2003

Adaptative models of flexible manufacturing systems

Campos-Rodríguez, Raúl; Alcaraz-Mejía, Mildreth; López-Mellado, Ernesto; Ramírez-Treviño, Antonio

Campos-Rodríguez, R.; Alcaraz-Mejía, M.; López-Mellado, E.; Ramírez-Treviño, A. (2003) Adaptative models of flexible manufacturing systems. IMACS Multiconference Computational Engineering in Systems Applications (CESA), At Lille, France, Volume: Symposium on Discrete Events in Industrial and Manufacturing Systems.

Adaptive Models of Flexible Manufacturing Systems*

R. Campos-Rodríguez, M. Alcaraz-Mejía, E. López-Mellado, A. Ramírez-Treviño

CINVESTAV-IPN Unidad Guadalajara.
Prol. López Mateos Sur 590, 45090 Guadalajara, Jal. México
elopez@gdl.cinvestav.mx

Abstract— This paper deals with the rapid reconfiguration of task controllers of Flexible Manufacturing Systems (FMS). This problem is concerned with the ability of the task coordination system to adapt dynamically to new situations issued from changes on the production requirements or from failures. A formalism allowing structural changes into an execution model of the controller expressed as a Petri net (PN) is proposed; it can be considered as a graph rewriting system that permits the modification of a PN adding, deleting, and replacing subnets. The definition of such formalism called RW Nets is presented and its application is illustrated through two simple case studies.

Keywords— Flexible Manufacturing Systems; Task modelling; Adaptive models; RWNets.

1. Introduction

Nowadays, flexibility is a key feature in Manufacturing Systems; it means the ability for adaptation to new situations issued from reprogramming requirements and/or from failures. In order to cope with this requirements coordination systems must be rapidly reprogrammed; thus, the coordination software must be based on models that allow the representation of operation sequences and the resource management.

In this work, the approach held for representing the execution model is PN based. PN has been widely adopted as description formalism for the specification of large and complex systems in the FMS community [1], because of its characteristics issued from its graphical nature and its simple mathematical support: clearness and compactness to represent complex behaviour, namely causality, concurrence, parallelism, synchronization, decisions, and information exchange.

The reconfiguration problem can be stated as "perform any change at any moment within a minimum time delay". This is very hard to accomplish, thus several constraints can be relaxed. First, at all, one must distinguish between changes imposed by modifications on the production requirements and changes inherent to recovery actions from failure situations. In failure recovery, the time is more critical than production reprogramming; thus, each case is addressed in different way concerning the instants when the changes can occur and the time response.

*This work has been supported by project CONACyT U41968Y

In this work, it is proposed the use of Petri nets for the modelling of the activity flow, and RWNets [13] for the reconfiguration of the Petri net model by adding, deleting, and replacing subnets; it is an extension of mobile nets [2], dynamic nets [3], and reconfigurable nets [4] which are classes of high level Petri nets. The definition of Rewriting (RW) Nets is presented and a scheme for its application to task reprogramming and fault recovery is proposed.

The paper is organized as follows: section 2 states the problem of task reconfiguration. Section 3 introduces the definition of RWNets. Finally, section 4 presents a couple of examples for illustrating the use of RWNets in dynamic task reconfiguration.

2. Coordination of manufacturing tasks

2.1 Task coordination

In FMS literature, task coordination is usually considered as the next upper level of the local control level into a hierarchical control scheme. Task coordination is supported, in most of the cases, on an execution model that describes mainly flow of the material, the activity sequences, and the resource management into the manufacturing system.

The task execution model can be described by PN, which can be implemented as the kernel of a coordination software. In order to address complex task coordination, development tools allowing edition, validation, and real-time software generation are suitable [5].

2.2 Adaptive models

In task reconfiguration, one must distinguish between changes imposed by modifications on the production requirements and changes inherent to recovery actions from failure situations. In failure recovery, the time is more critical than production reprogramming; thus, each case is addressed in different way according to the instants when the changes can occur and the required time response.

Figure 1 illustrates the case in which a piece of a task execution model must be modified: the operations A and B that were sequentially performed must be executed in

parallel. In figure 2 it is illustrated the case in which a failure during the execution of the activity A leads to an unknown state from which a recovery operation R must be performed to bring the controller back to an operational state into the original activity flow; so the new path is added to the model. This process can be considered as a learning mechanism.

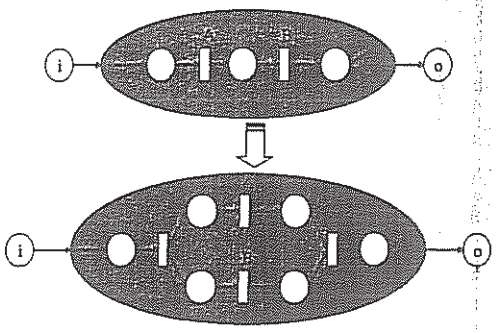

Figure 1. Production reprogramming

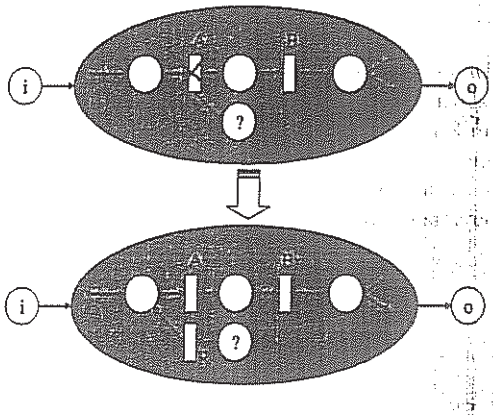

Figure 2. Failure recovery

In FMS two kinds of failures can occur: time-dependent and non time-dependent.

In time-dependent cases, failures include the violation of the maximum and minimum time constraints between events, and duration failures due to a condition or set of conditions, which cannot hold a specific amount of time [7]. In non-time-dependent cases, failures can be present when a required event does not occur, when an undesired event occurs, or when two incompatible events occur simultaneously.

There exist several techniques addressing different types of failures [8]. In order to illustrate the fault tolerance ability in FMS using adaptive models, backward error recovery is considered for our example, which is similar to one of the four methods discussed in [9], [10], and [6].

Backward error recovery method assumes that, under a normal functioning state, a task is executed causing a new faulty state in our system. Nevertheless, this faulty state can become normal state after a sequence of operations that are applied from the faulty state carrying it back to a last normal functioning state.

3 Adaptive execution models

In this work, it is proposed the use of Petri nets for the modelling of activity flow and RWNets [13] for the reconfiguration of the Petri net model by adding, deleting, and replacing subnets; it is an extension of Mobile nets [2], Dynamic nets [3], and reconfigurable nets [4] which are classes of high level Petri nets.

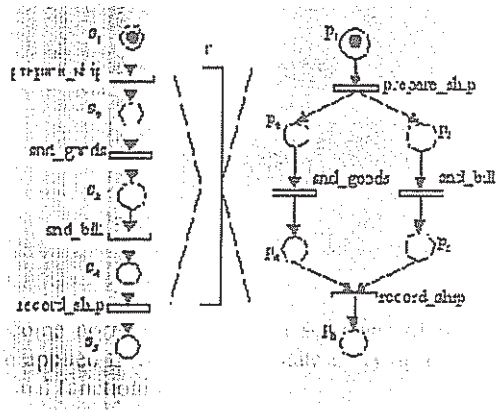

Figure 3. Dynamic changes into a PN model.

3.1 R-W nets

The RWNets allow realizing dynamic changes into a PN structure (see figure 3) [13]. The firing of transition r consumes the left side PN, which shows two activities, snd_goods and snd_bill, within a causal relationship. The firing of transition r produces the right side PN, which establishes the change of these two sequential activities into two parallel ones.

Definition 1. The set of marked Re-Writing nets (MRWnets) is defined recursively as follows:

a) A marked PN is a MRWnet

b) The pair (N, μ), where N=(T, ∂_- , ∂_+) such that: T is a finite set of transitions, ∂_- , $\partial_+ : T \rightarrow$ MRWNets are the Pre and Post functions, respectively, (the transitions consume and produce RWNets), and where μ∈RWnets, is called the initial marking, is an RWNet.

Given a t∈T, $\partial_-(t)$ is often called the pre set of t, and $\partial_+(t)$ is often called the post set of t.

In order to define the transition enabling, it is necessary to define Subnets and Isomorphism concepts for RWNets. Intuitively, a subnet can be seen as a "part of the net". In figure 4, considering the place names it is easy to see that W is a subnet of Y.

Definition 2. Subnets in RWnets (see figure 4). A RWNet (N', μ') is a subnet of another RWNet (N, μ), denoted by (N', μ') ⊆ (N, μ), iff

a) Its transitions are a subset of the set of transitions in the other RWNet. (T'⊆ T).
b) Its Pre function is a subset of the Pre function of the other one, restricted to its set of transitions ($\partial_-' \subseteq \partial_- |_{T'}$).
c) Its Post function is a subset of the Post function of the other one, restricted to its set of transitions ($\partial_+' \subseteq \partial_+ |_{T'}$).
d) And recursively its Pre and Post functions are subsets of the pre and post functions of the other RWNet, respectively, for all its transitions. ($\partial_i'(t') \subseteq \partial_i(t')$, $i \in \{+,-\}$, $\forall t' \in T'$).

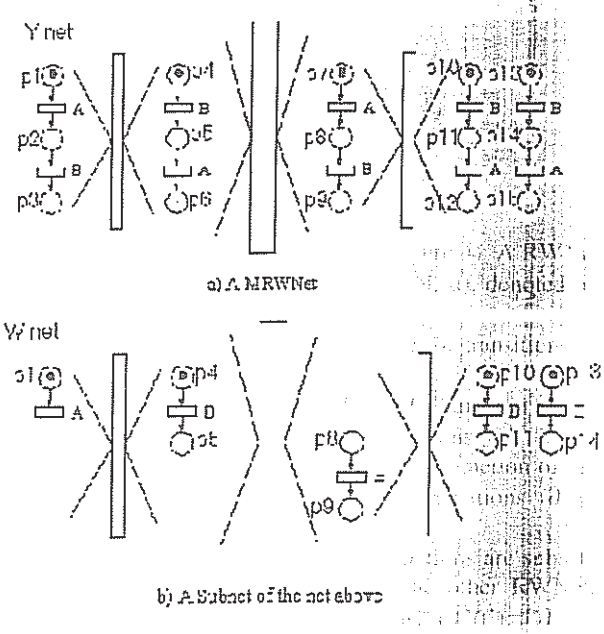

Figure 4. A subnet in RWNets

Intuitively, we can establish that two RWNets are isomorphic if and only if they can be rewritten in such way that they look similar and, recursively, their pre and post set.

Definition 3. Isomorphism. Two RWNets, (N, μ) and (N', μ') are isomorphic, denoted (N, μ) ≡ (N', μ'), iff there exists a pair of isomorphism <f, g>, where f: T → T', g: RWNets → RWNets, such that $\partial_i' \circ f = g \circ \partial_i$, $i \in \{+,-\}$, and, recursively $g(\mu) \equiv \mu$ and $\partial_i' \circ f(t) \equiv \partial_i(t)$, $\forall t \in T$. (See figure 5).

Definition 4. Binding. Let (N,μ)∈MRWNets, where N=(T, ∂_- , ∂_+). Let t∈T; a binding **b** for $\partial_-(t) \in$ MRWNets is a function **b**: MRWNets → MRWNets, such that $\partial_-(t) \equiv$ **b**($\partial_-(t)$).

Definition 5. Substitution. Let **b** be a binding. Let <f, g> be the pair isomorphism defined by **b**. For a (X,μ)∈MRWNets, (X,μ)[**b**] denotes (X,μ)<f,g>, i.e., the application of the pair isomorphism <f,g> to (X,μ).

Now, with these concepts previously introduced, it will be defined the transition enabling and the firing rule. Intuitively, a transition in MRWNets is enabled if and only if there exists a matching for its pre set in the current marking. Recall that the marking is, in fact, a MRWNet.

Definition 6. Transition enabling. Let (N,μ)∈MRWNets, such that N=(T, ∂_- , ∂_+). Let t∈T; t is enabled at μ under the binding **b**, denoted (N, μ)[t>$_b$, iff there exists a binding **b** for $\partial_-(t)$, such that **b**($\partial_-(t)$)⊆ μ .

The firing rule is defined as the removing of a structure and a marking by the action of the pre set of a transition, and the adding of structure and marking by the action of the post set of such a transition.

For this purpose, first of all, it is performed the difference between the pre set and the post set which represents the structure and marking that must be removed from the current marking, and which also may represents the structure and marking that must be added to the current marking. Remember that the current marking is an RWNet. The formal definition is given below.

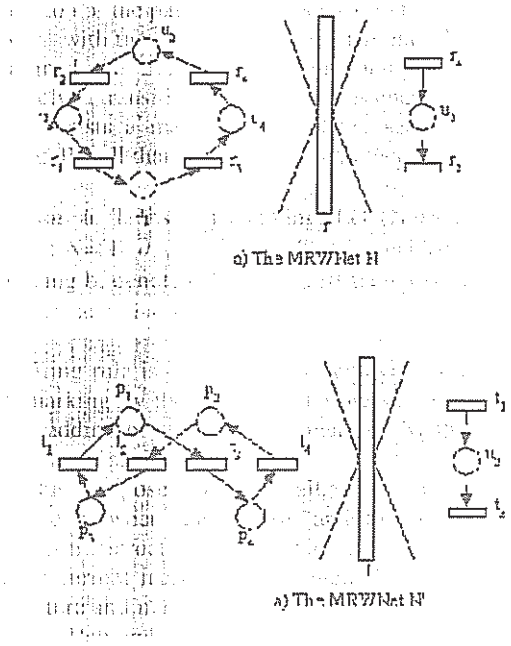

Figure 5. Two isomorphic nets.

Definition 7. Firing Rule. Let $\_[\rangle_b$, $\_ \subseteq$ MRWNets $\times$ MRWNets, be the smallest substitutive relation generated by:

$$(N, \mu)[t\rangle_b \Rightarrow (N, \mu), (N, \mu') \in \_[\rangle_b \_$$

where $\mu' = \mu \backslash (\partial_-(t)[b] \backslash \partial_+(t)[b]) \oplus (\partial_+(t)[b] \backslash \partial_-(t)[b])$, and $\backslash$ and $\oplus$ are the natural extension of difference and addition operation for graph to PN.

In figure 6.a, there is an enabled rewriting transition, since, in figure 6.b there exists a matching for its pre set.

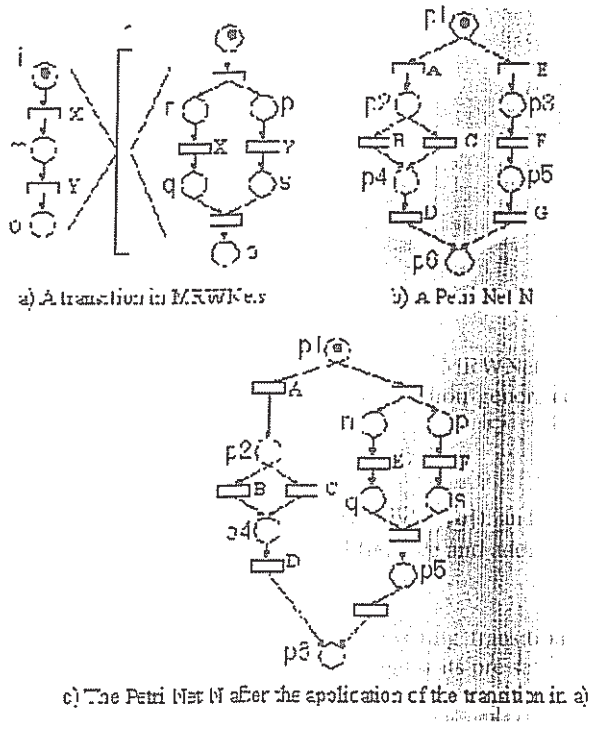

Figure 6. Transition enabling and Firing rule

In figure 6.c is shown the result of firing the enabled transition in figure 6.a as described in definition 5. The pre set in transition r formed with i, X, m, Y, o, is isomorphic to the subnet formed with p1, E, p3, F, p5, where i ≡ p1, X ≡ E, m ≡ p3, Y ≡ F, o ≡ p5. So, r can be fired and replace the subnet on the pre set of r with the subnet on the post set of r. Where i goes over p1, and o over p5, as can be seen in figure 6.c. In the remainder of this paper, any transition will be written separating the marking and the structure.

3.2 Dynamic reconfiguration

The application of RWNets to task reprogramming and fault recovery is proposed. For this purpose, RWNets can model the following mechanisms:

a) PN rewriting system. In form of rules, pieces of PN can be replaced by other PN. The pre and the post conditions of a transition (rule) are both PN structures (see figure 7.a).

b) The Pre is a reachable marking of a PN describing a task; the Post is a PN structure that must be added to the original net for recovery purposes (see figure 7.b).

4 Modelling manufacturing tasks

In order to analyse the RWNets, two examples are presented below. One example considers fault recovery and the second one considers task reprogramming.

4.1 Example 1: Failure recovery

• Scenario

The scheme of a simple FMS that processes one kind of parts (work pieces) is depicted in figure 8; it comprises the following components: three storage pallets (a, b, c), two robots (d, g), two rectifiers of flat surfaces (e, f; emery), and two drills (h, i) [5].

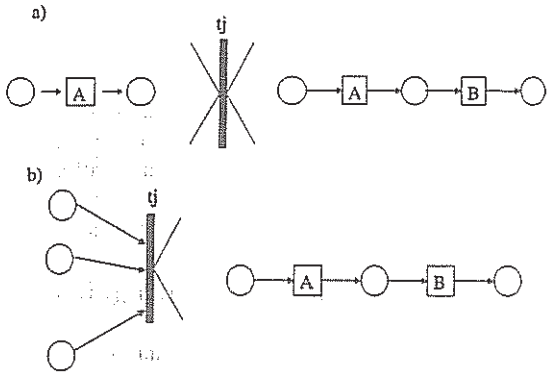

Figure 7. RW mechanisms.

The basic steps performed into the system are the following:

1) If there are "a", the robot "d" can take one, then the robot load the part into any of parts in the pallet the two rectifiers "e" or "f".
2) After the operations in the rectifiers "e", "f", the robot "d" unloads the part and put it on the pallet "b".
3) If there are parts in the pallet "b", the robot "g" can take one, then the robot load the part into any of the two drills "h" or "i".
4) After the operations in the drills "h", "i", the robot "g" unloads the part and put it on the pallet "c".

Pieces in the pallet "c" are finished pieces.

• PN model

The PN modelling the previous tasks can be obtained as shown in figure 9. There are 17 places representing the availability of resources or the operations, and there are 12 transitions, which represent the start and/or completion of operations, as described below:

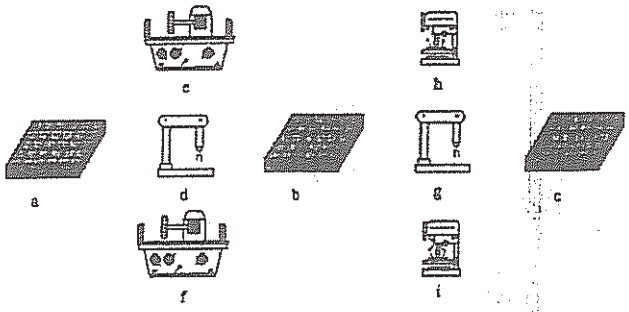

Figure 8. FMS layout.

t_1: Robot "d" moves the part from pallet "a" into the rectifier "e".
t_2: Robot "d" moves the part from pallet "a" into the rectifier "f".
t_3: The rectifier "e" performs its task on the part.
t_4: The rectifier "f" performs its t ask on the part.
t_5: Robot "d" moves the part from the rectifier "e" into pallet "b".
t_6: Robot "d" moves the part from the rectifier "f" into pallet "b".
t_7: Robot "g" moves the part from pallet "b" into the driller "h".
t_8: Robot "g" moves the part from pallet "b" into the driller "i".
t_9: Drill "h" performs its task on the part.
t_{10}: Drill "i" performs its task on the part.
t_{11}: Robot "g" moves the part from the driller "h" into the pallet "c"
t_{12}: Robot "g" moves the part from the driller "i" into the pallet "c".
la': Represents the rectifier of flat surfaces "e".
lb': Represents the rectifier of flat surfaces "f".
lc': Represents the robot "d".
ld': Represents the driller "h".
le': Represents the driller "i".
lf': Represents the robot "g".

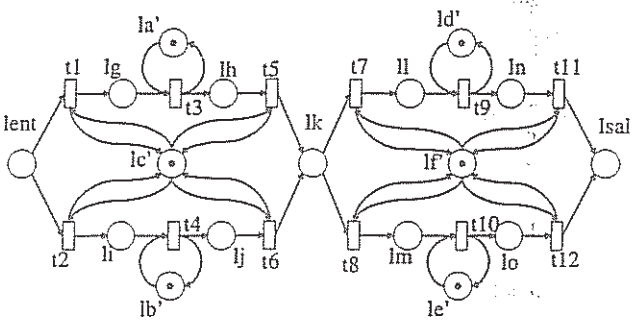

Figure 9. PN modelling the FMS

•RW rules

In the FMS several faulty states can exist. For example, in the place la' where the driller "e" is supposed to perform certain task on the part, for any reason, the driller task cannot be completed causing a fault on the system. To implement error recovery in the system due the case just mentioned, a few changes are made to the system model converting it into an RWNet=(N,μ). μ is shown in figure 10 and N is shown in figure 11.

The net N=(T, ∂_-, ∂_+), has only two rewriting transitions: the transition s = {{ta={lnt,•}→{(la,•)}, t={(la,•),(l,•)}→{(l,•)}, tb={(l,•)}→{(lb,•)}} ⊗ {(lnt,v), (la,u), (lb,•)}} →{lnt ⊗ (lnt,u+v)}, and the transition r = {(ll,v) →(lk,v)}, and its marking is a simple Petri Net μ = (N', μ'), where N'={T', ∂_-',∂_+'}, is defined:

N'= {t1={(lent,•),(lc',•)}→{(lg,•),(lc',•)},
t2={(lent,•),(lc',•)}→{(li,•),(lc',•)},
t3={(lg,•),(la',•)}→{(lh,•),(la',•)},
t4={(li,•),(lb',•)}→{(lj,•),(lb',•)},
t5={(lh,•),(lc',•)}→{(lk,•),(lc',•)},
t6={(lj,•),(lc',•)}→{(lk,•),(lc',•)},
t7={(lk,•),(lf',•)}→{(ll,•),(lf',•)},
t8={(lk,•),(lf',•)}→{(lm,•),(lf',•)},
t9={(ll,•),(ld',•)}→{(ln,•),(ld',•)},
t10={(lm,•),(le',•)}→{(lo,•),(le',•)},
t11={(ln,•),(lf',•)}→{(lsal,•),(lf',•)},
t12={(lo,•),(lf',•)}→{(lsal,•),(lf',•)},
tz_1={(la',•)}→{(lz_1,•)}, tz_2={(ld',•)}→{(lz_2,•)},
tz_3={(lb',•)}→{(lz_3,•)}, tz_4={(le',•)}→{(lz_4,•)} },
and μ'= {(la',•),(lb',•),(lc',•), (ld',•), (le',•), (lf',•)}.

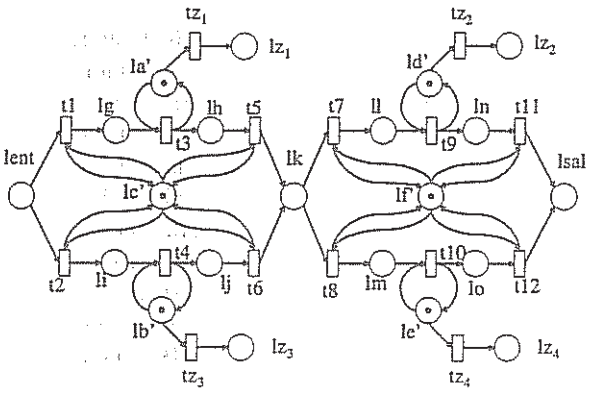

Figure 10. μ ∈ RWNet that models the system.

This transitions work in the following way: the transition s is enabled where a token is present in the place lb, and any number (bounded) of tokens are present in the places lnt and la, besides, it is necessary a matching for the net structure as shown in figure 11a.

On the other hand, the transition r, figure 11b, maps any marking present in the place ll to the place lk, performing a kind of 'restarting" of processes.

Notice that in this rewriting transition, there are no structure, which means that this transition can perform this operation in the particular case of the places ll and lk. In this way, is not necessary a matching for its pre set.

In case an error occur on la', under a certain conditions, the transition tz_1 is fired which leads to a faulty state which is just a mark in the place lz_1. At this point, the transition s is enabled; figure 12 shows the result of its application.

4.2 Example 2 task reprogramming

- **Scenario**

Consider a cell composed by a milling machine 'm1", two conveyors, one incoming "i" and one outgoing "o", and a robot "r" [14]. Consider a product type that needs only milling. The basic steps performed into the system are the following:

1. Robot "r" takes a raw part from the incoming conveyor "i", and loads it at "m1".
2. "m1" performs its tasks on the part.
3. Robot "r" unloads the part from "m1" and moves it on the outgoing conveyor "o".

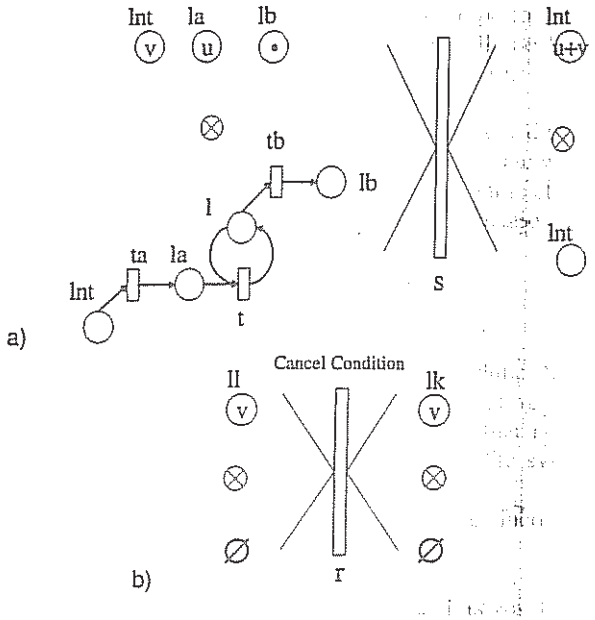

Figure 11. N ∈ RWNet that models the system.

- **PN Model**

The PN describing the previous tasks can be obtained as shown in figure 13. This model is a binary state machine. There are seven places representing the availability of resources or the operations, and there are four transitions, which represent the start and/or completion of operations, as described below:

P_1: Conveyor in "i".
P_2: Robot available "r".
P_3: Part Loaded.
P_4: Mill available "m1".
P_5: Milling operation.
P_6: Part unloaded.
P_7: Conveyor out "o".
t_1: Robot "r" takes a raw part from the incoming conveyor "i", and loads it at "m1".
t_2: "m1" starts its task on the part.
t_3: "m1" ends its task on the part.
t_4: Robot "r" unloads the part from "m1" and moves it on the outgoing conveyor "o".

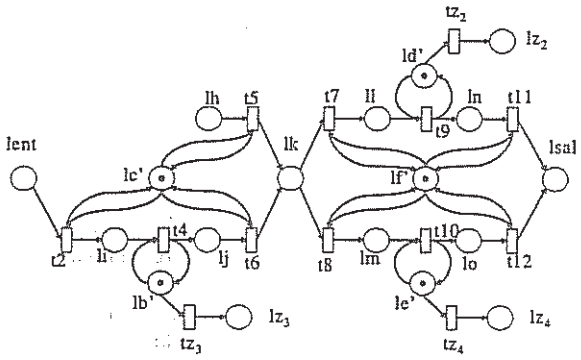

Figure 12. Resulting Model

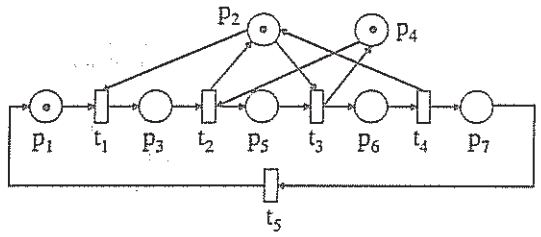

Figure 13. Task model

- **RW Rules**

Let suppose the operational specifications change in such a way that the raw part must be first drilled and then milled. The RW Rule considered to make this change is shown in figure 14. Changes made by the RW Rule can be observed on figure 15, where the new places describe:

P_8: Part loaded
P_9: Drill available
P_{10}: Drilling operation

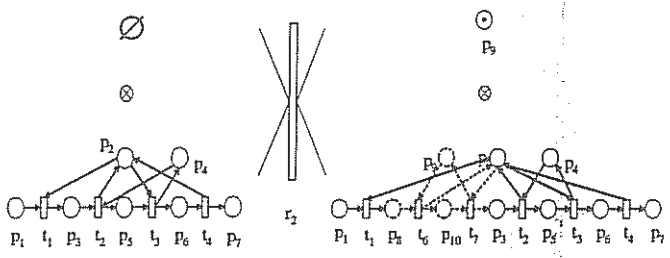

Figure 14. RW Rule to reprogram the system.

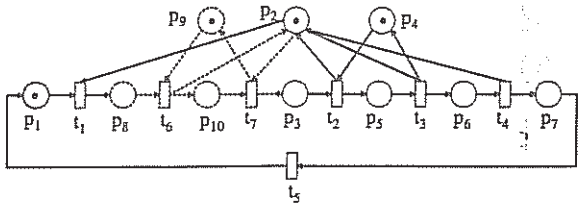

Figure 15. Resulting Model

Conclusions

This paper presented the definition of RW Nets and its application to the modelling of adaptive coordination systems for manufacturing processes. It has been shown that RW Nets provide the mechanisms (addition, deletion, and replacing of subnets) needed for the dynamic reconfiguration of Petri net models. This feature gives to a coordination system the ability to cope with changes imposed by modifications on the production requirements or changes inherent to recovery actions issued from failure situations.

References

[1] DiCesare, F. G.Harhalakis, J.M.Proth, M.Silva, F.B.Vernadat. Practice of Petri Nets in Manufacturing. Chapman & Hall. 1993

[2] M.Buscemi and V. Sassone High-Level Petri Nets as Type Theories in the Join Calculus. FOSSACS01. Ed. F. Honsell and M. Miculan. Num. 2030. Serie: LNCS. Pags: 104-120. Springer 2001.

[3] A. Asperti, A., and Busi, N., Mobile Petri Nets. Technical Report UBLCS-96-10, University of Bologna, Italy (1996).

[4] E. Badouel and J. Oliver. Reconfigurable nets: a class of high level Petri nets supporting dynamic changes. WFM' 98, CSR 98/07Lisbon, 1998.

[5] A.Gutiérrez, E. López, A. Ramírez. Specification and synthesis of control software for flexible manufacturing systems. Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics, Nashville, USA, Oct. 2000. pp.1703-1708

[5] C. Ortega. Fault Tolerance Systems in FMS [in spanish]. M.Sc. Thesis. Cinvestav-IPN. Guadalajara, Jal., México, 1998.

[6] M. C. Zhou and F. Dicesare. Adaptive Design of Petri Net Controllers for Error Recovery in Automated Manufacturing Systems. IEEE Transactions on Systems, Man and Cybernetics, Vol. 19, No. 5, pages 963-973. September 1989.

[7] J. de Figuereido and A. Perkusich. Faults and Timing Analysis in Real-Time Distributed Systems. Int. Journal Fuzzy Sets and Systems, Vol. 83, N° 3, 1996. North-Holland.

[8] R. Isermann. Supervision, fault-detection, and fault-diagnosis methods and introduction. Control engineering practice, Pergamon press,.Vol. 5, pp. 639-652, 1997.

[9] P..J. Fielding, F. DiCesare, and G. Goldbogen. Program augmentation for error recovery in automated manufacturing: A formulation. 1988 Proc. of the IEEE Int. Conf. on SMC, 1988.

[10] P.J. Fielding, F. DiCesare and G. Goldbogen. Error recovery in automated manufacturing through the augmentation of programmed processes. Journal on Robotics Syst., Vol 5, N° 4. 1988.

[11] C. Fournet and G. Gonthier. The Reflexive CHAM and the Join-Calculus. ACM Press, pag. 372-385, 1996.

[12] J. Zhibin, Zou M. J., Fung R. Y.K., Tu P.l Y.L., Colored Petri Nets with changeable structures (CPN-CS) and their applications in modelling one-of-a-kind production (OKP) systems, Computers and Industrial Engineering (41)3 (2001) pp. 297-308.

[13] R. Campos. Modular and Adaptive Modeling of Workflow Management Systems. MSc. Thesis. CINVESTAV, U. Guadalajara. México, Oct. 2002.

[14] M. Zhou, and K. Venkatesh. Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach. Series in Intelligent Control and Intelligent Automation, Vol. 6, World Scientific, 1999.