ABCM Symposium Series in Mechatronics - Vol. 3 - pp.405-413
Copyright © 2008 by ABCM

FLEXIBLE MANUFACTURING SYSTEMS MODELLING USING HIGH LEVEL PETRI NETS

Alexandre da Silva Ribeiro, aribeiro@cimatec.fieb.org.br
SENAI – CIMATEC, Salvador, Bahia, Brasil

Eduard Montgomery Meira Costa, edmonty@ig.com.br
Área 1 Faculdade de Ciência e Tecnologia, Departamento de Engenharia Elétrica, Bahia, Brasil

Eduardo José Lima II, eduardo@demec.ufmg.br
Universidade Federal de Minas Gerais – Departamento de Engenharia Mecânica, Minas Gerais, Brasil

Abstract. *Flexible Manufacturing Systems (FMS) produce products with different specifications through the automactic program change of machines as well as changing equipments and tasks sequences used to process the parts. So, the FMS model must include all the possibilities of produced parts and tasks to be carried through. The use of High Level Petri Nets allows to incorporate data structures to the Petri Net model, without, however, increase the model complexity. This work shows a comparative study of formal discrete event systems modelling methodologies based on High Level Petri Nets (Coloured PN, Predicate-Transition PN and Objetc PN) and its applicabiliy to flexible manufacturing systems modelling to elect the most suitable methodology to this type of system. It is shown a real flexible manufacturing system modelling in order to verify its practical applicability.*

Keywords: *Flexible Manufacturing Systems, Discrete Event Systems, High Level Petri Nets.*

1. INTRODUCTION

Discrete Event Systems (DESs) are systems whose state-space is discrete and the evolution is driven by the occurrence of discrete events, with instantaneous duration in time. Computational systems and manufacturing systems are examples of DESs.

Many DESs such as manufacturing cells are composed by sub-systems which have to obey a set of coordination constraints in order to work together (Lima II and Dorea, 2003). The Supervisory Control Theory, proposed in (Ramadge and Wonham, 1989), based on formal languages and finite state automata establishes necessary and sufficient conditions for the existence of a minimally restrictive supervisor and supplies a formal methodology for its computation. The practical use of this approach is, however, limited by the state-space explosion with the growth of the number of subsystems. Even though this problem can be alleviated with the use of modular control, (Ramadge and Wonham, 1989; Queiroz and Cury, 2000), the controller complexity could nevertheless make it difficult do be implemented in practice.

As an alternative to the complexity of Ramadge and Wonham approach, some works have explored the simplicity and the graphic power of Petri Nets (Yamalidou *et al.*, 1995; Moody and Antsaklis, 1998; Lima and Dorea, 2002). However, in more complex manufacturing systems, the resources used for the manufactured product transformation, as well as the product characteristics changes through the process may lead the Petri Nets models to a great complexity and difficult analysis. In this case, the use of High Level Petri Nets becomes attractive. This type of modeling keeps the graphical and analysis power of Petri Nets, added to the possibility of a detailed definition of the processes types and manufactured product characteristics.

This paper shows an analysis of tree types of High Level Petri Nets (Coloured Petri Nets, Predicate-Transition Petri Nets and Object Petri Nets) and its applicability to flexible manufacturing systems. In Section 2, it is discussed some aspects of flexible manufacturing systems. Section 3 describes a real system to be modeled. Section 4 presents the High Level Petri Nets, showing the cited system modeling using each type of Petri Net.

2. FLEXIBLE MANUFACTURING SYSTEMS

Flexible manufacturing systems (FMS) are an attempt to reconcile the efficiency of the production line with the flexibility of the job shop in order to satisfy a versatile demand at low cost (Silva and Valette, 1989). In FMS, one may introduce new product families in the system during its operation and with little effort, while the system handles concurrently a large variety of product families at a given time. In order to meet these requirements, a Flexible Manufacturing System if formed of:

- a set of flexible machines,
- an automatic transport system,
- a sophisticated decision making system to decide at each instant what has to be done and on which machine.

Flexible machines have the capability of performing various operations. They have and automatic tool storage and retrieval system and machining programs can be downloaded at any time. This flexibility can be called physical flexibility (Erschler and Tersac, 1988).

An automatic transport system is required in order to transport the parts to the machine where the next operation is to be executed. This system has to be sophisticated because, in absence of a physical production line, the layout does not necessarily correspond to the sequences of machine utilizations. Any location on the shop floor has to be reachable from any other one (Silva and Valette, 1989). This automatic transport system may be composed by conveyors, auto guided vehicles and industrial robots.

Although PLCs allow convenient connections to workcell devices, they do not provide an environment for the synthesis and implementation of sophisticated control strategies. Controversially, PCs can provide an effective platform to generate and implement such control strategies but are more cumbersome to connect to devices for input/output purposes. Therefore, herein a host computer is used as a planner and a PLC is used as an execution device (Lauzon *et at.*, 1996).

Considering that in a large flexible manufacturing system there are many independent subsystems, an essential element of integration is the computer. Through industrial networks, a central computer running a software SCADA (Supervisory Control And Data Acquisition) is capable of communicate with the processor of each element of the productive process (CLPs, CNCs, robots etc.), in order to coordinate all the operations to produce parts in accordance with previously established specifications.

Communicating with each equipment and local controller of all the FMS element, the SCADA is capable of acquire data, modify the processes set-points, command program exchange and start and interrupt tasks. Figure 1 shows an example of a manufacturing system controlled by a SCADA system.

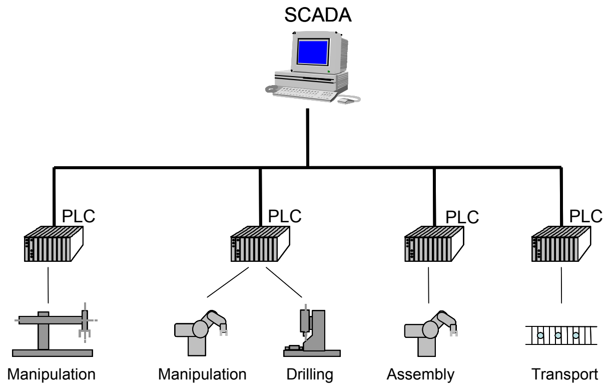


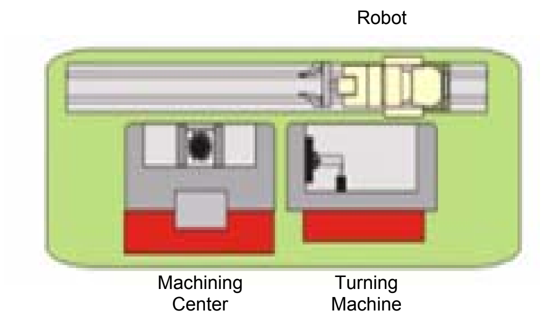
Figure 1 – Flexible manufacturing system controlled by a SCADA system.

3. DESCRIPTION OF THE FMS SYSTEM

The FMS system to be modeled is a machining cell (Fig. 2) composed by:

- an anthropomorphic robot model RV3AL manufactured by Mistubishi;
- two buffers to hold the pallets with the parts;
- a CNC turning machine model GALAX 10 manufactured by ROMI;
- a vertical machining center model Discovery 4022 manufactured by ROMI.

The robot moves on a prismatic axis to reach the machines. As the cell has two buffers, one for each machine, it is capable to process two parts in parallel.



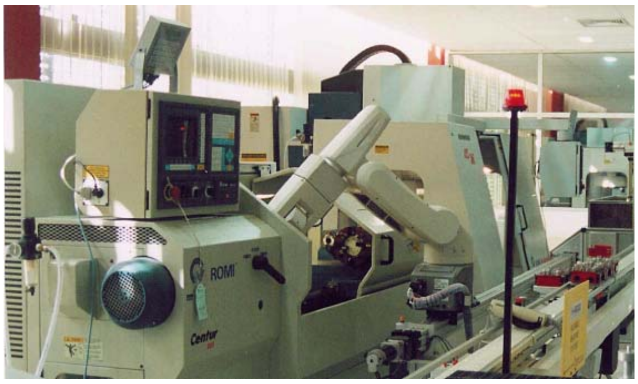


Figure 2 –Machining cell.

The machining cell receives four types of raw material to be processed as follows:

- Cylinder 1 is machined in the turning machine to produce Part 1 (Fig. 3(a));
- Cylinder 2 is machined in the turning machine to produce Part 2 (Fig. 3(b));
- Block 1 is machined in the machining center to produce Part 3 (Fig. 3(c));
- Block 2 is machined in the machining center to produce Part 4 (Fig. 3(d));

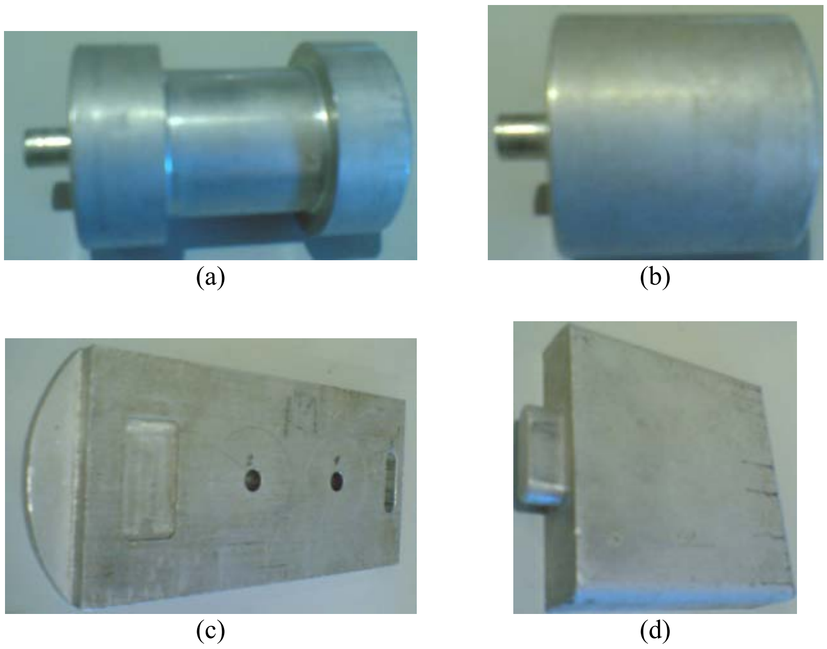


Figure 3 –Produced parts: Part 1 (a) and Part 2 (b) produced in the turning machine; Part 3 (c) and Part 4 (d) produced in the machining center.

The cell receives a pallet containing the raw material (cylinders or blocks) to be processed, as shown in Figure 3. According to the type of material, the robot takes the pallet and places it in the corresponding buffer. Pallets containing cylinders are placed in buffer 1 and pallets containing blocks are placed in buffer 2.

After releasing the pallet the robot takes the part and places it in the corresponding machine. After the machining, the robot takes the processed part, puts it again in the pallet and conducts the pallet out of the cell.

4. PETRI NETS

4.1. Ordinary Petri Nets

Petri Nets was introduced by Carl Adam Petri (Petri, 1962). A Petri Net is a directed bipartite graph whose structure is described by places and transitions connected by oriented arcs representing the flow relation from places to transitions and from transitions to places. In a marked Petri Net, each place has a zero or positive integer number of tokens, representing a partial state of the system.

When a transition fires, a number of tokens are removed from some places and added to others. An arc with weight D^+_{ij} from transition *j* to place *i* indicates that when transition *j* fires, place *i* will receive D^+_{ij} tokens. An arc with weight D^-_{ij} from place *i* to transition *j* indicates that when transition *j* fires, place *i* will lose D^-_{ij} tokens. Thus for a transition to fire, all of its input places must contain a minimum number of tokes. A transition that meets these conditions is enabled and is free to fire. A disabled transition may not fire. When transition fires, all of its input places lose a number of tokens, and all of its output places gain a number of tokens (Moody and Antsaklis, 1998).

When modeling manufacturing systems, transitions are related to events such machining start, machining end, movement start, movement end etc. Places are then related to partial states of the system as part ready to be processed, part been processed and part ready to be transferred.

Figure 4 show a simple net modeling the cited machining process. A token in place p_1 represents the part waiting to be processed. A token in place p_2 represents a part been transported to the machine. A token in place p_3 represent the part been processed and a token in place p_4 represent a part been removed from the machine. Finally, the tokens in place p_5 represent the processed parts.

Firing of transition t_1 means that a part arrived in the cell and firing of t_2 means the start of part transport. Firing of t_3 means the end of transport and start of operation. Firing of t_4 means end of operation and start of transport. Firing of t_5 means that the part was placed outside the cell.

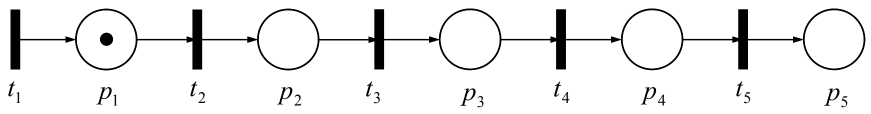


Figure 4 – Example of the machining model.

Although transition t_1 is always enabled, it will only fire when a part arrives in the machine. If this subsystem model is a part of a greater model, transition t_1 will have as input place one place representing the part been fed to the cell.

If a supervisory control logic is obtained by place invariants (Moody and Antsaklis, 1998), the model can be subjected to constraints to limit the number of parts been processed simultaneously (just one, in the simpler case of a single machine) and the number of parts been transported by the robot simultaneously (just one). Also a dead lock have to be prevented in the case when a part is been processed and the robot pick a new part to place in the machine. In this case, the robot could not release the part in the machine because it is processing another part and could not pick the processed part from the machine as it is with another part in its grasp.

The controlled model has two additional places (p_6 and p_7), representing respectively the robot resource and the machine resource are idle. Figure 5 shows the controlled system model.

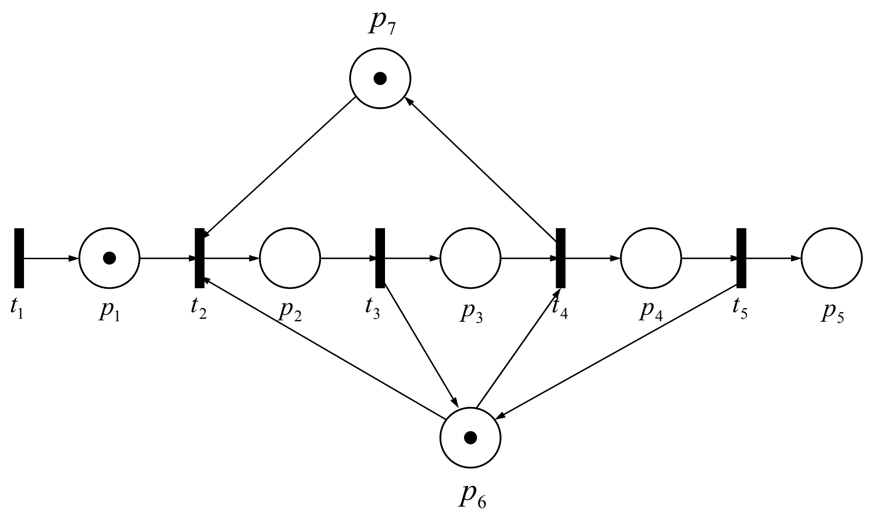


Figure 5 – Controlled system model.

4.2. High Level Petri Nets

The use of Petri Nets in real practical problems, mainly computational problems which work with great databases, had shown two main disadvantages (Villani, 2004). First, the concept of data does not exist and the models become excessively great because all the data manipulation has to be represented through the net structure (that is, through places and transitions). Second, hierarchy notion does not exist and, therefore, it is not possible to construct a great model through a separate set of sub-models with well defined interfaces.

High Level Petri Nets were proposed to solve these problems, incorporating data structures and hierarchic composition to the original Petri Net model (Villani, 2004). Among the High Level Petri Nets are Coloured Petri Nets (Murata, 1989; Lakos, 1995; Jensen, 1997), Predicate-Transition Petri Nets (Genrich, 1987) and Object Petri Nets (Lakos, 1995; Cardoso and Valette, 1997).

4.2.1. Coloured Petri Nets

To differentiate the tokens, different colors (integer numbers or set of labels) are associated to these. As a consequence, to each place it is associated a set of colors of the tokens which may belong to this place. To each transition it is associated a set of colors that corresponds to the different ways to fire the transition. In the simplest cases, when all the processes have the same structure and are independent to each others, the colors of transitions are directly associated to the processes, and the set of colors of the places and transitions is identical (Cardoso and Valette, 1997).

Using Coloured Petri Nets, the model shown in Figure 5 may include the two machines and the four processed parts in the studied manufacturing cell (Fig. 6)

Place p_7 may have two types (or colors) of tokens: mch_1 and mch_2, representing respectively the turning machine and the machining center. Place p_6 may have only one type of token (rb) representing the cell has only one robot.

Places p_1 and p_2 may have four types of tokens:

- $raw_{1,1}$ represents a raw part type 1 to be processed in machine 1
- $raw_{2,1}$ represents a raw part type 2 to be processed in machine 1
- $raw_{3,2}$ represents a raw part type 3 to be processed in machine 2
- $raw_{4,2}$ represents a raw part type 4 to be processed in machine 2

Place p_3 may also have four types of tokens:

- $prg_{1,1}$ represents program 1 is running in machine 1 to produce part 1
- $prg_{2,1}$ represents program 2 is running in machine 1 to produce part 2
- $prg_{3,2}$ represents program 3 is running in machine 2 to produce part 3
- $prg_{4,2}$ represents program 4 is running in machine 2 to produce part 4

Places p_4 and p_5 may have tokes of types $part_1$, $part_2$, $part_3$ and $part_4$ representing the four types of produced parts.

Firing of transition t_1 creates in place p_1 a token of type $raw_{x,y}$. x represents the type of raw material and y represents which machine is to be used to process the part. Values of x and y may be determined by sensors or by a message sent by the plant supervisor.

Transition t_2 is enabled if p_1 has a token, p_7 has a token of same y index of the token in p_1 and if the robot is idle (token rb in place p_6). When t_2 fires, it removes the token mch_y from p_7, rb from p_6 and $raw_{x,y}$ from p_1 and adds a token $raw_{x,y}$ to p_2. When t_3 fires, it changes the token $raw_{x,y}$ to $prg_{x,y}$ meaning the program is running and releases the resource rb. When t_4 fires, it changes the token $prg_{x,y}$ to $part_x$ releases the resource mch_y and takes the resource rb. Finally, t_5 moves the processed $part_x$ from place p_4 to p_5 and releases the resource rb.

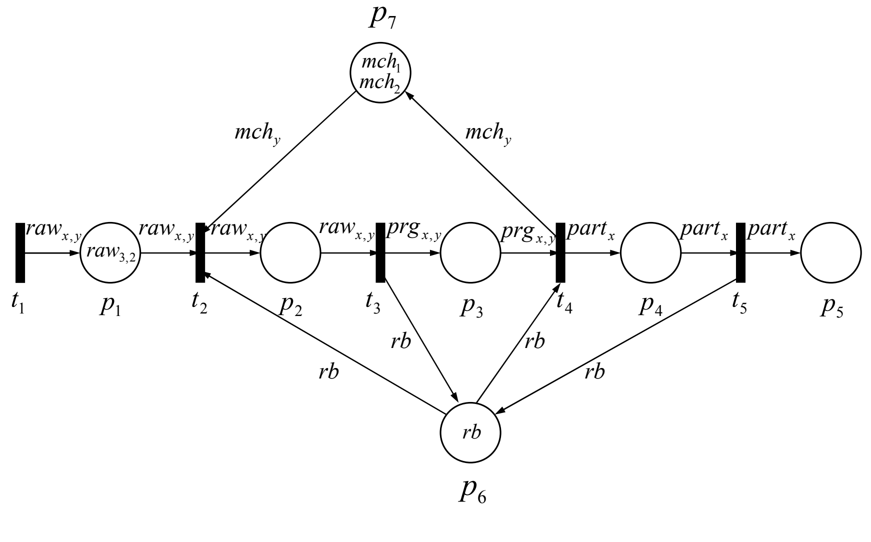


Figure 6 – Coloured Petri Net model.

A Coloured Petri Net can be converted to its equivalent Ordinary Petri Net by converting each place to a number of places equivalent to the number of possible token colors of this place. Each transition is converted to a number of transitions corresponding to the different ways of firing.

Figure 7 shows the equivalent Ordinary Petri Net to the Coloured Petri Net model from Figure 6. Transition t_1 is converted to t_{11}, t_{21}, t_{31} and t_{41}, meaning the four ways of firing t_1. Place p_1 is converted to p_{11}, p_{21}, p_{31} and p_{41}, as p_1 may have four Colors of tokens. Place p_7 is converted into 2 places, p_{17}, p_{27}, as the system has two machines. Place p_6 continues as p_6, as the system has only one robot.

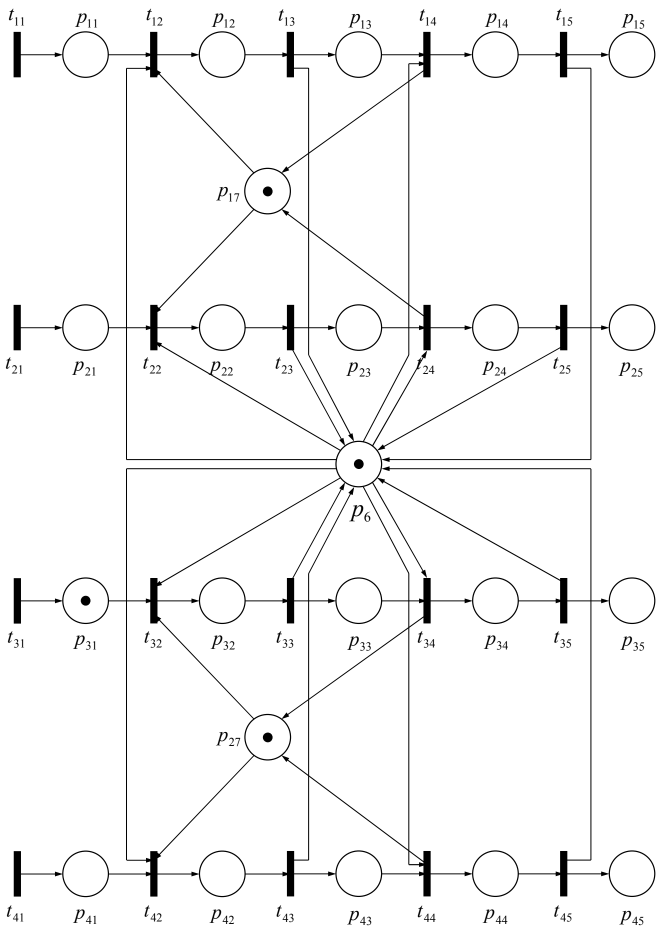


Figure 7 – Ordinary Petri Net model.

4.2.2. Predicate-Transition Petri Nets

In a Predicate-Transition Petri Net, the Ordinary Petri Net transitions are considered as rules in a propositional logic system (without variables). The description power is increased by substituting these rules by first order logic rules (rules with variables). Thus, the representation is not only more concise, but the model allows to better study the structural and behavioral properties of the system. A rule (transition) describes, then, a family of events and not only one event. The family is defined by the set of possible substitutions of variables by values (Cardoso and Valette, 1997).

In the Ordinary Petri Net in Figure 7, transitions t_{12}, t_{22}, t_{32} and t_{42} may be described by the rules:

if raw material type 1 is available *and* robot is idle *and* machine 1 is idle *then* transport raw material to machine 1
if raw material type 2 is available *and* robot is idle *and* machine 1 is idle *then* transport raw material to machine 1
if raw material type 3 is available *and* robot is idle *and* machine 2 is idle *then* transport raw material to machine 2
if raw material type 4 is available *and* robot is idle *and* machine 2 is idle *then* transport raw material to machine 2

Using first order rules and defining variables <*x*> and <*y*>, these rules may be substituted by:

if raw material type <*x,y*> is available *and* robot is idle *and* machine <*y*> is idle
then transport raw material <*x,y*> to machine <*y*>

Variable x may be substituted by values from 1 to 4 and y by values 1 or 2 during evaluation of the rule. It may be easily stated that the variable values can be associated to token colors in the equivalent Coloured Petri net (Figure 6).

4.2.3. Object Petri Nets

Object Petri Nets can be considered as a use of Predicate-Transition Petri Nets in context of an object approach. The tokens are not constant anymore, but n-tuple instances of object classes. An object class is defined by an attribute set, called properties, and an operation set, called methods. Among each class attributes, there is an implicit attribute containing the name of the place where the object is located. The classes are just definitions. The objects (tokens) are instances of classes. The transitions are associated to the methods, which change the object properties. An operation associated to a transition t only could be executed by a method if it is located in an input place of t (Cardoso and Valette, 1997).

Object Petri Nets enhance the description power of Predicate-Transition Petri Nets as it permits to describe many attributes for each token. In a complex manufacturing system, any information about each part may be defined by the corresponding object attributes.

In the cited example, the part class may describe if the part is done or it is a raw material, its type and operation to be done to produce the part. The Petri Net transitions change these attributes as they are fired.

Table 1 shows the model classes.

Table 1 – Classes definition.

Robot	Class name
Properties	
Name :string	Robot description
Place :integer	Place where object is located
Methods	
StartProgram	Start robot program (transitions t_2 and t_4)
EndProgram	End of robot program (transitions t_3 and t_5)

Machine	Class name
Properties	
Name :string	Machine description
AvailableOperations :set of operations	Available operations (or programs)
Place :integer	Place where object is located
Methods	
PrepareMachine	Prepares machine to receive part (transition t_2)
StartProgram	Start machining program (transition t_3)
EndProgram	End of machining program (transition t_4)

Part	Class name
Properties	
Name :string	Part description (serial number etc.)
Type :set of parttype	Type of the part
Operation :operations	Operation to be done in part
Done :boolean	Indicates whether the part is done or it is raw
UsedRobot :Robot	Robot used during transport
UsedMachine :Machine	Machine used during process
Place :integer	Place where object is located
Methods	
Create	Instantiates object (transition t_1)
TransportToMachine	Raw part is transported into machine (transition t_2)
StartProgram	Start program in machine (transition t_3)
TransportOutOfMachine	Produced part is transported out of the machine (transition t_4)
OutOfCell	Produced part is placed in cell output (transition t_5)

Class properties are declared as simple integer, boolean or string variables (Name, Done, Place), variables of user-defined types as ‘operations’ and ‘parttype’ or even referenced to other classes (‘Robot’ and ‘Machine’). Properties as ‘Machine.AvailableOperations’ and ‘Part.Type’ are defined as sets of user-defined variable types.

If the part is to be processed in other cells in a major system, property Part.Operation may be defined as an array of ‘operations’, defining the sequence of operations to be done.

Figure 8 shows the machining cell model using Objects Petri Net. The Petri Net marking means that there is a part to be processed and the robot and the two machines are idle.

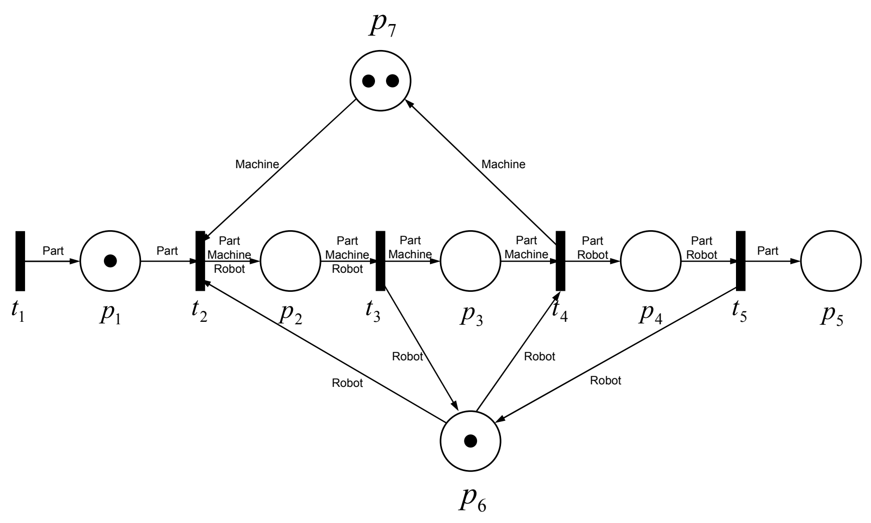


Figure 8 – Model of the machining cell using Object Petri Nets.

Transition t_1 instantiates objects of the class Part in place p_1. When one object Part is instantiated, it includes the attributes values: part name, part type and the operation to be done in it.

Transition t_2 fires when one object Part is instantiated, one object Robot exists in place p_6, one object Machine exists in place p_7 and the attribute Operation in the object Part has a value present in the set AvailableOperations of the machine. The part is then transported by the robot (method TransportToMachine is executed in object Part, method StartProgram is executed in object Robot and method PrepareMachine is executed in object Machine).

Transition t_3 fires when the robot finishes the transport by executing method EndProgram in Robot object and StartProgram in machine and Part objects).

Transition t_4 fires when the machine finishes the program and the robot is available (in place p_6). Then, method StartProgram is executed in object Robot, EndProgram is executed in object Machine and TransportOutOfMachine is executed in object Part.

Finally, transition t_5 fires when the robot finishes the transport and method EndProgram is executed in object Robot and OutOfCell is executed in object Part.

When the methods are executed, they change the properties of the respective objects. In the class Machine, method PrepareMachine (transition t_2) changes value of Place from p_7 to p_2, method StartProgram (transition t_3) changes this value from p_2 to p_3 and method EndProgram (transition t_4) changes it from p_3 back to p_7.

As another example, when method TransportToMachine (transition t_2) is executed in class Part, it changes value of Place p_1 to p_2. It also sets values of UsedRobot and UsedMachine to the instances of Robot and Machine in order to associate the resources to the Part (it is done to avoid conflicts when both machines are producing parts).

5. CONCLUSIONS

In this paper the modeling of a real Flexible Manufacturing System cell using High Level Petri Nets in order to evaluate its applicability to real systems modeling was shown.

Coloured Petri Nets and Pedicate-Transition Petri Nets were adequate to handle part types and machining operations. However, if the modeled system is subsystem of a major one and the parts have to be processed by more than one machine, the number of token colors (or variable values) may became large as it is a combination of all processes do be done.

The model built using Object Petri Nets was able to include all the necessary properties of the produced parts in a much concise way. In the example, only two machines are used to process the parts. However, if a new machine is to be used, it will not increase the model complexity, as it will only be instantiated a new object for this machine. Also, if a new set of parts is to be produced, it will only affect the set of types and available operations in the defined classes.

6. ACKNOWLEDGEMENTS

Authors would like to thank SENAI-CIMATEC and FAPEMIG for the support in developing this work.

7. REFERENCES

Cardoso, J. e Valette, R., 1997. "Redes de Petri". Ed. da UFSC, Florianópolis.

Erschler, J. and Tersac, G., 1988. "Flexibilité et rôle de l'opérateur humain dans l'automatisation intégrée de production" Rapport LAAS, No 88173.

Genrich. H., 1987. "Predicate/transition nets. Lecture notes in Computer Science" Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986. Part I, v.254., p.207-247.

Jensen, K., 1997. "Coloured Petri nets: basic concepts, analysis methods and practical use". 2.ed. Springer-Verlag, Berlin.

Lakos, C., 1995. "From Coloured Petri Nets to Object Petri Nets", Conference on the Application and Theory of Petri Nets, Torino, Italy.

Lauzon, S.C., Ma, A.K.L., Mills, J.K., Benhabib, B., 1996. "Application of discrete-event-system theory to flexible manufacturing", Control Systems Magazine, IEEE, 16(1), pp. 41-48.

Lima, E.A. and Dorea, C.E.T., 2002. "An algorithm for supervisory control of discrete event systems via place invariants" Proceedings of the 15th IFAC World Congress. Barcelona, Spain.

Lima II, E.J. and Dorea, C.E.T., 2004. "Synthesis and PLC Implementation of Supervisory Control Via Place Invariants for a Manufacturing Cell" In: INCOM 2004, Salvador - BA. Proceedings of the 11th IFAC Symposium on Information Control Problems in Manufacturing. Elsevier.

Moody, J.O. and Antsaklis, P.J., 1998. "Supervisory Control of Discrete Event Systems Using Petri Nets", Kluwer Academic Publ., Boston.

Murata, T., 1989. "Petri Nets: Properties, analysis and applications". Proceedings of the IEEE, 77(4), pp. 541–580.

Petri, C.A., 1962. "Kommunikation mit automaten" Bonn: Institut für Instrumentelle Mathematik, Schriften des IMM Nr. 3.

Queiroz, M.H. and Cury, J.E.R., 2000. "Modular Control of Composed Systems" Proceedings of the American Control Conf.

Ramadge, P.J.G. and Wonham, W.M., 1989. "The Control of Discrete Event Systems" Proceedings. of the IEEE, 77(1), pp.13-30.

Silva, M. and Valette, R., 1989. "Petri Nets and Flexible Manufacturing" Lecture Notes in Computer Science, pp. 374-417.

Villani, E., 2004. "Modelagem e Análise de Sistemas Supervisórios Híbridos". Doctoral thesis – Universidade de São Paulo, São Paulo.

Yamalidou, K., Lemmon, M. and Antsaklis, P.J., 1995. "Feedback Control of Petri Nets Based on Place Invariants", Automatica 32(1), pp. 15-28.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.