AND/OR graph for representing manufacturing process plans. They proposed the AND/OR graph representation of alternative process plans and routeings for FMS control activities. A similar representation scheme is proposed in [1]. Several workers have utilized

* 이 논문은 1996학년도 한국학술진흥재단의 공모과제 연구비에 의해 연구되었음.

** 대전대학교 정보산업통신공학부 산업공학과

Petri nets to represent flexible process plans [10, 16]. Lee and Kim [14] proposed an integration approach of process plan selection and scheduling based on a flexible process plan which is represented in a modified Petri net. Cho *et. al.* [4] and Lee [12] showed that a process plan can be formally extended for a shop floor control problem on a unified model of an AND/OR graph or a Petri net. Although all these results guaranteed that a formal process plan can be utilized for flexibly supporting shop floor control activities such as planning and scheduling, there exist no process plan models which progressively represent all of the syntax and the semantics of flexible process sequence/routeings, and activities for supporting shop floor control in a unified formal basis.

In this paper, an object-oriented Petri net model for the representation of a flexible process plan is proposed, which is multi-faceted for representing a flexible process plan for supporting planning, scheduling and shop floor control. The generic nature of a multi-faceted, object-oriented Petri net model allows the progressively formal derivation of various requirements of a) an object process plan such as alternative processes and precedences, b) a static process plan such as process routeings and scheduling, and c) a dynamic process plan such as process activities. The aim of this paper is to make progress toward modelling a flexible process plan for planning, scheduling and shop floor control in a unified, formal basis. An illustrative example is also provided for validating the proposed process plan model.

2. Formal Integration of Process Planning and Shop Floor Control

In this chapter, an overall architecture of the formal integration model of process planning and shop floor control (*IP²C*, Integrated Process Planning and Control) is explained in comprehensive detail. In the *IP²C*, the model architecture and the role of a process plan model are also provided in this chapter.

2.1 Architecture of IP²C

The formally integrated model of process planning, scheduling, and shop floor control [13], as shown in Figure 1, consists of the following four functions interfaced with four facets of the process plan model which are based on a unified Petri net with objects:

- *Flexible Process Planning (FPP)* [15]: *FPP generates a flexible process plan, which is modeled in an object-oriented process plan data model with multiple processes, sequences and resources.*
- *Shop Floor Configuration (SFG): In SFG, a shop floor is configured in physical layout and composition, which is modeled as lots of resources in a hierarchical structure. SFG, represented in a Petri net with objects, provides configurational details on a shop floor to scheduling and shop floor control activities.*
- *Shop Floor Operation Planning & Scheduling (SOP): SOP performs selection of an optimal process plan and operational schedule on an execution basis, with considering shop floor status such as system load, capacity, and availability. In order to do it, an object process plan model generated in FPP is transformed to the suitable structure of a process plan, that is: a flow critical path or an activity critical path of a flexible process plan on a timed Petri net.*
- *Shop Floor Control (SFC): SFC performs monitoring and controling a shop floor on a control model of a flexible process plan and SFG, both of which are based on a timed Petri net.*

The formal integration model, *IP²C*, has benefits of the following: *a)* providing alternative process plans, *b)* determining a process plan/schedule simultaneously with considering shop floor status, and *c)* modeling them in a unified formal model of Petri nets which can also support a shop floor control function through progressive refinement of the process plan model.

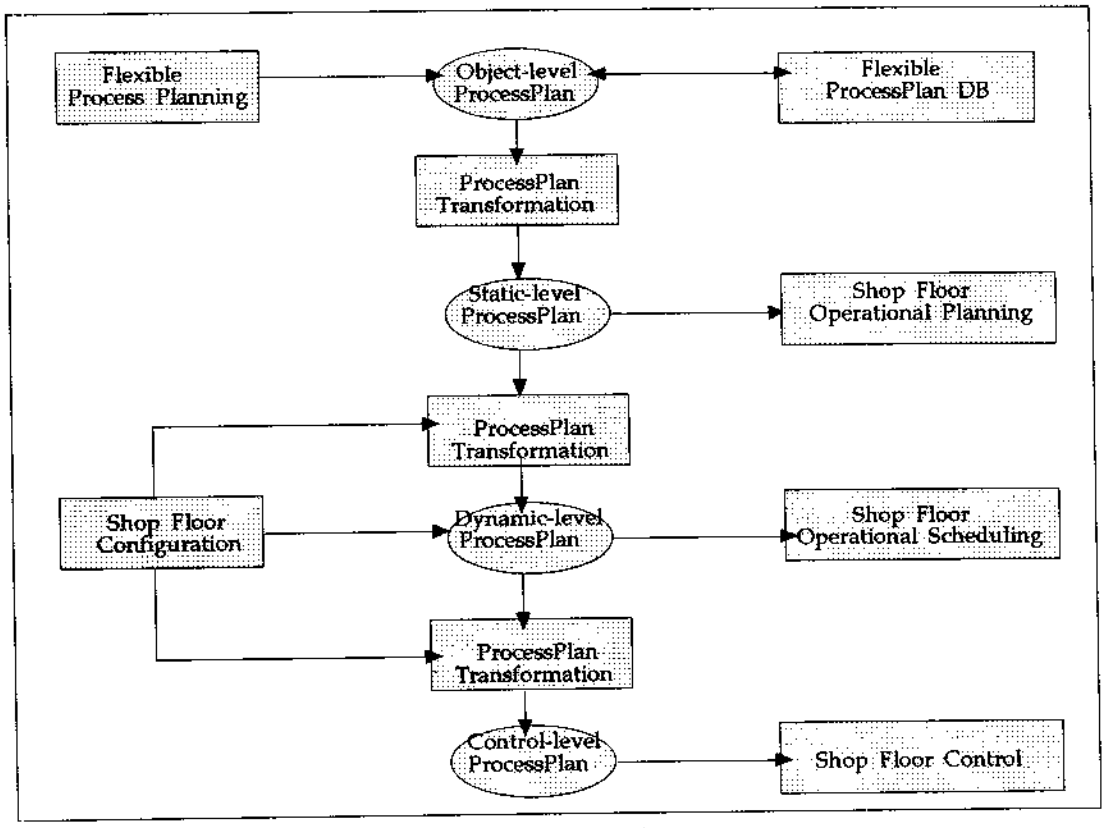


Figure 1. Model generation procedure of IP²C

2.2 Role of Process Plan

How to represent a flexible process plan, which process planning results in, is a very critical issue in making automated manufacturing systems highly reliable because the process plan must guide all production progress in a technical view. A process plan must be made real-time for putting all the process information into shop floor control by foreseeing the current state or the possibly changed state of shop floor. A process plan must also have more detailed information for manufacturing systems which are getting more automated, than that for traditional manufacturing systems. In my opinion, however, it is very difficult to model such process plan requirements in separate manner with shop floor control. Although traditional process plans have contributed themselves in several internal aspects, they have to be changed according to the current state of shop floor or further be refined for shop floor control because they have been generated without considering such the realistic information requirements.

In this paper, we propose a process plan model which is multi-faceted for downstream applications such as representing a flexible process plan, storing it, and supporting planning, scheduling and shop floor control, respectively. Table 1 shows a process plan data model proposed here. The process plan model is formally based on the combination of objects and Petri nets, which incorporates *a*) the semantics of each application and *b*)

the syntax of being transformed among the facets. The process plan model is stratefied into the four facets as listed below:

- *The object model plays the role of representing process plan objects, storing them into a process plan database, and retrieving the process plan for downstream applications such as planning, scheduling, and shop floor control.*
- *The static model (a process flow Petri net) plays the role of representing a process sequence with alternatives and then supporting a set of flow critical paths for shop floor operation planing.*
- *The dynamic model (an activity flow Petri net) plays the role of supporting a set of activity critical paths with resources for shop floor scheduling.*
- *The control model (a dynamic event flow Petri net) supports shop floor control activities in an execution basis.*

3.1 Object Model of Flexible Process Plan

Object-oriented modeling provides a mechanism for representing information by using models organized around real-world objects and concepts. The basic construct is the object which combines both the data structure and its behavior into a single entity. An object may be defined as a concept, abstraction, or thing with crisp boundaries and meaning for the problem at hand [15]. Depending on the information requirements of the downstream applications such as planning, scheduling and shop floor control, a flexible process plan must include a variety of objects and their characteristics about production processes and resources, in flexible manner. The object model of a flexible process plan includes information about Part, Feature, Transition_Process, Setup _Process, MachiningOperation_Process, Machine, Fixture, Tool, Cutting Parameters, Cost, Time, and the like. The object process plan model is described as the following

Table 1. Multi-faceted process plan model

facet	modeling concept	formalism	functions supported
Object-level	Process Object Model	Object-oriented Data Model	Flexible Process Plan Data Base
Static-level	Process Flow Model	Place-timed Petri Net Model with Objects	Flexible Operation Planning
Dynamic-level	Process Activity Model with Resources	Place-timed Petri Net Model with Objects	Flexible Operation Scheduling
Control-level	Shop Floor Dynamic Event model	Colored, Timed Petri Net Model with Objects	Shop Floor Control

3. Multi-faceted Model of Flexible Process Plan

In this chapter, a multi-faceted process plan model is explained in comprehensive detail. The process plan model proposed here covers the object, static and dynamic model of a flexible process plan (We will present the control model of a flexible process plan in another paper).

class/attribute relation graph in the form of an acyclic directed graph:

Definition 1: The object model of a flexible process plan, OP²N (an object model of Process Plan Net), is an acyclic directed graph with objects with the following five tuples:

OP^2N := (C, R, A_C, A_R, D) where

1) C denotes a finite set of classes (nodes with type) which consists of the following items in a process plan:

1.1) Internal Class (IC) means an internally defined class within process plan data model. IC can be further classified into the following classes:

−Meta Class (ProcessPlanClass) is a root class of all the other classes, which represents a whole process plan of a part;

−Process Class∈{SetUpProcess, OperationProcess, TransitionProcess}. Class Process is a finite set of classes denoting processes which consist of process plan activities on each level i∈{Machine_Level (M_), Fixture_Level (F_), Cutting_Level (C_)}. A process object is further classified into mutually excluded subprocess objects like Resource, Method, Feature.

−MethodClass∈{SetupMethod, OperationMethod, TransitionMethod}. Class Method means the methodology for doing a process, which is specialized into SetupMethod, CuttingMethod, and TransitionMethod, on each level.

1.2) External Class (XC) means an externally defined class within process plan data model, that is: XC has a reference point or ID to external classes such as machine resource DBs or part/feature DBs etc., in class attributes. The following classes are included in XC:

−ResourceClass∈{SetupResource, OperationResource, TransitionResource} (∈{machine, jig, fixture, tool, pallet, conveyor, robot, etc.}). Class Resource is a finite set of classes which denote the suggested resources for making the process. The class is specilized into SetupResource, CuttingResource, and Transition Resource on each level, each of which is corresponded to machine, jig, fixture, pallet, or tool.

−FeatureClass∈{SetupFeature, OperationFeature, TransitionFeature}. Class Feature is a finite set of classes which represent a part feature referred during making the process. The class is specilized into SetupFeature, TransitionFeature and OperationFeature.

2) R denotes a finite set of relations (links with type) among the process plan objects such as classes and attributes, which consists of the following:

2.1) Class Abstraction hierarchy: Class C is specialized (reversely generalized) into several immediate subclasses SCs which are mutually excluded each other. Class hierarchy in a process plan is structured as follows:

−Part/Feature hierarchy relation∈{RotationalProcessPlan, PrismaticProcess Plan, SheetMetalProcessPlan}. Class ProcessPlan is specialized into Rotational ProcessPlan, PrismaticProcessPlan, and SheetMetal-ProcessPlan, according to the shape of a part.

−Process hierarchy relation∈{M_Process, F_Process, C_Process}. Class Process is specialized into M_Process, F_Process, and C_Process. Processes on each level are further specialized into SetupProcess, OperationProcess, and TransitionProcess, correspondingly.

2.2) Class composition hierarchy: Class C is aggregated from several other classes SCs which are part of the class. Class composition hierarchy in a process plan is listed as follows:

−Process composition hierarchy∈{ProcessResource, ProcessMethod, ProcessFeature}. Class Process is aggregated from Resource, Method, Feature classes for all kind of processes on each level.

3) A_C denotes a finite set of attributes for defining the process plan attributes/ objects which are associated with a class C.

4) A_R denotes a finite set of attributes for defining relations between classes/ attributes of a process plan which are associated with a relation R.

5) D is a set of domains which support the value of attributes A_C and A_R. It includes user-defined data types (classes) as well as basic domains such as

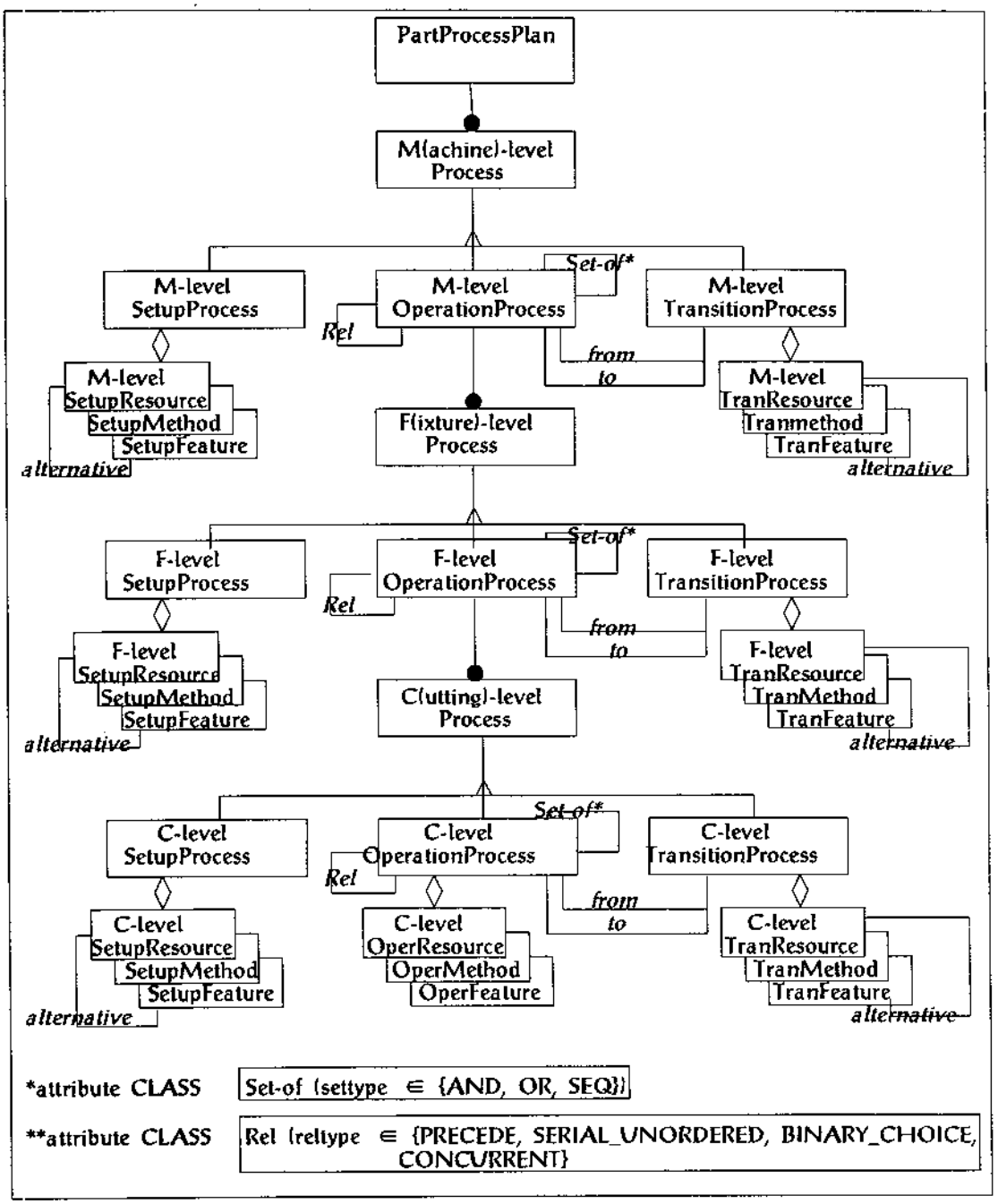


Figure 2. Object model of flexible process plan

integers, reals, characters, strings, etc.

Figure 2 shows the object model of a flexible process plan, as described above. The object process plan model proposed here has the several benefits of the following:

- *The OP²N allows modelling the flexible process plan which is capable of representing processes, resources, methods, or sequences with alternatives.*
- *The OP²N is internally or externally defined such that it includes almost all activities, resources, methods, and features, required for shop floor control automated manufacturing systems as well as traditional manufacturing systems.*
- The OP²N is structured according to process and resource hierarchy on machine, fixture, tool levels, respectively. It has benefits for the process plan model to be easily transformed to the proper hierarchical form for downstream applications such as scheduling and shop floor control activities.

Example: An illustrative prismatic part, *P*, is given in Figure 3 and the *OP²N* for producing the part is shown in Figure 4, correspondingly. In Figure 4, a *ProcessPlan* object of the part is defined in a root class, which is immediately decomposed to *M_Process*, *M_Setup* and *M_Transition* objects (*mp_#10000*, *ms_#10000*, *mt_#10000*). In the figure, the *M_Process* object consists of the several '*set-of M_Process*' (*mp_#12000*, *mp_#13000*, *mp_#15000*, *mp_#16000*) and '*single M_Process*' (*mp_#11000*, *mp_#14000*, *mp_#15000*) objects, respectively. The '*set-of M_Process*' object is in turn comprised up another '*setof M_Process*' or '*single M_Process*' objects in a recursive manner. A '*set-of M_Process*' object is defined with the association type (*SEQ*, *OR*, *AND*) of the lower *M_Process* objects. A *single M_Process*, which means a 'possibly unchangeable process' on machine level, is decomposed to *F_Process*, *F_Setup* and *F_Transition*, on fixture level. In the figure, for example, the *mp_#14000* M_Process object is decomposed to the *fp_#14000*, *fs_#14000* and *ft_#14000* F_Process objects, respectively. Similar with the *M_Process*, the *F_Process* objects can consist of several '*set-of F_Process*' or '*single F_Process*' objects recursively. A '*F_Process*' object may be further decomposed into *C_Process* objects on cutting (tool) level.

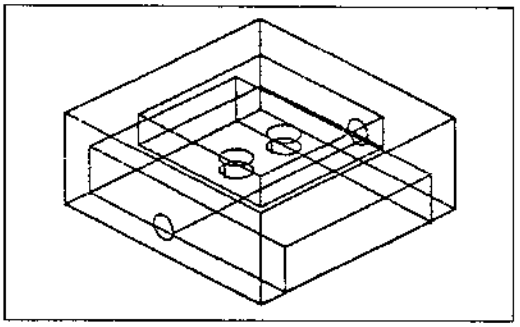


Figure 3. Part example

3.2 Static Model of Flexible Process Plan

In this section, the static model of a flexible process plan is explained in comprehensive detail. The static process plan model represents the flow of process tasks from an initial part (raw stock) to a final part. The process flow includes process with alternatives as well as process sequence with alternatives. At the static process plan model, each task is specified in terms of the possibly nonchangeable process tasks like setup, transition, operations, with reference to the relevant objects such as resources, methods, and features. The static model is designed for logical processing which includes all the necessary processes and their flow relations, but not for dynamic processing of a shop floor. The static process plan model can be described in the form of a place-timed Petri net, which is listed below:

Definition 2: The Static Model of a flexible process plan, SP²N (Static model of Process Plan Petri Net), is formally described as a place-timed Petri net with objects with the following ten tuples:

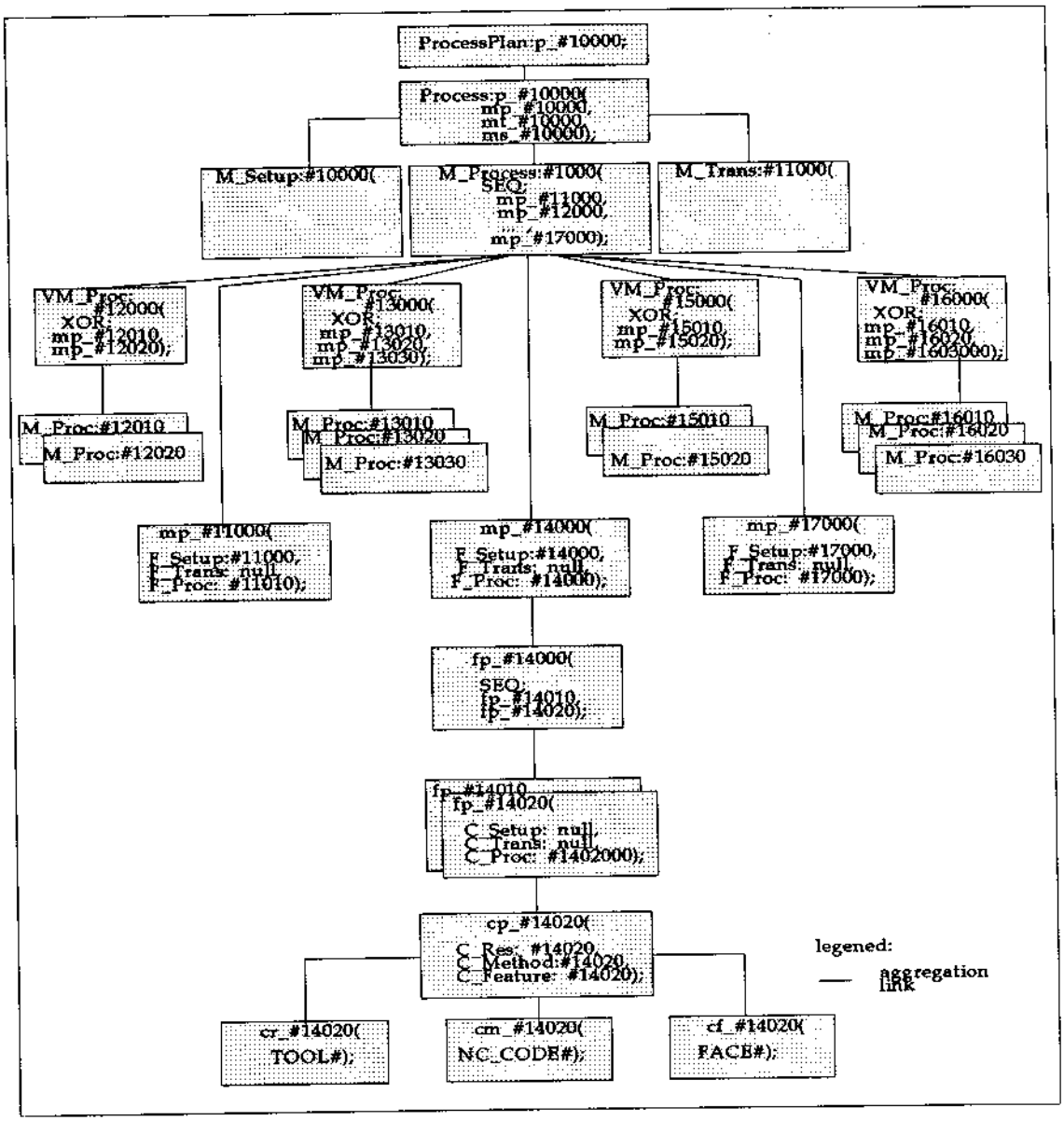


Figure 4. Instance object structure of object process plan

$SP^3N = \langle P, T, F, A_P, A_T, d, f_P, f_T, f_d, M_0 \rangle$

where the following applies.

1) $P = \{p_1, p_2, \ldots, p_m\} = P_{MP} \cup P_{FP} \cup P_{CP} \cup P_{null}$ *such that* $P_{MP} \cap P_{FP} \cap PP_{CP} \cap P_{null} = \Phi$. *P is a finite set of places representing the set of a) the possibly nonchangeable process objects, which consists of the subsets of: M_Process, F_Process, or C_Process objects on each, and b) the null objects which assist only the structural need such as process start/end or branching.*

2) $T = \{t_1, t_2, \ldots, t_n\} \cup \{t_{reset}\}$ *is a finite set of transitions denoting the process flow and transition logic between the corresponding input and output process places. The transition* t_{reset} *models the resetting of a process flow for the next process to start with.*

3) $F = (P \times T) \cup (T \times P)$ *is a set of arcs denoting a input and output process flow connections between* $P \rightarrow T$ *and* $T \rightarrow P$.

4) A_P *is a finite set of attributes associated with process places P, which model the related objects such as RESOURCE, METHOD, FEATURE on each subset of P, correspondingly. It allows 'Set-Of' objects to model alternatives as well as a single object.*

5) A_T *is a finite set of attributes associated with transitions T, which model the related process transition objects.*

6) *d is a finite set of times denoting the duration of a process place* $p \in P$.

7) $f_p : P \rightarrow A_P$ *is a mapping function from process places P to process place attributes* A_P.

8) $f_T : P \rightarrow A_T$ *is a mapping function from transitions T to transition attributes* A_T.

9) $f_d : P \rightarrow d$ *is a mapping function from process places P to time duration d*

10) $M_0 : P \rightarrow \{0, 1\}$ *is an initial marking of the Petri net.*

In the *SP³N*, a token in a place indicates that the current state of part-producing progress (*process_status*) is in the process object associated with the place. In an initial marking of the *SP³N*, there exist tokens in output places connecting the transition t_{reset}, which mean the possibly initial process objects. In the *SP³N*, the places are enabled in a sequential manner such that either a process place or a null place is enabled. A process place may be associated with the place attribute of multiple resources or methods (*resource or method conflicts*). When a specified time *d* of an enabled place *p* has been passed, output transitions of *p* are being candidated for firing immediately. There can be multiple transitions which are fired possibly (*process or sequence conflicts*). Then one and only one transition among them is fired. One process flow cycle is completed after t_{reset} is fired. A process state in any time can be represented by enabling places and their duration times. Each of the alternative process flows in a flexible process plan is a subnet of the *SP³N*, in which conflict resolution results (*process plan alternatives*).

The static process plan model *SP³N* has the following properties:

- *In the given SP³N, one and only one process place can be enabled at any time. The property comes from the constraints: a single part can be processed one at a time.*
- *In the SP³N, there are explicitly two types of conflicts (for firing): a) process sequence conflicts for competing process sequences and b) process conflicts for competing processes.*
- *Given the SP³N, the subnet SN* $\subseteq$ *SP³N representing one static process plan in which conflict resolution results in, is explicitly conflict-free in a set of processes and persistent.*
- *The static representation SP³N & the subnet SN of a process plan are live and safe.*

Proofs for the *SP³N* properties are not on the central focus and therefore omitted in this paper. The properties listed above guarantee that a) the *SP³N* can model a flexible process plan flow, *b)* one static process plan *SN* can be derived by resolving conflicts of the *SP³N*, c) the plan will not deadlock from the liveness property, and *d)* all process objects in *SN* are unique from the safeness property. The *SP³N* therefore provides the sufficient framework for representing a flexible process plan in the view of a process flow.

Example: The static process plan *SP³N* is shown Figure 5 for the given part in Figure 3. which is a static process plan on cutting tool level. The *SP³N* represents 72 alternative process plans ($2 \times 3 \times 2 \times 2 \times 3$ alternative transitions) and 2 alternative process sequences (2 sequences in *AND* transitions), in an explicit manner. Each of the alternative process plans is uniquely defined. Each process place $p \in P_{CP}$ of the *SP³N* is attached to an

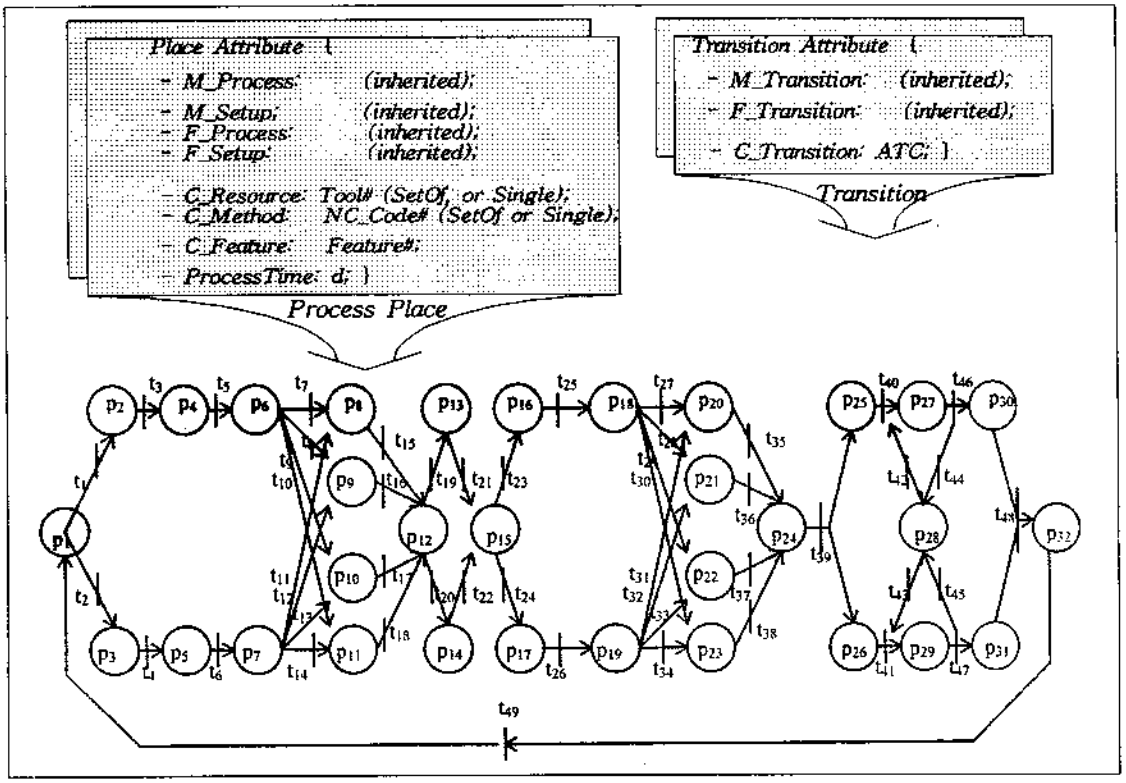


Figure 5. Petri net structure of static process plan

associated place attrribute P_A, which contains *a)* data about the *M_Process* and *F_Process* inherited from the superclasses, *b)* data about *C_Resource* (cutting tool) and *C_Method* from associated objects, and c) process time *a*. Each transition $t \in T$ of the SP^3N is also attached to an associated transition attribute T_A, which contains data about *a) M_Transition* and *F_Transition* inherited from the superclass and *b) C_Transition*.

3.3 Dynamic Model of Flexible Process Plan

A dynamic model of a flexible process plan represents the possibly nonchangeable process activities which should be detailed for shop floor operational scheduling. Information requirements of the dynamic process plan include that *a)* all process activities should be specified in the resource level and *b)* the real-time status of a system, resources and processes should be represented implicitly or explicitly. Therefore, the dynamic process plan model may be described as a process/machine interaction model which is capable of representing the real-time status of a system consisting of resources and processes. The static process plan model explained in the previous section, can be stepwise transformed to the dynamic process plan model. The dynamic process plan model is formally described in the form of a place-timed Petri net with relevant objects, as listed in the following:

Definition 3: The dynamic model of a flexible process plan for shop floor operational scheduling, DP^3N (Dynamic model of Process Plan Petri Net), is formally described as a place-timed Petri net with objects with the ten tuples as listed below:

$DP^3N = \langle P, T, F, A_P, A_T, d, f_P, f_T, f_d, M_0 \rangle$

where the following applies:

1) $P=\{p_1, p_2, \ldots, p_m\} = P_{PR} \cup P_{AT} \cup P_{RS} \cup P_{null}$ *such that* $P_{PR} \cap P_{AT} \cap PP_{RS} \cap P_{null} = \Phi$. *P is a finite set of places representing the respective subset of a) the possibly nonchangeable process_state places, b) the process_activity places, c) resource places, and d) null places for the structural need of process start/ end or branching.*

2) $T = \{t_1, t_2, \ldots, t_n\} \cup \{t_{reset}\} = T_{FLOW} \cup \{t_{reset}\}$ *such that* $T_{FLOW} \cap \{t_{reset}\} = \Phi$. *T is a finite set of transitions denoting a) the process flow transition logic and b) reset, correspondingly.*

3) $F = (P \times T) \cup (T \times P)$ *is a set of arcs denoting input and output connections of process flows and activities between* $P \rightarrow T$ *and* $T \rightarrow P$.

4) $A_P = A_{PR} \cup A_{RS}$ *is a finite set of place attributes associated with places* P_{PR} *and* P_{RS}, *correspondingly.*

5) $A_T = A_{FLOW} \cup A_{reset}$ *is a finite set of transition attributes associated with transitions* T_{FLOW} *and* $\{t_{reset}\}$, *correspondingly.*

6) *d is a finite set of times denoting the duration of a timed-place* $p \in P_{AT}$.

7) $f_P : P \rightarrow A_P$ *is a mapping function from places P to place attributes* A_P.

8) $f_T : P \rightarrow A_T$ *is a mapping function from transitions T to transition attributes* A_T

9) $f_d : P \rightarrow d$ *is a mapping function from timed places* P_{AT} *to time duration d.*

10) $M_0 : P \rightarrow \{0, 1\}$ *is an initial marking of the Petri net.*

The dynamic process plan DP^3N, as defined above, is place-timed. In the DP^3N, a token in a place $p \in P$ indicates {the state of a process (*process_status*), the activity status (*activity_status*), the state of resources (*resource_status*)}. In the initial marking of the DP^3N, tokens are positioned in the corresponding places representing an initial state of processes, activities and resources, which are a subset of output places $OP(t_{reset})$ of the reset transition. From the initial marking of P, a single process place p may be enabled in the sequential manner, provided that at least one input resource place $p \in P_{RS}$ & $p \in IP(OT(p))$ has a token. It is possible for the transition to be linked with multiple resource places in tokens (*resource conflict*). Among them a token in only one resource place may be transited after time duration. The system state at any time can be represented by {*process_status, activity_status, resource_status*}. Each of alternative process activities with resources of a flexible process plan is a subnet of the DP^3N, in which conflict resolution results (*operational schedule alternatives*).

The dynamic process plan model DP^3N has the following properties:

- *One and only one process activity place* $p \in P_{ACT}$ *can be enabled at any time from the constraints: a single part can be processed one at a time.*
- *Given the* DP^3N, *there exist three types of conflicts (for firing): a) process conflicts for competing processes, b) sequence conflicts for competing process sequences, and c) resource conflicts for competing resources available.*
- *Given the* DP^3N, *the subnet* $DN \subseteq SP^3N$ *representing one dynamic process plan in which conflict resolution results in, is explicitly conflict-free in a set of* ⟨*process activity, resource*⟩*'s and persistent.*
- *The dynamic process plan* $DN \subseteq DP^3N$ *representing process activities and resources to be performed, is an operational schedule of the part.*
- *The dynamic representation* DP^3N *& the subnet DN of a flexible process plan are live and safe.*

Proofs for the DP^3N properties are not on the main focus and therefore omitted in this paper. The properties listed above guarantee that the DP^3N can support shop floor operational scheduling. One dynamic process plan DN can be derived by resolving conflicts specified in the DP^3N definition. The DN does not deadlock from the liveness property. The DN is unique from the safeness property. The DP^3N therefore provides the sufficient

framework for representing a flexible process plan in the view of process activities with resources.

type provided that there exist multiple machines for a *M_Process*.

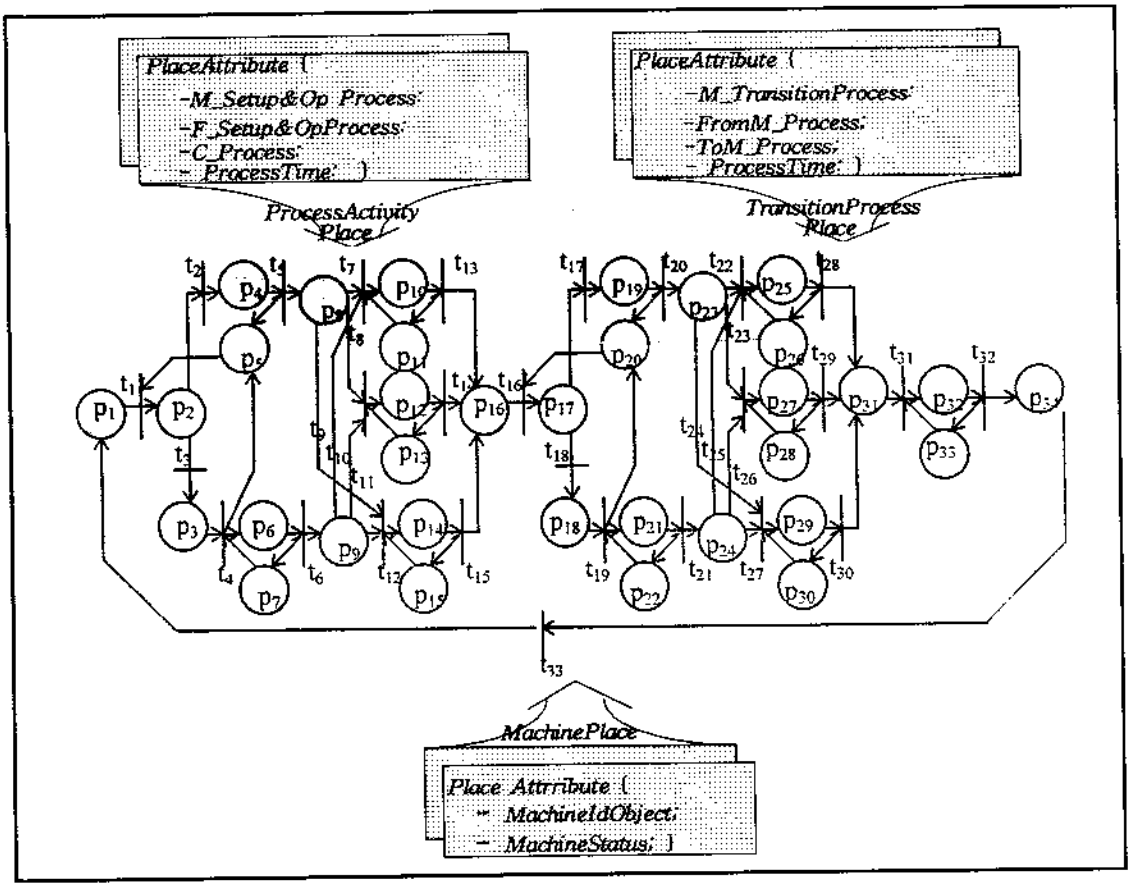


Figure 6. Petri net structure of dynamic process plan

Example: Figure 6 shows the DP^3N for the given part in Figure 3, which is a place-timed Petri net on machine level. The DP^3N may be further refined to the fixture level DP^3N provided that fixtures should be modeled actively. In the figure, the DP^3N represents 72 alternative dynamic process plans and 2 process sequences with four types of machines, that is: *Vertical_Milling_Center* (*VM*), *Drilling_ Machine* (*DR*), *Boring_Machine* (*BR*), and *Grinding_Machine* (*GR*). It is possible to represent alternative machines by modeling multiple machines with an identical

4. Application

In this section, the multi-faceted process plan model is demonstrated to show the validity of the proposed model for a flexible manufacturing environment in detail manner.

4.1 Manufacturing Environment

For repetitively producing the part given in Figure 3, a manufacturing system consists of four types of machines as followed: *drilling machine (DR), boring machine (BR),*

grinding machine (GR) and vertical milling machine (VM), as shown in Figure 7. *VM* may perform both milling and drilling processes. They are linked by *a synchronous conveyor (CV) (It means that there exist no sharing problem in transportation)*. There are three types of fixtures in the manufacturing system, that is: *Vise (VF), Plate (PF), and Jig (JG)*, for cutting operation processes on each machine. All setup processes of changing fixtures and tools are assumed to be internally done by *AWC* (Automatic Work Changer) and *ATC* (Automatic Tool Changer) without external processing. The manufacturing system processes parts in a batch production manner and the sequence of processes for a part depends on its routeing. In the flexible manufacturing environment parts may be alternatively processed in multiple routes.

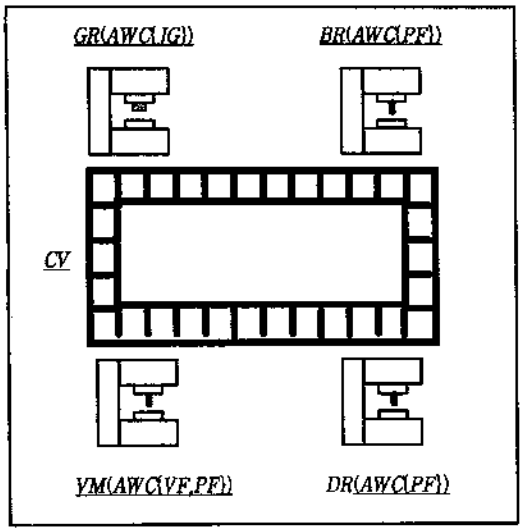


Figure 7. Manufacturing system

4.2 Modeling Results of OP^2N, SP^3N and DP^3N

For the given part and the manufacturing environment, Figure 8 shows all the process objects in the OP^2N in comprehensive detail. The corresponding SP^3N and DP^3N are consecutively shown in Figure 9 and 10.

As shown in Figure 3, the prismatic part is composed of 8 manufacturing features, that is: {3 *faces*, 2 *patterned holes*, *1 deep hole*, *1 pocket*, *1 slot features*}.

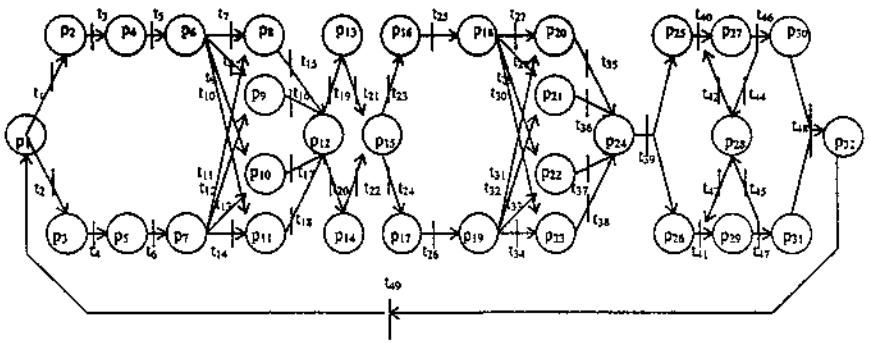
As shown in Figure 8, the whole machining processes for producing the part were decomposed into the nondecomposable process objects of the manufacturing system, as followed: {6 *vertical milling processes*, *4 drilling processes*, *2 boring processes*, *1 grinding processes; 7 vise fixture processes*, *7 plate fixture processes*, *3 jig fixture processes; 26 c-level operation process*}, in a hierarchical structure. From the modeling result of the OP^2N, all the processes on each level were identified in an object oriented structure, in which they can be modeled with the modular/alternative processes for explicitly supporting the possible process change of the flexible manufacturing environment.

In Figure 9, the process objects of the OP^2N were modeled as a process flow model (SP^3N) with the timed Petri net structure correspondingly, in which they can be sufficiently modeled within a Petri net structure with operation process objects attached to places and transition process objects to transitions. It is possible to transform from the OP^2N to the SP^3N according to mapping relations between the two models. From the modeling result of the SP^3N, the process flow may be flexibly described in complete manner such that it includes transition processes as well as operation processes for producing the part. The SP^3N can support the formal analysis of the process flow problem such as process plan selection or re-planning for the flexible manufacturing environment.

In Figure 10, all the process activities and machines were modeled as a process/ machine interaction model (DP^3N) with an object-oriented timed Petri net. As shown in Figure 10, the DP^3N may model the transition processes (for example, p_2), the competing processes (for example, $\{p_3, p_6\}$) with different types of machines (for example, $\{p_3, p_7\}$), and the resource sharing (for example, *VM*: $\{p_3, p_{11}, p_{29}, p_{36}\}$). From the modeling result of the DP^3N, therefore, it can support the formal analysis about the

```
/* Process Objects of Machine Level Operation:
   M_ProcessObject (ProcessId, ProcessType[Resouce,Method,Feature], SubProcesses)*/

   M_OpProcess (MP#1, VerticalMillingProcess[VM#, - , MF#1], FP#1);
   M_OpProcess (MP#2, - , OR{MP#21, MP#22});
   M_OpProcess (MP#21, VerticalMillingProcess[VM#, - , MF#21], FP#21);
   M_OpProcess (MP#22, DrillingProcess[DR#, -, MF#22], FP#22);
   M_OpProcess (MP#3, - , OR{MP#31, MP#32, MP#33});
   M_OpProcess (MP#31, VerticalMillingProcess[VM#, -, MF#31], OR{FP#311,FP#312});
   M_OpProcess (MP#32, DrillingProcess[DR#, -, MF#32], FP#32);
   M_OpProcess (MP#33, BoringProcess[BR#, -, MF#33], FP#33);
   M_OpProcess (MP#4, VerticalMillingProcess[VM#, - , MF#4], SEQ<FP#41, FP#42>);
   M_OpProcess (MP#5, - , OR{MP#51, MP#52});
   M_OpProcess (MP#51, VerticalMillingProcess[VM#, -, MF#51], FP#51);
   M_OpProcess (MP#52, DrillingProcess[VM#, -, MF#52], FP#52);
   M_OpProcess (MP#6, - , OR{MP#61, MP#62, MP#63});
   M_OpProcess (MP#61, VerticalMillingProcess[VM#, -, MF#61], OR{FP#611,FP#612});
   M_OpProcess (MP#62, DrillingProcess[DR#, -, MF#62], FP#62);
   M_OpProcess (MP#63, BoringProcess[BR#, -, MF#63], FP#63);
   M_OpProcess (MP#7, GrindingProcess[GR#, - , MF#6], SEQ<FP#71, AND{FP#72, FP#73}>);

/* Process Objects of Fixture Level Operation:
   F_ProcessObject (ProcessId, ProcessType[Resouce,Method,Feature], SubProcesses)*/

   F_OpProcess (FP#1, ViseFixtureProcess[VISE#, FM#1, FF#1], SEQ<CP#11, OR{CP#12, CP#13}>);
   F_OpProcess (FP#21, ViseFixtureProcess[VISE#, FM#21, FF#21], SEQ{CP#211, CP#212}>);
   F_OpProcess (FP#22, PlateFixtureProcess[PLATE#, FM#22, FF#22], SEQ(CP#221, CP#222));
   F_OpProcess (FP#311, ViseFixtureProcess[VISE#, FM#311, FF#311], CP#311);
   F_OpProcess (FP#312, PlateFixtureProcess[VISE#, FM#312, FF#312], CP#312);
   F_OpProcess (FP#32, PlateFixtureProcess[PLATE#, FM#32, FF#32], CP#32);
   F_OpProcess (FP#33, PlateFixtureProcess[PLATE#, FM#33, FF#33], CP#33);
   F_OpProcess (FP#41, ViseFixtureProcess[VISE#, FM#41, FF#41], SEQ<CP#411, OR{CP#412, CP#413}>);
   F_OpProcess (FP#42, ViseFixtureProcess[VISE#, FM#42, FF#42], CP#42);
   F_OpProcess (FP#51, ViseFixtureProcess[VISE#, FM#51, FF#51], SEQ<CP#511, CP#512>);
   F_OpProcess (FP#52, PlateFixtureProcess[PLATE#, FM#52, FF#52], SEQ<CP#521, CP#522>);
   F_OpProcess (FP#611, ViseFixtureProcess[VISE#, FM#611, FF#611], CP#611);
   F_OpProcess (FP#612, PlateFixtureProcess[VISE#, FM#612, FF#612], CP#612);
   F_OpProcess (FP#62, PlateFixtureProcess[PLATE#, FM#62, FF#62], CP#62);
   F_OpProcess (FP#63, PlateFixtureProcess[PLATE#, FM#63, FF#63], CP#63);
   F_OpProcess (FP#71, JigFixtureProcess[JIG#, FM#71, FF#72], CP#71);
   F_OpProcess (FP#72, JigFixtureProcess[JIG#, FM#72, FF#72], CP#72);
   F_OpProcess (FP#73, JigFixtureProcess[JIG#, FM#73, FF#73], CP#73);

/* Process Objects of Cutting Level Operation:
   C_ProcessObject (ProcessId, ProcessType[Resouce,Method,Feature])*/

   C_OpProcess (CP#11, FacingProcess[FACEMILL#, CM#11, CF#11]);
   C_OpProcess (CP#12, E_PocketingProcess[ENDMILL#, CM#12, CF#12]);
   C_OpProcess (CP#13, P_PocketingProcess[ENDMILL#, CM#11, CF#11]);
   C_OpProcess (CP#211, CounterDrillingProcess[C_DRILL#, CM#211, CF#211]);
   C_OpProcess (CP#212, TwistDrillingProcess[T_DRILL#, CM#212, CF#212]);
   C_OpProcess (CP#221, CounterDrillingProcess[C_DRILL#, CM#221, CF#221]);
   C_OpProcess (CP#222, TwistDrillingProcess[T_DRILL#, CM#222, CF#222]);
   C_OpProcess (CP#311, SpadeDrillingProcess[S_DRILL#, CM#311, CF#311]);
   C_OpProcess (CP#312, MillingProcess[END_MILL#, CM#312, CF#312]);
   C_OpProcess (CP#32, SpadeDrillingProcess[S_DRILL#, CM#32, CF#32]);
   C_OpProcess (CP#33, BoringProcess[BORE_TOOL#, CM#33, CF#33]);
   C_OpProcess (CP#411, FacingProcess[FACEMILL#, CM#411, CF#411]);
   C_OpProcess (CP#412, E_SlotingProcess[ENDMILL#, CM#412, CF#412]);
   C_OpProcess (CP#413, P_SlotingProcess[ENDMILL#, CM#413, CF#413]);
   C_OpProcess (CP#42, FacingProcess[FACEMILL#, CM#42, CF#42]);
   C_OpProcess (CP#511, CounterDrillingProcess[C_DRILL#, CM#511, CF#511]);
   C_OpProcess (CP#512, TwistDrillingProcess[T_DRILL#, CM#512, CF#512]);
   C_OpProcess (CP#521, CounterDrillingProcess[C_DRILL#, CM#521, CF#521]);
   C_OpProcess (CP#522, TwistDrillingProcess[T_DRILL#, CM#522, CF#522]);
   C_OpProcess (CP#611, SpadeDrillingProcess[S_DRILL#, CM#611, CF#611]);
   C_OpProcess (CP#612, MillingProcess[END_MILL#, CM#612, CF#612]);
   C_OpProcess (CP#62, BoringProcess[BORTOOL#, CM#62, CF#62]);
   C_OpProcess (CP#63, SpadeDrillingProcess[S_DRILL#, CM#33, CF#33]);
   C_OpProcess (CP#71, GrindingProcess[GRINDTOOL#, CM#71, CF#71]);
   C_OpProcess (CP#72, GrindingProcess[GRINDTOOL#, CM#72, CF#72]);
   C_OpProcess (CP#73, GrindingProcess[GRINDTOOL#, CM#73, CF#73]);
```

Figure 8. Object process plan



place list:

place	place_attribute(meaning)
p_1	MP#1→FP#1→CP#11
p_2	MP#1→FP#1→CP#12
p_3	MP#1→FP#1→CP#13
p_4	MP#21→FP#21→CP#211
p_5	MP#22→FP#22→CP#221
p_6	MP#21→FP#21→CP#212
p_7	MP#22→FP#22→CP#222
p_8	MP#31→FP#311→CP#311
p_9	MP#31→FP#312→CP#312
p_{10}	MP#32→FP#32→CP#32
p_{11}	MP#33→FP#33→CP#33
p_{12}	MP#4→FP#41→CP#411
p_{13}	MP#4→FP#41→CP#412
p_{14}	MP#4→FP#41→CP#413
p_{15}	MP#4→FP#42→CP#42
p_{16}	MP#51→FP#51→CP#511
p_{17}	MP#52→FP#52→CP#521
p_{18}	MP#51→FP#51→CP#512
p_{19}	MP#52→FP#52→CP#522
p_{20}	MP#61→FP#611→CP#611
p_{21}	MP#61→FP#612→CP#612
p_{22}	MP#62→FP#62→CP#62
p_{23}	MP#63→FP#63→CP#63
p_{24}	MP#7→FP#71→CP#71
p_{25}	null(for ASYNCHRONY)
p_{26}	null(for ASYNCHRONY)
p_{27}	MP#7→FP#72→CP#72
p_{28}	null(for ASYNCHRONY)
p_{29}	MP#7→FP#73→CP#73
p_{30}	null(for ASYNCHRONY)
p_{31}	null(for ASYNCHRONY)
p_{32}	null(for AND branching)

transition list:

transition	transition_attribute(meaning)
t_1	CT#1:CP#11→CP#12
t_2	CT#2:CP#11→CP#13
t_3	FT#1:FP#1→FP#21
t_4	MT#1:MP#1→MP#22
t_5	CT#2:CP#211→CP#212
t_6	CT#3:CP#221→CP#222
t_7	FT#2:FP#21→FP#311
t_8	FT#3:FP#21→FP#312
t_9	MT#2:MP#21→MP#32
t_{10}	MT#3:MP#21→MP#33
t_{11}	MT#4:MP#22→MP#31
t_{12}	MT#5:MP#22→MP#31
t_{13}	FT#4:FP#22→FP#32
t_{14}	MT#6:MP#22→MP#33
t_{15}	FT#5:FP#311→FP#41
t_{16}	FT#6:FP#312→FP#41
t_{17}	MT#7:MP#32→MP#4
t_{18}	MT#8:MP#33→MP#4
t_{19}	CT#4:CP#411→CP#412
t_{20}	CT#5:CP#411→CP#413
t_{21}	FT#7:FP#41→FP#42
t_{22}	FT#8:FP#41→FP#42
t_{23}	FT#9:FP#42→FP#51
t_{24}	MT#9:MP#4→MP#52
t_{25}	CT#6:CP#511→CP#512
t_{26}	CT#7:CP#521→CP#522
t_{27}	FT#10:FP#51→FP#611
t_{28}	FT#11:FP#51→FP#612
t_{29}	MT#10:MP#51→MP#62
t_{30}	MT#11:MP#51→MP#63
t_{31}	MT#12:MP#52→MP#61
t_{32}	MT#13:MP#52→MP#61
t_{33}	FT#12:FP#52→FP#62
t_{34}	MT#14:MP#52→MP#63
t_{35}	MT#15:MP#61→MP#7
t_{36}	MT#16:MP#61→MP#7
t_{37}	MT#17:MP#62→MP#7
t_{38}	MT#18:MP#63→MP#7
t_{39}	null(for AND branching)
t_{40}	FT#13:FP#71→FP#72
t_{41}	FT#14:FP#71→FP#73
t_{42}	FT#15: FP#73→FP#72
t_{43}	FT#16: FP#72→FP#73
t_{44}	null
t_{45}	null
t_{46}	null
t_{47}	null
t_{48}	reset

* MT: Machine-Level Transition
FT: Fixture-Level Transition
CT: Cutting -Level Transition

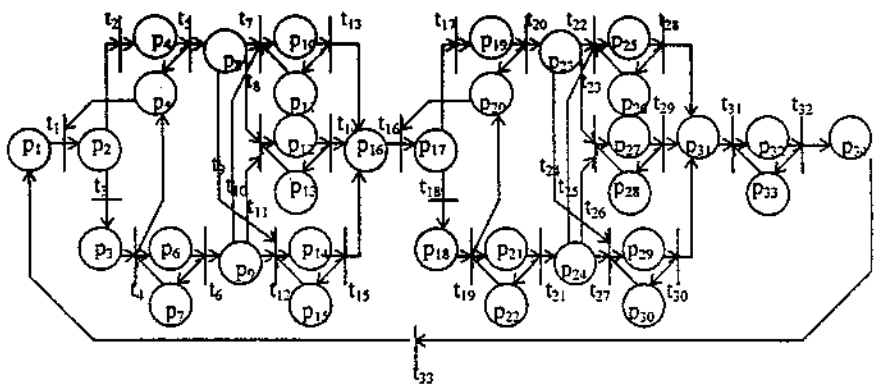
Figure 9. Static process plan

process activity problem such as operational scheduling, dispatching, or rescheduling for the flexible manufacturing environment.

Therefore, the multi-faceted process plan model proposed here is capable of providing an integrated formal basis of a flexible process plan for shop floor control.

5. Concluding Remarks

An object-oriented Petri net model for the representation of a flexible process plan, in this paper, was proposed for the integration approach of a process plan, scheduling and shop floor control. In the representation framework, the process plan model was represented in a multi-faceted structure: *a)* an object model, *b)* a static model and *c)* a dynamic model. The multi-faceted process plan can model



place list:

place	place_attribute(meaning)	place	place_attribute(meaning)
p_1	ProcessState(for Process Start)	p_{18}	MachineTransition[MT#9:VM→DR]
p_2	ProcessAct [FP#1→SEQ<CP#11,OR{CP#12,CP#13}]	p_{19}	ProcessAct [FP#51→SEQ{CP#511,CP#512}]
p_3	MachineTransition[MT#1:VM→DR]	p_{20}	Machine [VM]
p_4	ProcessAct [FP#21→SEQ{CP#211,CP#212}]	p_{21}	ProcessAct [FP#52→SEQ{CP#521,CP#522}]
p_5	Machine[VM]	p_{22}	Machine [DR]
p_6	ProcessAct [FP#22→SEQ{CP#221,CP#222}]	p_{23}	MachineTransition[MT#10:VM→DR] [MT#11:VM→BR]
p_7	Machine[DR]	p_{24}	MachineTransition[MT#12:DR→VM] [MT#14:DR→BR]
p_8	MachineTransition[MT#2:VM→DR][MT#3:VM→BR]	p_{25}	ProcessAct [SEQ{FP#61→ CP#611, FP#612→ CP#612}]
p_9	MachineTransition[MT#4:DR→VM] [MT#6:DR→BR]	p_{26}	Machine[VM]
p_{10}	ProcessAct [SEQ{FP#311→ CP#311, FP#312→ CP#312}]	p_{27}	ProcessAct [FP#62→ CP#62]
p_{11}	Machine[VM]	p_{28}	Machine[DR]
p_{12}	ProcessAct [FP#32→ CP#32]	p_{29}	ProcessAct [FP#63→ CP#63]
p_{13}	Machine[DR]	p_{30}	Machine[BR]
p_{14}	ProcessAct [FP#33→ CP#33]	p_{31}	MachineTransition[MT#15:VM→GR][MT#17:DR→GR] [MT#18:BR→GR]
p_{15}	Machine[BR]	p_{32}	ProcessAct [SEQ{FP#71→ CP#71, OR{FP#72→ CP#72, FP#73→ CP#73}]
p_{16}	MachineTransition[MT#7:DR→VM][MT#8:BR→VM]	p_{33}	Machine[GR]
p_{17}	ProcessAct[FP#41→SEQ<CP#411,OR{CP#412,CP#413}]	p_{34}	ProcessState(for Process End)

Identical places: {p_5, p_{11}, p_{20}, p_{26}}/{p_7, p_{13}, p_{22}, p_{28}}/{p_{15}, p_{30}}

Figure 10. Dynamic process plan

ar alternative process plan and then support planning, scheduling, and shop floor control, in a unified formal model. Petri nets with objects were selected for the representation of the process plan model because it is generically capable of modeling process flows and process activities needed for the multi-faceted process plan model. The representation model has benefits of *a)* performing planning, scheduling and shop floor control functions in a unified formal model and *b)* formally transforming respective models from an object model to a dynamic model through a static model. In this paper, the proposed process plan model was demonstrated and validated with an illustrative example.

We will extend this research to the development of *a)* the model generation procedure of the multi-faceted process plan, *b)* rapid prototyping from a flexible process plan to shop floor control and c) the *CASE* (Computer-Aided Software Engineering) tool for an integrated system of planning, scheduling and shop floor control, respectively.

References

[1] Carton, B.A. and Ray, S.R., "ALPS: A Language for Process Specification", *International Journal of Computer Integrated Manufacturing*, Vol.4, No.2, 105-113, 1991

[2] Chang,T.C., *Expert Process Planning for Manufacturing*, Addison-wesley, 1990

[3] Chang,T.C. and Wysk,R.A., *An Introduction to Automated Process Planning*, Prentice-Hall, 1985

[4] Cho, H.B., Derebail. A., Hale, T., and Wysk, R.A., "A Formal Approach to Integrating Computer Aided Process Planning and Shop Floor Control", *ASME: Journal of Engineering for Industry*, Vol.116, No.1, 108-116, 1994

[5] Cho, H.B., "An Intelligent Workstation Controller for Computer Integrated Manufacturing", *Ph.D. Thesis*, Texas A&M University, 1993

[6] Hayes, C.C., "P^3: A Process Planner for Manufacturability Analysis", *IEEE. Transactions On Robotics and Automation*, Vol. 12, No. 2, 220-234, 1996

[7] Huang, S.H., Zhang, H.C., and Smith, M.L., "A Progressive Approach for the Integration of Process Planning and Scheduling", *IIE Transactions*, Vol. 27 456-464, 1995

[8] Joshi,S.B. and Smith,J.S., *Computer Control of Flexible Manufacturing Systems*, Chapman & Hall, 1994

[9] Jung, M.Y. and Lee, K.H., "A Conceptual Framework of Automated Process Planning", *International Journal of Computers and Industrial Engineering*, Vol. 34, No. 1-4, 1995

[10] Kruth, J.P. and Detand. J., "A CAPP System for Nonlinear Process Plans", *Annals of the CIRP*, Vol. 41, No. 1, 489-492, 1992

[11] Larson, N.E. and Alting, L., "Dynamic Planning Enriches Concurrent Process and Production Planning", *International Journal of Production Research*, Vol. 30, No. 8, 1861-1876, 1992

[12] Lee, K.H., "Formal Integration of Process Planning and Shop Floor Control for CIM", *Ph.D. Dissertation*. POSTECH, Korea, 1995

[13] Lee, K.H., "Formal Integration of Process Planning and Shop Floor Control for CIM", *Proceedings of the Korean Institute of Industrial Engineers Conference*, 1996

[14] Lee, K.H. and Kim, B.K., "An Integrated Approach of Process Plan Selection and Scheduling", *Journal of the Society of Korea Industrial and Systems Engineering*, Vol. 19, No. 39, 1-8, 1996

[15] Lee, K.H. and Jung, M.Y., "Flexible Process Sequencing Using Petri Net Theory", *International Journal of Computers and Industrial Engineering*, Vol. 35, No. 2, 279-290, 1995

[16] Lee, K.H. and Jung, M.Y., "Petri Net Application for Flexible Process Planning", *International Journal of Computers and Industrial Engineering*, Vol. 34, No. 1-4, 1995

[17] Lenderink and Kals, "The Integration of Process Planning and Machine Loading in Small Batch Part Manufacturing", *Robotics and Computer Integrated Manufacturing*, Vol. 10, No. 1-2, 89-98, 1993

[18] Mettala, E.G. and Joshi, S., "A Compact Representation of Alternative Process Plans/Routings for FMS Control Activities", *Journal of Design and Manufacturing*, Vol. 3, 91-104, 1993

[19] Thomas, J.P., Nissanke. N, and Baker, K.D., "A Hierarchical Petri Net Framework for the Representation and Analysis of Assembly", *IEEE. Transactions On Robotics and Automation*, Vol. 12, No. 2, 268-279, 1996

[20] Tonshoff, H.K., Beckendorff, and Anders, N., "FLEXPLAN-A Concept for Intelligent Process Planning and Scheduling", *CIRP International Workshop on Computer Aided Process Planning*, 1989

[21] Zhou, M.C., DiCesare. F, and Desrochers, A.A., "A Hybrid Methodology for Synthesis of Petri Nets for

Manufacturing Systems", *IEEE. Transactions On Robotics and Automation*, Vol. 8, No. 3, 350-361, 1992

97년 3월 최초 접수, 97년 10월 최종 수정

Appendix. The Definition of Petri Nets

A Petri net, also known as a place-transition net (*PTN*), is a formal graph model for the description and analysis of systems that exhibit both asynchronous and concurrent properties. They provide a natural way to represent and analyze the systems such as *FMSs* with such properties. Petri nets have evolved into a powerful tool for representing and analyzing asynchronous concurrent systems.

In this section, ordinary- Petri nets, *OTN*, and timed- Petri nets, *TPN*, are defined briefly.

Definition A1. An ordinary- Petri net, O*PN*, is fomally described as the following five tuples:

$OPN = (P, T, I, O, M)$

where
1) $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places drawn as circles;
2) $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions drawn as bars;
3) I: $(P \times T)$ is an input function from P to T drawn as directed arcs;
4) O: $(T \times P)$ is an output function from T to P drawn as directed arcs;
5) M ia a marking vector to represent the state of the system, drawn as black dots in places;

Definition A2. A timed- Petri net, *TPN*, is fomally described as the following six tuples:

$TPN = (P, T, I, O, M, \rho)$

where
1) $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places drawn as circles;
2) $T = \{t_1, t_2, ..., t_n\}$ is a finite set of transitions drawn as bars;
3) I: $(P \times T)$ is an input function from P to T drawn as directed arcs;
4) O: $(T \times P)$ is an output function from T to P drawn as directed arcs;
5) M ia a marking vector to represent the state of the system, drawn as black dots in places;
6) ρ: $P \rightarrow t_d$ is a mapping function from places P to time domain t_d;

In place-timed Petri nets, an execution time t_d is associated with each place $p \in P$. When a timed place initiates its firing, it takes t_d units of time to complete its firing.