International Journal of Management Science
Vol 3, No 1, May 1997

A PETRI NET-BASED CELL CONTROLLER FOR A FLEXIBLE MANUFACTURING SYSTEM

GERRIT K. JANSSENS
Computer Science and Operations Management,
University of Antwerp(RUCA), Belgium

MARIO T. TABUCANON
School of Advanced Technologies,
Asian Institute of Technology, Thailand

ABSTRACT

In a flexible manufacturing system, a cell controller is able to identify and evaluate a number of alternative decisions to meet the objectives set by the factory level controller. In this paper, a Petri net-based cell controller is presented to accomplish this task. A static model is developed by using the Integrated Computer Aided Definition(IDEF0) method to represent clear functional relationships among the objects of the system. Based on the static model, two Petri net models are developed for the physical part flow and for the information flow. Multiple decision alternatives are generated from the physical part flow model and are synchronized with the information flow model for execution of the selected alternative.

1. INTRODUCTION

A flexible manufacturing cell is a single part of a Flexible Manufacturing System (FMS) integrated with material handling devices and with numerically controlled machine tools. It simultaneously processes a product mix of a variety of part types. Functionally, the cell controller is responsible for the control of the machines and for the routing of the workpieces within the cell. The flexibility requirements in manufacturing small batches with a rapidly varying product mix have stressed the contribution of the features of concurrency and synchronization in the overall performance of the process [2]. By introducing these features, flexible manufacturing cells pose new directions in decision making and control of manufacturing systems.

The large number of variables characterizing the decision process in an FMS environment and the intricate nature of the mutual effects of the decision process requires a hierarchical level of decision making [1]. Although several hierarchical control and scheduling techniques have been proposed, very few define in detail the decision making and control processes at the cell level.

The main objective of this paper is to develop an efficient decision structure and control mechanism for real time implementation of asynchronous concurrent activities of an FMS at the cell level. The problem considered focuses the general context of discrete manufacturing confining to the cell's operations and its related information flow, primarily pointing towards a product mix of different part types.

2. BACKGROUND

Several researchers have pioneered in the design of cell controllers(e.g. TU and Sorgen[14], O'Grady et al.[11], Lin and Solberg[9], and Favrel et al.[7].) Few attempts have been done to integrate the physical and informational flows in an FMS(e.g. Boucher and Jafari[3]). In a flexible manufacturing environment, all manufacturing functions are automated to a high degree to perform operations effectively and efficiently in each cell. But automation requires a greater flexibility, which implies a greater complexity in the control of the manufacturing functions. Since the aim is to develop a cell controller in a flexible manufacturing environment, complexity becomes an inherent fact. In cell control operations complexities arise both from the physical part flow and the information flow modules. Consequently, cell controller design will require a simultaneous evolution of transactions with a considerable amount of parallelism and synchronization. This stresses the need for a comprehensive tool for efficient modelling of a cell controller. Petri nets being an excellent graphical modelling tool, supported by a well developed mathematical theory, is thought to be relevant for the cell controller design. This dynamic modelling technique will be supported by the IDEF0 technique(discussed in section 3). Moreover, a Petri net possesses many advantages as a dynamic modelling tool as is known from the literature and from applications in various industries. Some of the advantages of using Petri nets in the design of cell controllers are discussed below.

The FMS can be described in a graphical form allowing an easy visualization and communication of complex interactions between different components. Petri net based models also show those states of the system where

contention can occur between activities which share some resources and where control may be exercised. An important feature is that Petri net based models are executable, i.e. simulation code can be generated automatically from the net specification. Performance measures can be obtained by direct simulation of the net, without the need to write additional software. This feature is particularly attractive when it is necessary to emulate the behavior of the system [2]. Moreover, the system and its Petri net based models can be validated by using logical and mathematical properties.

A Petri net can be synthesized using both bottom-up and top-down approaches. Hence, it is possible to design a system, whose behavior is either known or easily verifiable, in a systematical way [16].

Petri nets can model in an exact way features such as, priorities, synchronization, forking, blocking and multiple resource holding [15]. On the other hand, the asynchronous nature of a Petri net makes it suitable for representation of real time systems. A Petri net also possesses inherent concurrency and parallelism and can be used as a hierarchical modelling technique. This allows us to represent a complex system through simpler models at various levels of abstraction. Reviews of application of Petri nets in a manufacturing environment appear in Janssens[8] and in DiCesare et al.[6].

3. SYSTEM DESCRIPTION

Flexible manufacturing can be viewed as a system with two macro states: physical and informational. From the physical point of view, an FMS consists of a set of machines, equipment, materials and products. The informational system comprises a set of logical processes controlled through programs for changing system states. Consequently, the complexity of the system is drastically increased due to evolutionary nature of the states.

In such a situation, integration of physical and informational system becomes much more complex and hence an extensive research for synchronization is required from the root level of manufacturing. The Petri net based cell controller receives a preliminary munufacturing plan for a product or a batch of products, performs an analysis, feedbacks all possible decision alternatives to the factory level controller and finally coordinates the implementation of the selected decision within the cell. As the cell controller is related both to an internal module (coordinated the physical tasks within the cell) and to an external module (request program from the factory controller), we consider a typical cell configuration in a flexible production environment

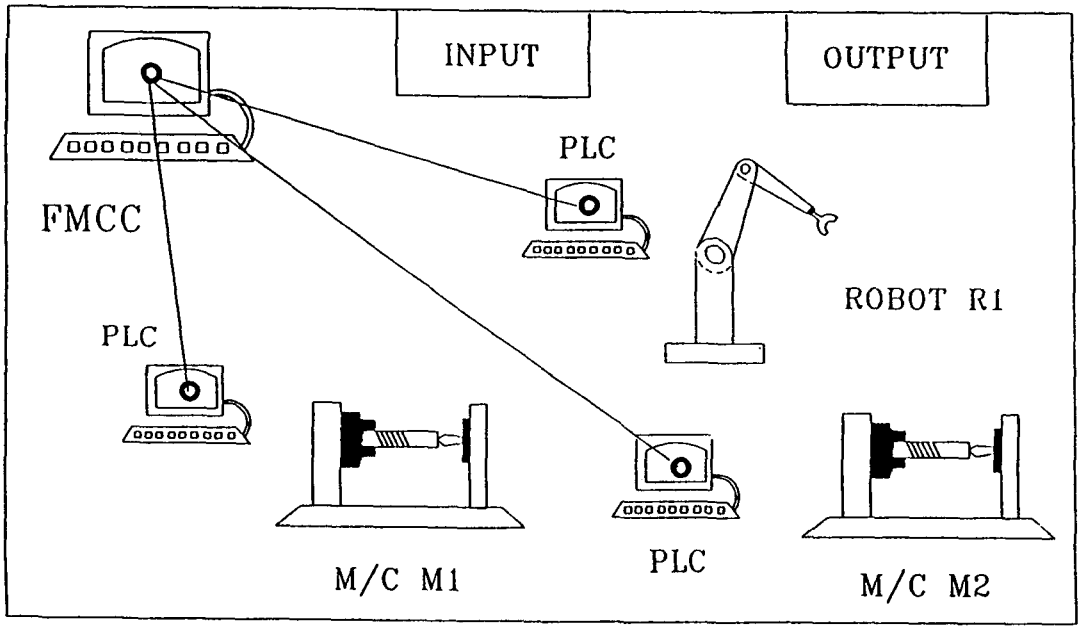

Figure 1. Flexible Machining Cell

shown in Figure 1.

4. NEED OF IDEF0 TECHNIQUE IN CELL CONTROLLER DESIGN

The literature on system analysis suggests that a high percentage of design errors in contemporary computer-based information systems can be supported by the information system. This suggests that one must try to determine a set of correct and complete set of requirements before embarking on the design of a system [4]. In controller design, the interrelationships of the objects (information, data, and material) needs to be simplified necessitating the need of a static modelling technique, IDEF0, a top-down methodology with graphical potrayal is suitable for a hierarchical decomposition of a complex munufacturing system of an FMS. Using the IDEF0 method, syntactical consistencies between design and specification can be checked and an automated support to dynamic modelling techniques for cell controller design is provided. A state of the art review on IDEF0 can be found in Colquhoun et al. [5].

5. PROBLEM FORMULATION

In a flexible manufacturing system, a cell controller requires sufficient intelligence to identify and evaluate a set of decision alternatives to meet the objectives set by the factory level controller. The objectives can be expressed as: throughput, minimum makespan and due dates. The system state changes due to the occurrence of certain events in real time. In a real life situation, changes of the states directly imply changes in the status of different system components. In this dynamic situation, cell controllers usually are semi-autonomous. They receive instructions from a higher level controller and then arrange activities within the cell to follow the instructions. In this system, all decisions are made by the factory level controller and programs are downloaded into the equipment controllers through the cell control system. This system is called a centralized production system. In other cases, the whole responsibility is given to the cell level controllers for decision making as well as for inter-cell communications. In this case, only the status of each cell is reported periodically to the factory level controller. The responsibility of the factory level controller comes in when there is a significant departure from the planned activities.

Both the above systems have their relative advantages and disadvantages in their respective mode of operations. Our aim is to develop a cell controller for an FMS providing decisive power to the cell controller for real time scheduling and for control of operations within the cell. It is required to build a control mechanism and a decision structure for effective utilization of available cell resources. The system is far from easy to build due to many interacting system components. This necessitates a systematic approach for the generation of decision alternatives in a manufaturing cell. The prime objective of the cell controller is to fulfill the objectives set by the factory level controller. Every alternative generated by the cell controller should provide a clear picture: at what time, which part will visit which machine for which operation, including the flow path of respective programs for execution of these operations.

With this knowledge on the features of manufacturing operations in an FMS, the need is identified for an efficient control mechanism and a decision structure for real time implementation of operations in the cell.

To investigate this problem, a typical flexible machining cell with two general purpose NC machines (M1 & M2), one programmable robot (R1), input/output buffers and a cell controller is considered (Figure 1). The factory controller is sending a batch of product number PROD-Q01 to the machining

cell with an objective to minimize the makespan for the assigned batch.

The batch consists of three part types- Part X, Part Y and Part Z. For simplicity it is assumed that no earlier work has been assigned to the cell and the ratio of the product mix is 1:1:1. For each part type the factory controller provides necessary time requirement for each of the operation along with program codes as shown in the Table 1. Figure 2 shows the operational constraints of each part. For example, part X cannot be processed on machine M2 until it finishes operations on machine M1. It is the responsibility of the cell controller to analyze which part will enter which machine at what time so the batch can be processed in minimum time. One additional system constraint has been imposed by the factory level controller. The part programs may not be available in the central database management system for loading to the equipment controllers at the time required (the model is proposed to operate in centralized program distribution). The cell controller has to look for an alternative which will be synchronized with the available part programs in the centralized database management system to build an executable integrated model to reach the specified goal.

Table 1. Activity Durations and Program Codes of Parts

Activity No	Activity description	Part Type X		Part Type Y		Part Type Z	
		Time	Prog. Code	Time	Prog. Code	Time	Prog. Code
1	Robot R1 Loading Part From I/B Buffer to M1	1	R1X01	2	R1Y01	*	*
2	M1 Processing a Part	3	M1X01	9	M1Y01	*	*
3	Robot R1 Unloading part From M1 and Loading M2	2	R1X01	*	*	*	*
4	Robot R1 Loading Part From I/B Buffer to M2	*	*	*	*	1	R1Z01
5	M2 Processing a Part	4	M2X01	*	*	6	M2Z01
6	Robot R1 Unloading Finished Part	2	R1X03	3	R1Y02	2	R1Z02

M1 and M2 are Machines Batch No Pr0D-Q01
R1 is Robot
I/B Buffer is Input/output Buffer

The following simplifying assumptions are made:
- tools and auxiliary equipments for processing each operation are available in the tool magazine,
- there is no machine or tool failure during processing of the parts,

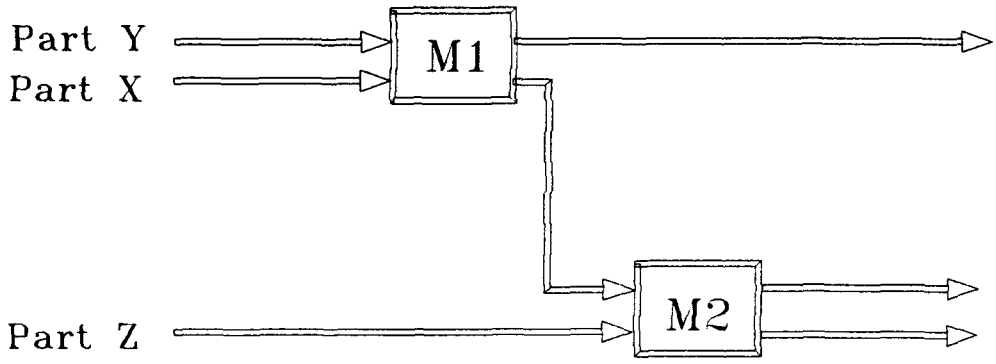

Figure 2. Operational Constraints of Parts

- machines and robot can handle one operation at a time
- negligible time is requried for loading of part programs to the programmable controllers of the cell.

6. METHODOLOGY

An FMS is a complex discrete event dynamic system with a large number of interconnected components. Better performance and high flexibility can be achieved in it by effective control and decision-making procedures and by appropriate redundancy management (Teng and Black [13]). Flexibility in this case implies a variability of decision alternatives with respect to the specified objective assigned to the cell controller. The modelling technique used to model at various stages to capture the static and dynamic behavior of the system at the cell level are described in the following sections. The method is illustrated in Figure 3.

6.1 Phase I : IDEF0(Static) Modelling

IDEF0 is top down design methodology where modularization and elaborations are two key concepts which makes the design and analysis of complex system manageable. The system design is organized in modules and each module is elaborated in more detailed design until it can be implemented in a reasonable states. One of the important features in static modelling is to capture the relational functions of the system's objects. For this task, a static model (the IDEF0 model) is developed in three hierarchical levels: level 0 (Factory level); level 1 (Cell level); level 2 (Operational level).

The system is decomposed in a hierarchical way to create children which in

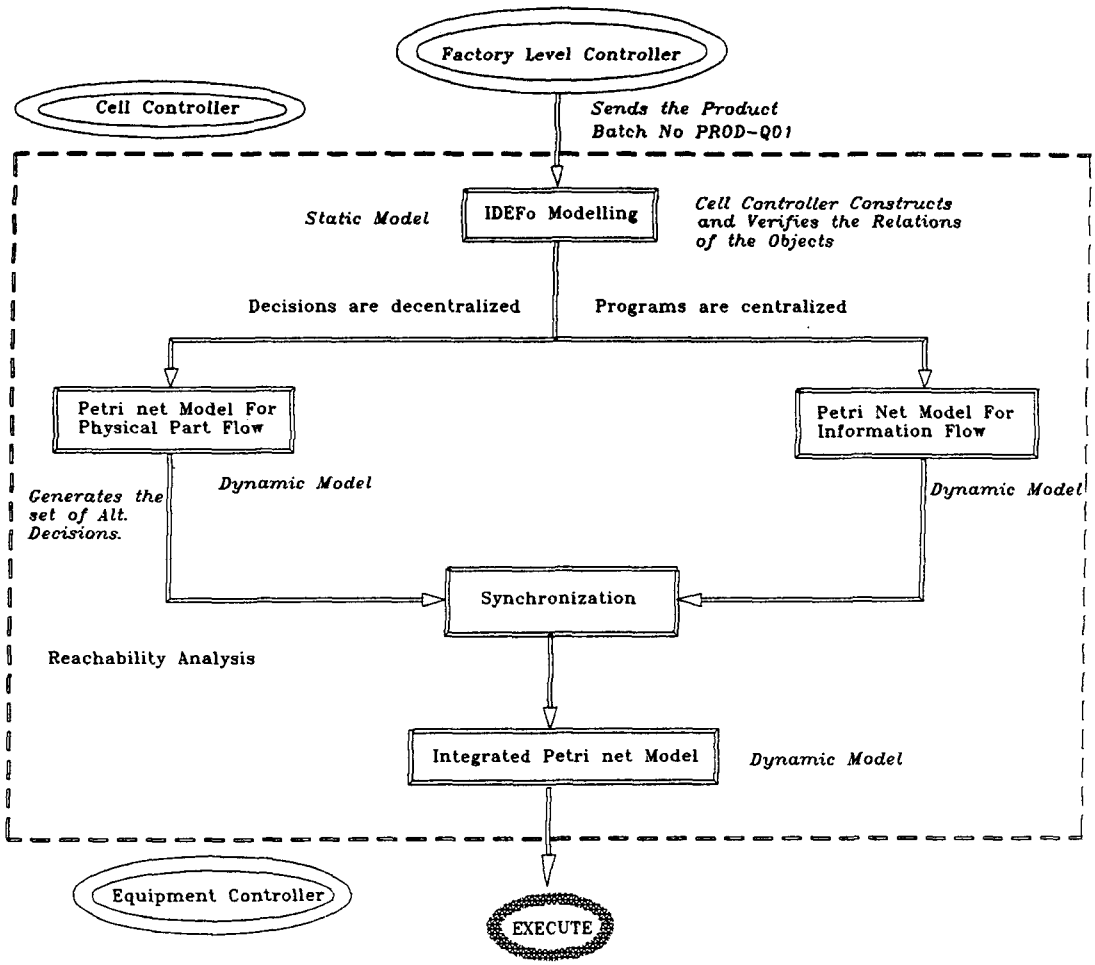

Figure 3 : Methodology and Model development Structure

turn may be parent for next level of child creation. This creation continues until the system reaches the lowest level. At the highest level, the parent node describes the function/activity "Produce Batch of Product No. PROD-Q01" . To implement this activity in the assigned cell, the cell controller has to check for consistency of all the objects. First of all it is necessary to define the input objects (here the input object is necessary raw materials for the batch of product) and then define output objects (here it finished parts of the batch). To transform the input objects to output objects, the mechanism needed are machine M1 & M2 and Robot R1. The next part is to determine the objects which control the parent activity of batch production. To execute the transformation, part programs for the operation of the machines and robot are needed. So the availability of the part programs will definitely control the

parent function in zero level. Other control command is availability of cell controller for communicating necessary informations.

In the next step, activity node of level 0 is decomposed to the functions: 'Produce product X' , 'Produce product Y' and 'Produce product Z' . Inputs, outputs, mechanism and control functions are shown in Figure 4. For a detailed analysis of the function at level 1, further decompositions are carried out up to the lowest level. This level contains the operations performed by machines and robots. Four distinct reports are generated from the model: the

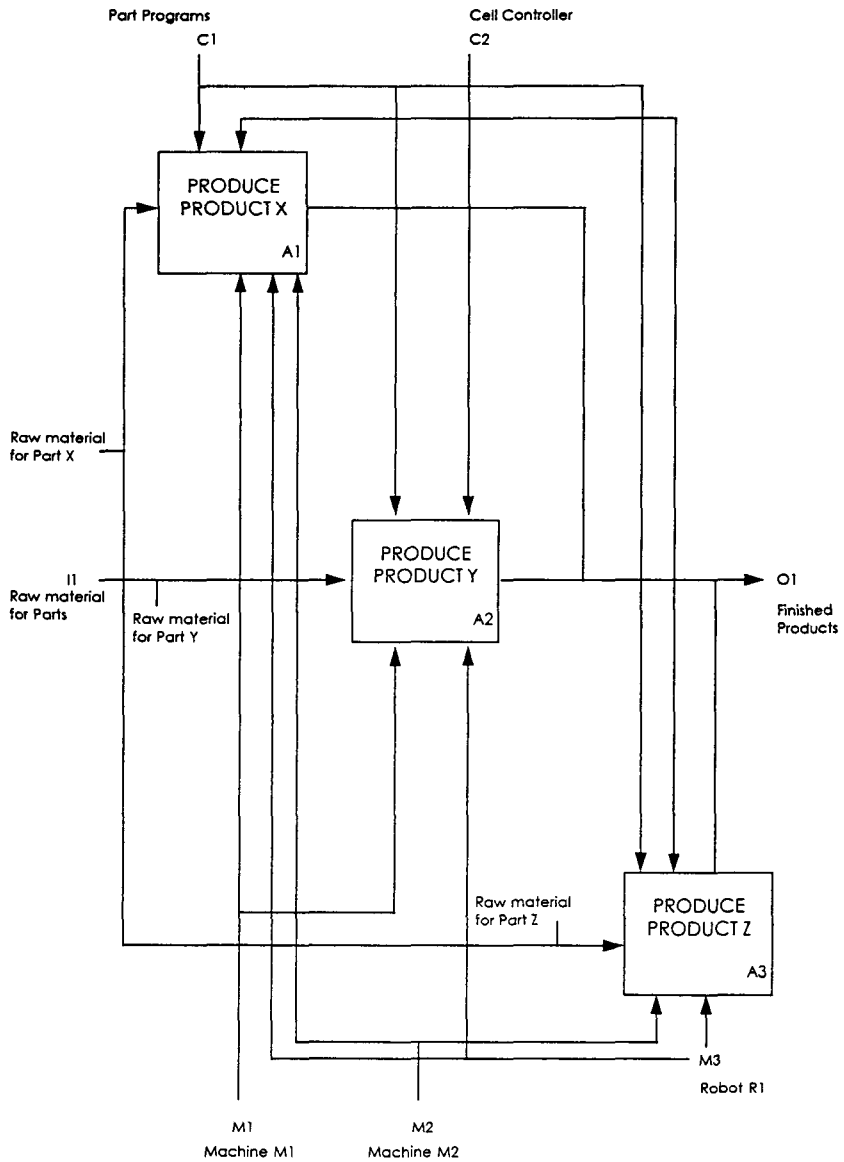

Figure 4. IDEF0 model at highest level

IDEF0 report, Activity report, Arrow report and Consistency report.

The reports indicate whether the model is consistent with all necessary linking of defined objects. The IDEF0 report gives a complete view on all activities with details of input, output, mechanism and control functions. The Activity report shows the functions together with all interrelated objects The Activity and Arrow reports are used for the conversion from the static (IDEF0) model to dynamic (Petri net) model. The Arrow report shows the individual objects and their flow path through the functions.

The model described above provides a sufficiently complete definition of the structure for creating dynamics and semantics of the given batch of production to carry out further analysis of the system.

6.2 Phase II: Petri Net Modelling

In this phase, Petri net models are developed based on the IDEF0 model for real time qualitative and quantitative analysis of the given problem. For developing the Petri net models the conventions followed are (e.g. Silva and Valette [12]):

(i) Conditions in the system such as availability of raw materials, machines, robot, end programs are represented by the Petri net's *places*.

(ii) Events occurring in the system (such as loading/unloading of parts, execution of programs, finishing of processing activity) are represented by *transitions*. The transitions incorporated here are *timed transitions*. A deterministic time value is associated with each transition.

(iii) States holding the conditions of the system are represented by the *tokens*. (A token is represented by a black dot in the specified place). Presence of a token in a specified place indicates the specified condition to be true. For example, a token is at place P1 in Figure 5 implies that the robot R1 is available for service.

(iv) Directed arcs determine the direction of material and information flow in the system. By putting weights on these arcs, pre and post conditions are specified for firing of a particular transition. No specification of a value indicates a weight equal to one.

On the basis of the Arrow and the Activity reports, the Petri net models are constructed by converting and analyzing the activities and arrows of IDEF0 model with the following convention:

Arrows represent pre and post conditions while functions (activities) are represented by transitions or (transition-place)-pairs. Those conditions which are the assumptions in the Petri net model are not shown in IDEF0 modelling, but

they play an important role (such as availability of tools or machine center (M/C) breakdown).

6.2.1 Physical Part Flow model

To analyze the dynamic behavior of the physical part flow, a Petri net model (PNM) is constructed on the basis of the Arrow and the Activity reports shown in Figure 5. The initial markings are represented by black dots. Interpretation of the places and transitions is described in Table 2 and Table 3, respectively.

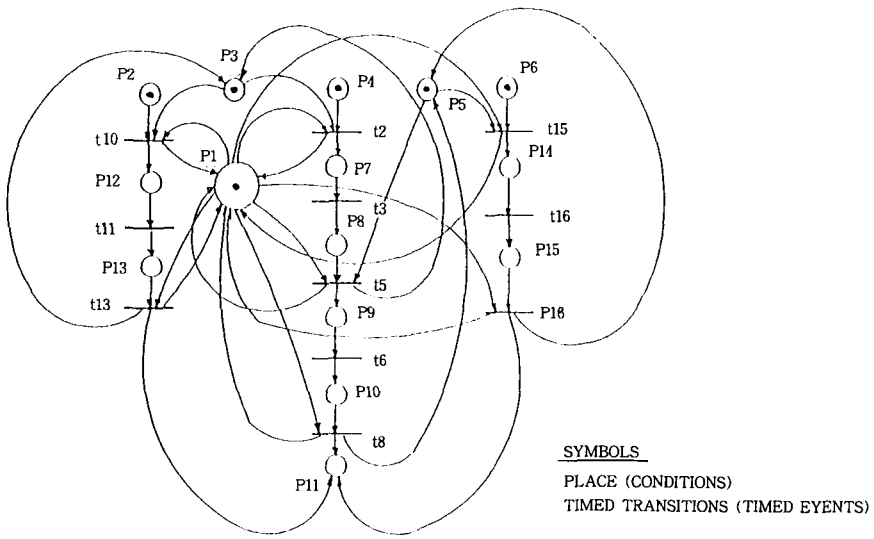

Figure 5. Petri Net Model of Physical Part Flow

Table 2. Interpretation of places in the PNM shown in Fig. 5

Place	Interpretation
P1	Robot R1 is available for service
P2	Raw material for part type Y ready
P3	Machine M1 ready
P4	Raw material for part type X ready
P5	Machine M2 ready
P6	Raw material for part type Z ready
P7	Machine M1 processing part type X
P8	Part type X ready to unload from machine M1
P9	Machine M2 processing part type X
P10	Part type X ready to unload from machine M2
P11	Finished arts ready in output buffer
P12	Machine M1 processing part type Y
P13	Part type Y ready to unload from machine M1
P14	Machine M2 processing part type Z
P15	Part type Z ready to unload from machine M2

Table 3. Interpretation of timed transitions in the PNM shown in Figure 5 (*Timed value of the transitions are shown in parentheses*)

Transition (timed)	Interpretation
t2	Robot transfers part type X from input buffer to machine M1(1)
t3	Machine M1 finishes processing part type X (3)
t5	Robot transfers part type X from machine M1 to machine M2(2)
t6	Machine M2 finishes processing part type X (4)
t8	Robot R1 transfers part type X from machine M2 to output buffer(2)
t10	Robot transfers part type Y from input buffer to machine M1(2)
t11	Machine M1 finishes processing part type Y (9)
t13	Robot transfers part type Y from machine M1 to output buffer(3)
t15	Robot transfers part type Z from input buffer to machine M2(1)
t16	Machine M2 finishes processing part type Z (t)
t18	Robot transfers part type Z from machine M2 to output buffer(2)

Since the Petri net model is an abstract and formal tool to describe the evolution of systems, the system can be studied through a coverability tree. Such a tree has a root, the initial marking M_0. The nodes of the tree respresent markings generated from the root and its successors. For a bounded Petri net, the coverability tree is called the *reachability tree* since it contains all possible reachable markings (Murata [10]). A reachability tree represents all the possible reachable states of the system starting from the initial marking. From the reachability tree, the other properties of the Petri net can be verified to prove the validation of the model.

To analyze the system's dynamic behavior the reachability tree is constructed for the physical part flow model as shown in Figure 6. The tree is split into three distinct parts shown in Figures 6(a), 6(b) and 6(c). The reachability tree in Figure 6(a) indicates those possible states where the cell controller selects part type Y to start first operation of the cell, i.e., loading of part type Y to machine by robot R1. Similarly, Figures 6(b) and 6(c) show the possible states in selecting first part type X or part type Z, respectively. In such a situation the cell controller needs sufficient intelligence to give priority to the job which fulfills the assigned objective.

Many situations may arise in selecting competing events on the way to finalize batch of production. In this case, it is necessary to make a choice among alternative actions and to solve conflicts among competing events. All possible firing sequences are enumerated from the reachability tree and a total of 112 are found. The Petri net can describe non-determinism and parallelism of activities but there is no formalism to represent its sequences. Moreover the sequences of activities can contain repetitions, non-determinism (choice) and

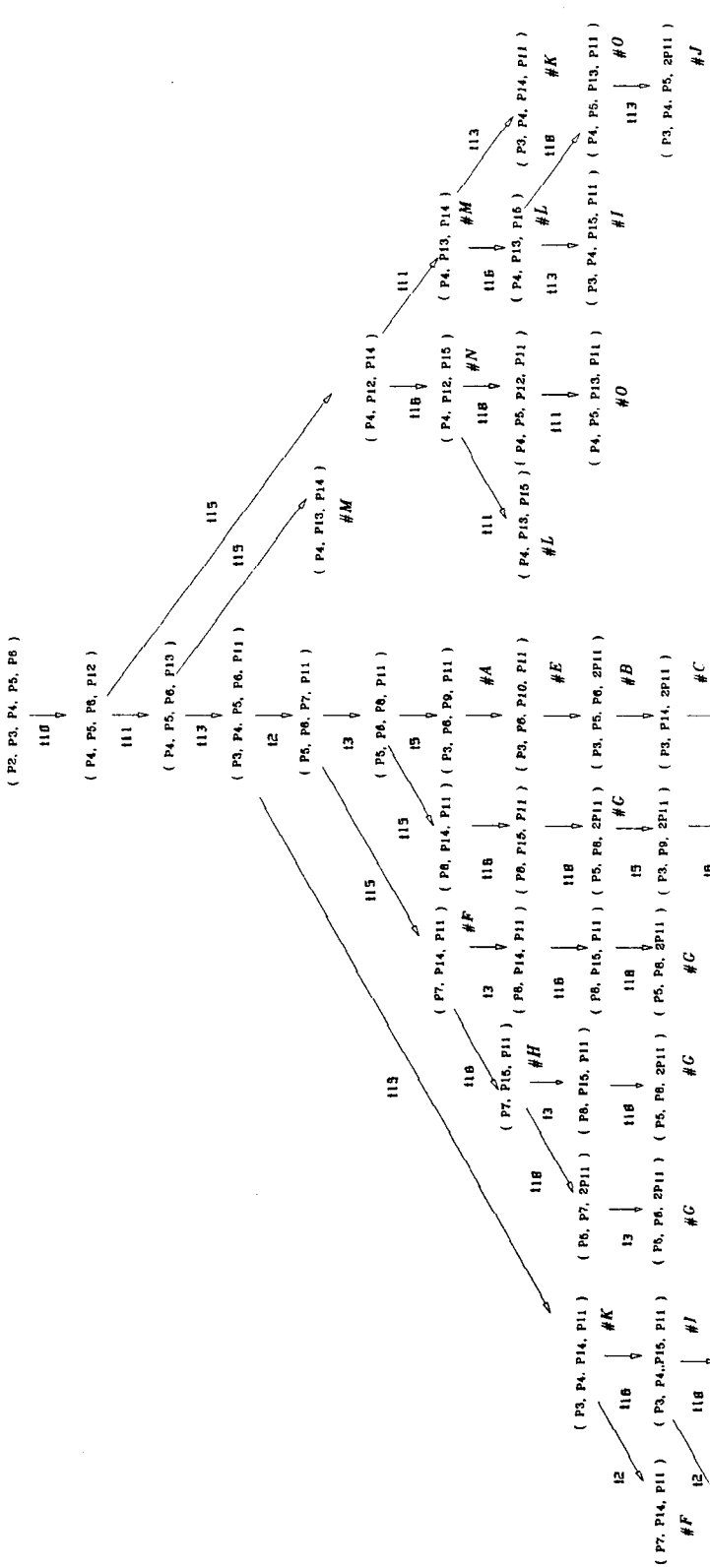

Figure 6(a). Reachability Tree for the Petri Net Model in Figure 5

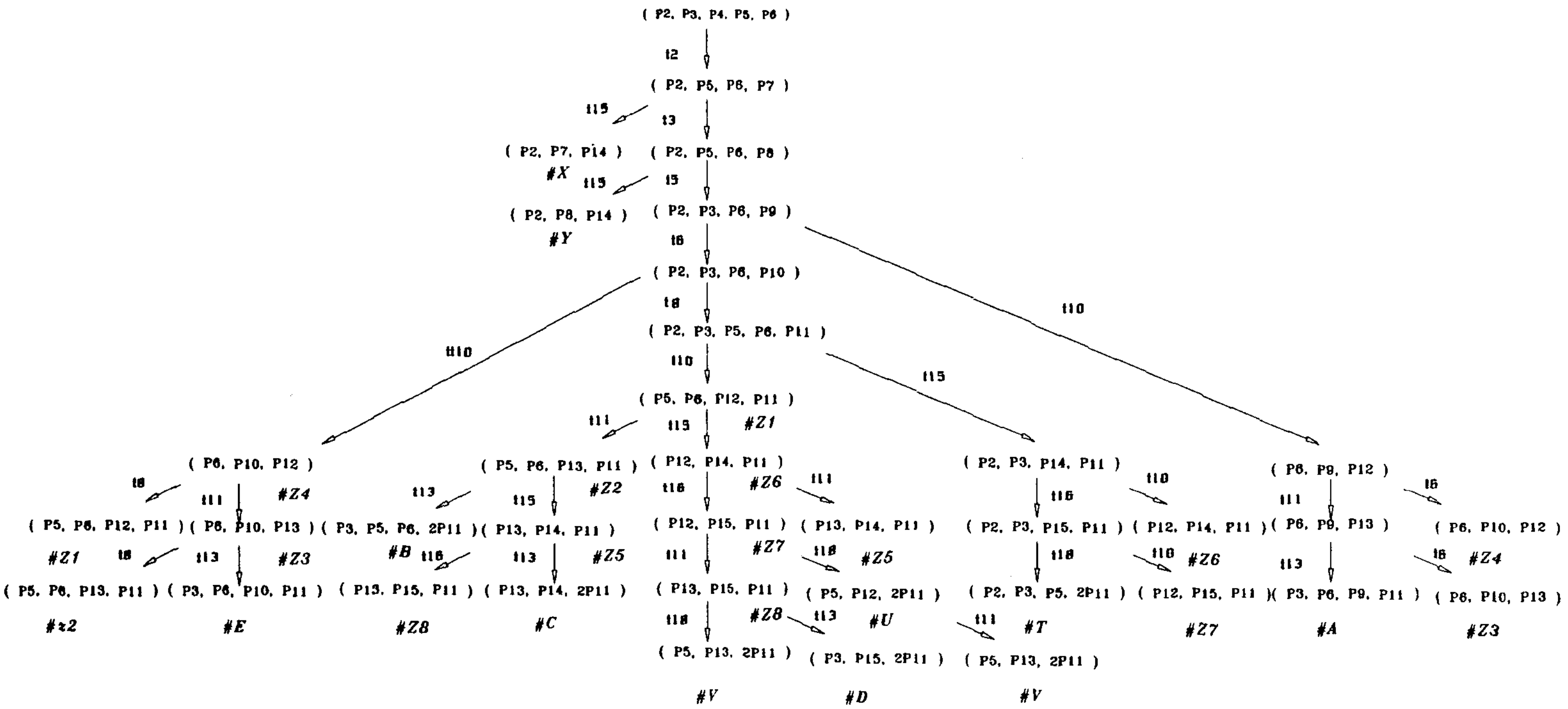

Figure 6(b). Reachability Tree for the Petri Net Model in Figure 5

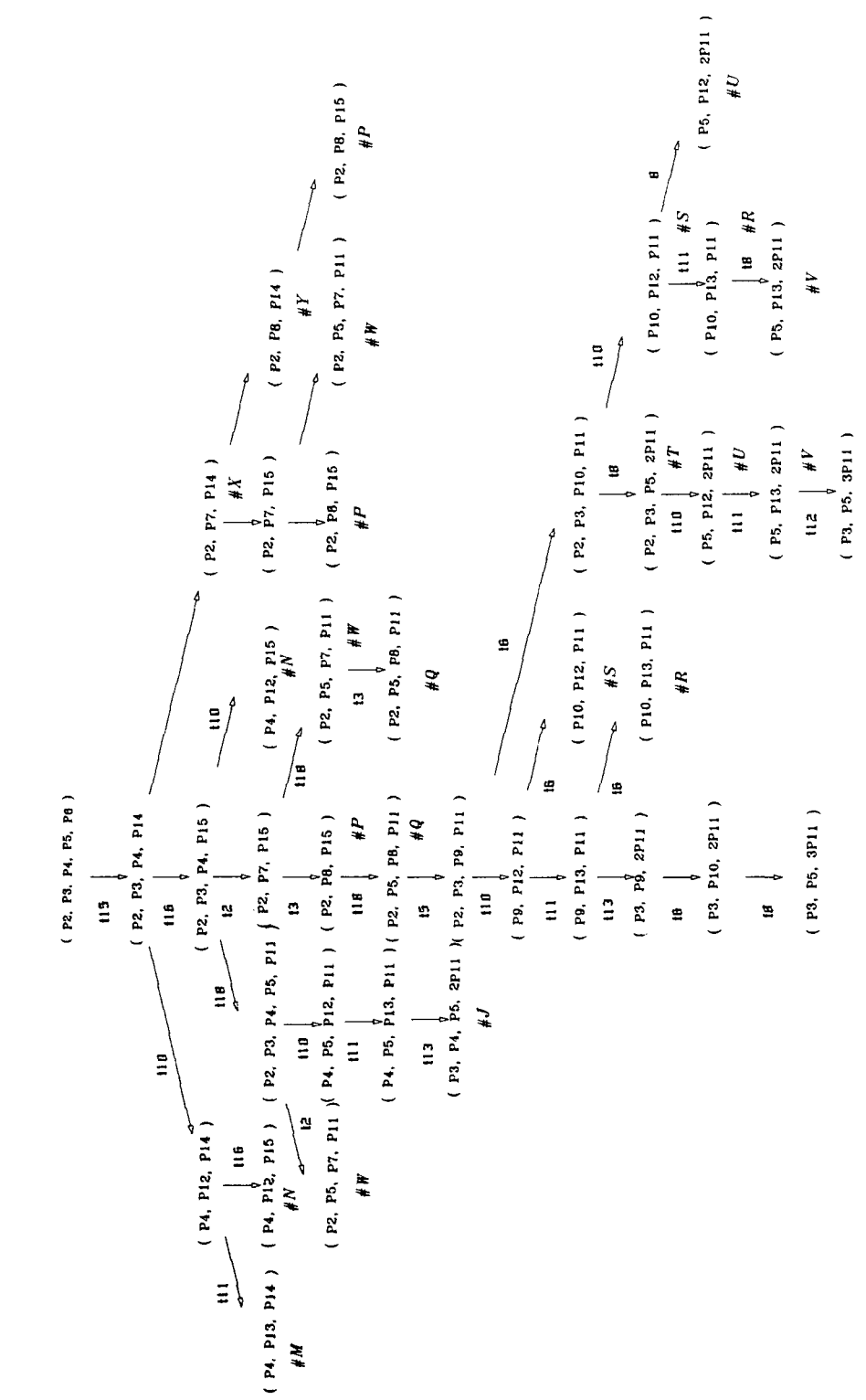

Figure 6(c). Reachability Tree for the Petri Net Model in Figure 5

parallelism, In such cases the problem of modelling sequences becomes more complex. Therefore, the sequences which possess such properties need an in-depth analysis to reveal the necessary information along with the set of alternative decisioins for the makespan in an ascending order of durations. To solve this complicated task, a simple algorithm is developed to calculate the makespan for each firing sequence.

To develop an algorithm for the calculation of the makespan the following constraints are imposed by the system:

Operational or Sequential constraints: These constraints shows the precedence relations of operations. They are–

(a) for part type X: Operations X1(t2)–X2(t3)–X3(t5)–X4(t6)–X5(t8):
(b) for part type Y: Operations Y1(t10)–Y2(t11)–Y3(t13):
(c) for part type Z: Operations Z1(t15)–Z2(t16)–Z3(t18).
Sequences of transition firings shown in parentheses.

Mutual Exclusion Constraints: These constraints represent the operations which share one of the resources in the system. Operations X1(t2), X3(t5), X5(t8), Y1(t10), Y3(t13), Z1(t15) and Z3(t18) share the resource 'machine M 2' . The algorithm proceeds as follows:
Select a sequence from the alternative decision chart, and compute start and finish time of the first operation.
Select the next activity. Check parallelism of the selected activity with the last operation.

If parallel,
Intitialize its start time with last operation.
Check if it is start of new part.
If new part,
compute start and finish time of the first activity of the next part type. Go to *Select next activity.*
If not new part,
Check the finish time of the last operation/mutual exclusion of precedent activity.
If precedent's finish time is greater than the initialized time value,
Reset initial value to this new value.
Compute and update the total time for the alternative. Go to *Select next activity.*
If the initialized value is smaller or unchanged

Compute and update the total time for the alternative. Go to *Select next activity.*

If not parallel,

Check precedent activity for operational and mutual exclusion resources (constraints).

If operational,

Update start and finish time by simple addition for the respective part. Go to *Select next activity.*

If resource,

Check if it is start of another part.

If start of a new part,

Compute start time of the new part taking the finish time of the last operation's start time. Go to *Select next activity.*

If not a new part,

Re-check the generation of sequences because there is a probability that the model is wrong or deadlocked.

Complete all activities of the sequence and enumerate the total makespan for the alternative.

The algorithm is restricted by the assumed configuration of the system. In this case at most two activities can be concurrent. However, it is believed that by using the same procedure the system can be generalized.

From the algorithm all possible time durations for the entire batch of product can be calculated. The algorithm generates a new type of diagram, a 'concurrent behavioral diagram' , showing start and finish time of each operation for each activity duration, the degree of concurrency and the invariant set of alternatives. As an example, a behavioral diagrm is shown for the minimum makespan objective (Figure 7). Now we can select to execute an alternative which minimizes the makespan and is synchronized with the available part programs in the centralized database management system controlled through the factory controller.

From the reachability tree other properties can be verified:

Boundedness: The model is 1-bounded except for the target buffer place (Output Buffer). The assumption that the number of processes executed by machines and robot is limited to at most one, is validated.

The number of tokens in the target buffer indicates the number of finished parts of the given batch of product.

Liveness: The Petri net is live. It guarantees deadlock-free operation.

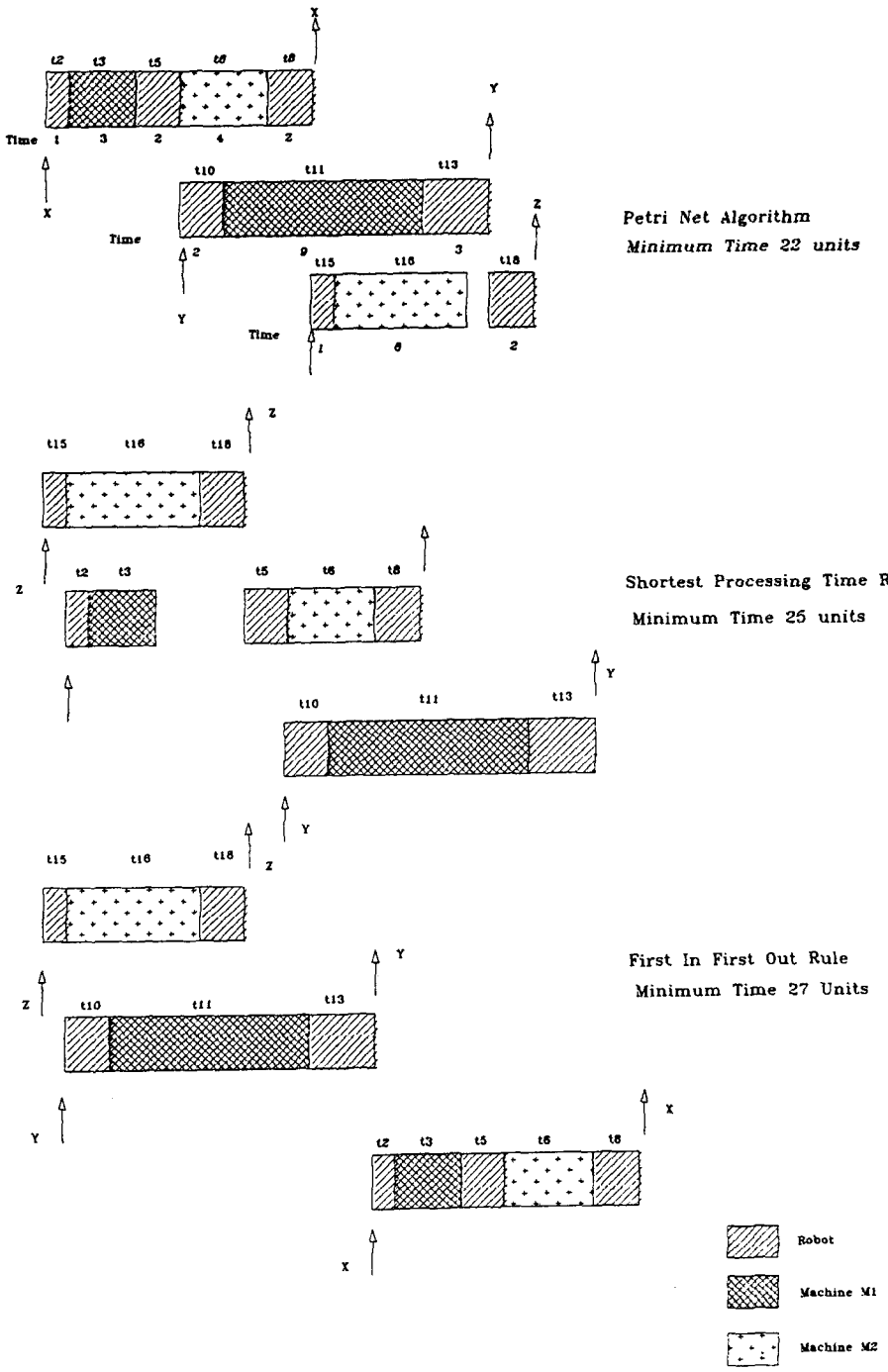

Figure 7. Comparison of Petri Net algorithm, SPT and FIFO rule for minimal makespan

Reachability: Reachability of various states from the initial or any intermediate states can be clearly seen from Figure 6. For instance in Figure 6(a) the state (P4, P13, P14) can be reached after firing the transitions t10, t11 and t15. Here the state (P4, P13, P14) represents processing of part type Y. Part type Z and availability of raw materials for part type X, respectively.

The following conventions are used to develop a physical part flow model in the software using inbuilt specification of the software.

Token attributes:

#m1 ⇒ availability of machine M1 for operation
#m2 ⇒ availability of machine M2 for operation
#robot ⇒ availability of robot R1 for operation

Timed transitions: Timed transitions are shown with a variable named "delay" , assigned with different timed values for different transitions; e.g., delay: =3 implies that three time units are assigned to the related transition.
Arrow attributes: Each arrow possesses attributes according to its necessity, shown in respective figures. For example in module Physical. Part Y, the arrow from place M/C M1 shows an attribute #m1 which means this arrow is allowed to carry the token having an attribute #m1.
Target Buffer: Place P11 repressents the buffer in Figure 5. Tokens in this place indicate finished parts.

With these attributes each time the software randomly selects the fireable transitions from the potential candidate events. Hence the model is simulated many times and it gives the simulated time and some of the sequences that we have computed by reachability graph. This proves the validity of the computed results as well as of the model.

6.2.2 Information flow model

In a similar manner as explained for the physical part flow model, we have constructed the information flow model from the IDEF0 model by creating more places for the arrow of 'cell controller' such as 'cell controller ready to request program' , 'programs request for respective operations' and 'program requested' . in the same way, the arrow 'part programs' is concerned with the availability of respective part programs in the database management system controlled by factory controller, and programs ready for execution.

However to clear the model we create a place for 'program back', where the programs are accumulated after execution of each operation. The tokens in

the information model hold the following attributes:

#cc ⇒ availability of cell controller for communication

#X1 ⇒ program request for operation X1, i.e. loading of part X from input buffer to machine M1

#Prog1 ⇒ availability of a part program in the central database management system controlled by the factory controller for operation X1

#X2 ⇒ program request for operation X2, i.e. processing of part type X in machine M1

#Prog2 ⇒ availability of a part program in the central database management system controlled by the factory controller for operation X2

#X3 ⇒ program request for operation X3, i.e. loading of part from machine M1 to machine M2

#Prog3 ⇒ availability of a part program in the central database management system controlled by the factory controller for operation X3

#X4 ⇒ program request for operation X4, i.e. processing of part type X in machine M2

#Prog4 ⇒ availability of a part program in the central database management system controlled by the factory controller for operation X4

#X5 ⇒ program request for operation X5, i.e. unloading of finished product from machine M2 to output buffer

#Prog5 ⇒ availability of a part program in the central database management system controlled by the factory controller for operation X5

#Y1 ⇒ program request for operation Y1, i.e. loading of part type Y from input buffer to machine M1

#Prog6 ⇒ availability of a part program in the central database management system controlled by the factory controller for operation Y1

#Y2 ⇒ program request for operation Y2, i.e. processing of part type Y in machine M2

#Prog7 ⇒ availability of a part program in the central database management system controlled by the factory controller for operation Y2

#Y3 ⇒ program request for operation Y3, i.e. unloading of part type Y from the machine M1

#Prog8 ⇒ availability of a part program in central database management

system controlled by the factory controller for operation Y3

#Z1 ⇒ program request for operation Z1, i.e. loading of part type Z from input buffer to machine M2

#Prog9 ⇒ availability of a part program in the central database management system controlled by the factory controller for operation Z1

#Z2 ⇒ program request for operation Z2, i.e. processing part type Z in machine M2

#Prog10 ⇒ availability of a part program in the central database management system controlled by the factory controller for operation Z2

#Z3 ⇒ program request for operation Z3, i.e. unloading of finished part Z from machine M2

As each machine or robot can execute only one program at a time, the capacity of the place 'program execution' is limited to one.

With these features, the model is run in the software and shows deadlock in some situation which can be visualized in animation of the model. Deadlocks occur because the physical part flow model is not synchronized with the information flow model for the desired alternative to execute. To solve this problem, it is necessary to find some synchronization procedure by adopting some specification.

6.3 Phase III: Integrated Petri net Model for Physical Part Flow and Information Flow

A Petri net structure can be viewed as partial ordering of transitions (events). Knowledge of the complete dynamic behavior of the physical part flow and information flow should be revealed in this model. A decision structure for each set of alternatives can be viewed from their respective concurrent behavioral diagrams. For instance take the set of sequences for minimum makespan schedule which is enumerated as 22 time units using our Petri net algorithm. From the behavioral diagram shown in Figure 7 for the respective transition sequence both show invariance with respect to discrete time events for availability of part programs in the central database management system. From the figure it is clear that to execute this decision in the assigned cell, necessary part programs must be available in central database management system in the following discrete time epochs shown in Table 4.

To synchronize the physical part flow model with the information flow model two dummy nodes P1 and P2 are created. From the behavioral diagram it is clear that the operating X4 and Y1 are concurrent and both strated at

Table 4. Part programs with discrete time epochs for the minimum makespan sequence.

Time	Part Program	Program code	Transitions
0	#program1	R1X01	t2
1	#program2	M1X01	t3
4	#program3	R1X02	t5
5	#program4	M1X01	t6
5	#program6	R1Y01	t10
8	#program7	M1Y01	t11
10	#program5	R1X03	t8
12	#program9	R1Z01	t15
13	#program10	M2Z01	t16
17	#program8	R1Y02	t13
20	#program11	R1Z02	t18

the same time. By creating the dummy node P1, the available token in the place as an output result of operation X3 leads to exection of concurrent activities X4 and Y1. Similarly another dummy node activates operations X5 and Z1. In this approach an illustrated instance can be animated to visualize the real time implementation of the decision. This is only one instance demonstrated. With this approach assigning different objectives to the cell controller such as due dates for specific jobs, the same generated list will provide all possible due dates from which the cell controller can select the necessary one for execution. However, the selected sequence should also by synchronized with the available part programs in the central database management system in the proposed system.

For the same product batch, results of the Petri net based algorithm are compared with Shortest Processing Time(SPT) and First In First Out (FIFO) scheduling rule(Figure 7). In the SPT rule, the minimum makespan is 25 time units, while in FIFO scheduling, the minimum time is 27 units. On the other hand, the Petri net based algorithm gives 22 units of time for the same batch of production. This proves the efficiency of the Petri net based cell controller. In this work, we have shown a dynamic approach for generating real-time decisions and control structures to solve the FMS scheduling problem.

7. CONCLUDING REMARKS

In our approach, a static model is first developed to understand the underlying

detailed functional relationships between various manufacturing activities with the IDEF-method. This model provides sufficient definition for developing semantics of the dynamic model.

To construct the dynamic model, it is found that the batch production system can be decomposed into two distinct domains: physical part flow and information flow. In the physical part flow Petri net model, the reachability tree technique is used to derive the decision structure of the problem. Sequencing the part process plan should be such that the detailed knowlege about the real state of the system is available. In this context, a simple algorithm is developed to obtain the minimal time to complete a batch of products assigned to the cell.

Alternative decision sets are generated for various objectives, In a later stage, for the chosen objective, the selected alternative is synchronized to develop an integrated model for the product batch. Complete information concerning the entry of parts to the machines, their exits from the machines, makespan of the batch, processing durations, and idle periods can be seen in a new type of diagram, which we called the 'concurrent behavioral diagram' .

In the present study, a comparative study of other scheduling techniques such as Shortest Processing Time (SPT) and First In First Out (FIFO) is made. It is found that the present methodology provides better results and more information with a higher flexibility of alternative decisions. This approach however is limited to a few machines and a few parts.

The proposed methodology is superior to simulation because the latter does not have the opportunity to study deadlock, priority, and non-determinism in real-time. The combination of using IDEF and Petri nets has proven to be an efficient control mechanism at the cell level.

REFERENCES

[1] Alberti N., U. Commare U. and S. Diega, "Cost efficiency: an index of operational performance of an automated production environment," in K.E. Stecke and R. Suri, *Proc. of the Third ORSATIMS Conference on Flexible Manufacturing Systems*, MIT, USA, (1989), pp. 67-72

[2] Archetti F. and Sciomachen S., "Development, Analysis and Simulation of Petri Net Models. An Application to AGV Systems," in F. Archetti, M. Lucertini and P. Serafini(eds.), *Operations research Models in Flexible Manufacturing Systems,* CISM Courses and Lectures no. 306, (1989), pp. 91-112, Springer Verlag, Wien.

[3] Boucher, T.O. and M.A. Jafari, "Design of a factory floor sequence controller from a high level system specification," *Journal of Manufacturing*

Systems, vol. 11 no. 6, (1992), pp. 401-417.

[4] Bravco R.K. & S.B. Yadav, "A methodology to model the functional structure of an organization," *Computers in Industry*, vol. 6, (1985), pp. 345-361.

[5] Colquhoun G.J., R. W Baines and R. Crossley, "A state of the art review of IDEF0," *Int. Journ. Computer Integrated Manufacturing*, vol 6, no. 4, (1993), pp. 252-264.

[6] Dicesare, F., G. Harhalakis, J.M. Proth, M. Silva and F.B. Vernadat, *Practice of Petri Nets in Manufacturing,* Chapman & Hall, London, 1993

[7] Favrel J., G.R Oh, P. Baptiste P. and K.H. Lee, "Hierarchical control structure of FMS by Petri Net Method," in Yoshikawa and Burbridge(eds.), IFIP Publication, (1987), pp. 223-243.

[8] Janssens, G.K., "Petri nets for the specification, simulation and analysis of manufacturing system," in W. Kuehn and N.N. Nagarur (eds.), *WORKSIMS 94-Simulation in Manufacturing Systems,* Asian Institute of Technology, Bangkok, Thailand, (1994), pp. 46-53.

[9] Lin, Y.-J. and J.J. Solberg, "Flexible routing control and scheduling," in K.E. Stecke and R. Suri(eds.), *Proc. of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems*, MIT, USA. (1989), pp. 45-56.

[10] Murata, T., "Petri nets: properties, analysis and applications," *Proceedings of the IEEE,* vol. 77, no. 4, (1989), pp. 541-577.

[11] O'Grady P.J., H. Bao and K.H. Lee, "Issues in intelligent cell control for Flexible Manufacturing Systems," *Computers in Industry*, vol. 9, (1987), pp. 26-36.

[12] Silva, M. and R. Valette, "Petri nets and Flexible Manufacturing," in G. Rozenberg(ed.), *Advances in Petri Nets '89*, Lecture Notes in Computer Science no. 424, (1989), pp. 374-417.

[13] Teng, S.H. and J.T. Black, "Cellular Manufacturing System modelin: the Petri net approach," *Journal of Manufacturing Systems*, vol. 9, no. 1, (1990), pp. 315-320.

[14] Tu Y. and A. Sorgen, "Real time scheduling and control of transportation in Flexible Manufacturing Cells," *Computers in Industry*, vol. 16, (1991), pp. 315-320.

[15] Viswanadham, N. and Y. Narahari, "*Performance Modelling of Automated Manufacturing Systems,*" Prentice Hall. 1992

[16] Zhou, M.C. and F.Dicesare, "*Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*," Kluwer Academic Publ., Boston 1993