THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2000 ISSN 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS

SUPERVISORY CONTROL TECHNIQUE FOR AN ASSEMBLY WORKSTATION AS A DYNAMIC DISCRETE EVENT SYSTEM

Daniela Cristina CERNEGA and Viorel MÎNZU

"Dunarea de Jos" University of Galati, România
Str. Domneasca, No. 111, 6200 Galati, România
e-mail:Daniela.Cernega@ugal.ro, Viorel.Minzu@ugal.ro

Abstract. This paper proposes a control problem statement in the framework of supervisory control technique for the assembly workstations. A desired behaviour of an assembly workstation is analysed. The behaviour of such a workstation is cyclic and some linguistic properties are established. In this paper, it is proposed an algorithm for the computation of the supremal controllable language of the closed system desired language. *Copyright © 2001 IFAC.*

Keywords: discrete event dynamic systems, assembly systems, supervisory control

1. INTRODUCTION

This paper deals with a control problem for an assembly workstation using the *supervisory control* technique proposed by Wonham et al (see Ramadge and Wonham, 1987; Wonham and Ramadge, 1987; Ramadge and Wonham, 1989).

To control an assembly system means to execute a preplanned assembly process, taking into account the mutual exclusion, the concurrence of tasks and the cyclical usage of resources.

A supervisor is an automaton that is connected with the controlled discrete event system. Therefore, a closed loop system is formed. A desired behaviour of the discrete event system may be expressed by sequences of events, which form a language. Generally, a supervisor that assures the desired behaviour exists, if two conditions are met (see the supervisor existence theorem presented in Ramadge and Wonham, 1989). The first one is the L_m-closure of the language mentioned before and the second is its controllability.

Formal properties of a language that models the behaviour of the assembly workstation as a discrete event system (DES) are used in section 3 to define a class of DES with cyclic working.

In section 4 is proposed an algorithm for the computation of the supremal controllable language of the closed system desired language and an example is shown.

2. SUPERVISORY CONTROL PROBLEM

A discrete event system is a dynamic system with a discrete state space. The time instants at which state transitions occur are, in general, unpredictable (Ramadge and Wonham, 1987).

The state transitions of a discrete event system are determined by *events*. The occurrence times of the events are ignored in order to simplify the models considered in this paper, in which the timing information is not crucial. In such models listing (in order) the events that occur along the state trajectory specifies a state trajectory. This lead to so call *logical* DES *models* (Ramadge and Wonham, 1989), in

This paper was recomended for publication by S. Caraman

which the interest is oriented to the sequences or strings of events that the process responds at. A string of events leading to a specified state of the system can be considered as an entry signal.
The set of all the physically possible sequences of events indicates *the possible behaviour* of the discrete event system. This behaviour can be modelled with a formal language *L*.

The approach in supervisory control problem regards the discrete event system to be controlled, i.e. the 'plant' in traditional control terminology, as a generator, ***G***, of the formal language *L*.

The generator ***G*** can be defined as an automaton:

$$\boldsymbol{G} = (Q, \Sigma, \delta, q_0, Q_m),$$

where Q is the *state set*,
Σ is a finite set of symbols referred as *event labels*,
δ: $Q\times\Sigma \rightarrow Q$ is (the partial) *transition function*
q_0 is the *initial state*,
$Q_m \subseteq Q$ is the subset of *marked states*.

The marked behaviour of the generator ***G***, $L_m(\boldsymbol{G})$, is the set of strings which lead to marked states.

Usually a controlled discrete event system has the nonblocking property, i.e. $\overline{L_m(\boldsymbol{G})} = L(\boldsymbol{G})$, where $\overline{L_m(G)}$ denotes the prefix closure of the marked language $L_m(\boldsymbol{G})$.

To control a DES it is necessary that certain events of the system must be disabled (i.e. preventing from occurring) when desired in order to influence the evolution of the system by prohibiting the occurrence of key events at certain times. To model such control the set of events Σ is partitioned into *controllable* and *uncontrollable* events i.e. $\Sigma = \Sigma_c \cup \Sigma_u$. The events in Σ_c can be disabled at any time, while those in Σ_u model events over which the control agent has no influence.

The aim of supervisory control is not to modify L_m, but to achieve a prescribed language $K \subseteq L_m(\boldsymbol{G})$ for the system equipped with the *supervisor* forming together the *closed* system.

Formally, a supervisor is defined in (Ramadge and Wonham, 1989) as a map, *f*, specifying for each string of events the control input to be applied at that point. A supervisor may be also represented (Wonham and Ramadge, 1987) as state realisation of the map *f* which is a pair $\boldsymbol{S}=(S, \Psi)$ where *S* is an automaton and Ψ is the command function.

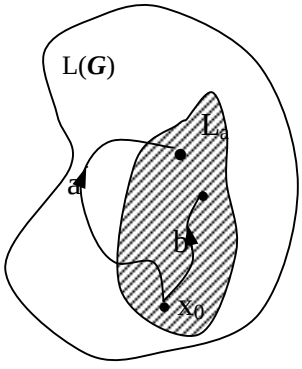

Figure 1 The formal languages $L(\boldsymbol{G})$ and L_a

In this paper the state realisation of a supervisor will be called simply *supervisor*.

Hence, a supervisor is a "controller" that decides in every state of the process (modelled as a generator) which controllable event has to be enabled, and which has to be prevented from occurring in order to achieve the prescribed behaviour.

The prescribed language *K* may be specified directly by giving the closed loop behaviour, or indirectly through the specifications for the closed loop system.

The supervisory control problem will be considered fully solved when it is shown a controller that forces the specifications to be met, exists and it is constructible.

In this paper the prescribed language, *K*, for the closed system is called the *admissible language*.

The relation between the physically possible behaviour of the process ($L(\boldsymbol{G})$) and the admissible language denoted by L_a is shown in figure 1. A state trajectory like "a" in figure 1 has to be prevented from occurring using the control action.

The supervisor which is solution for the supervisory control problem must ensure that the behaviour of the closed loop system (the generator ***G*** equipped with the supervisor ***S***), denoted by $L(\boldsymbol{S}/\boldsymbol{G})$, is identical with the prescribed admissible language $L_a=K$.

To apply supervisory control technique one needs adequate models for controlled process expressed by automata and/or formal languages.

3. ASSEMBLY WORKSTATION MODELS FOR THE PURPOSE OF SUPERVISORY CONTROL

An assembly system is a manufacturing system, which makes a product or a family of products essentially by meeting parts. It is composed by workstations, each one performing one or several tasks.

To execute the preplanned assembly processes, a control problem must be solved taking into account the mutual exclusion of some tasks, their concurrent execution, and so on. In order to solve a control problem, the assembly system must be modelled as a DES, considering all the aspects: parts, robots, fixtures.

In this section, a systematic method to model assembly workstations to be controlled is proposed.

In order to obtain an assembly workstation model to be used for the supervisory control problem, there are some steps to be accomplished.

The construction of the assembly graph. In (Minzu and Henrioud, 1993) the authors have proposed the assembly graph as a model of the assembly process, in the case of single product assembly systems. A model of an assembly workstation using the assembly graph was described in (Minzu and Henrioud,1997). The advantage of using the assembly graph consists in the correct definition of tasks (both the base and the secondary parts are defined) and in the systematic method for its construction.

The construction of the activities-resources graphs. In the assembly graph nodes represent tasks and arrows represent the precedence relation between them. Each task is performed by a resource. The activities-resources graph is obtained adding the resources sequences on the assembly graph as shown by Mînzu and Cernega, 1999. Usually, the task sequences are chosen in order to eliminate any deadlock of the system. A resource is delivered to its first task immediately after the completion of the last task of its sequence. Hence, the resources have a cyclic behaviour that determines the cyclic work in the considered system.

The conversion of the activities resources graphs into the Petri net. As a discrete event system, the assembly workstation is better modelled like a Petri net, because the prerequisites for the execution of any task are more explicit.

In an assembly workstation the working is cyclic and the achievement of the assembly product needs the execution of all the tasks in the assembly workstation. As a consequence the Petri net has a specific property: when the state of the Petri net comes back to the initial marking all the transitions were fired.

The controlled Assembly Petri net is the result of extending the obtained Petri net with external control places. The control problem dictates the positions of the control places.

For example, in figure 2 is shown a controlled Petri net model for a specified control problem. Due to the specifications process behaviour, the transition T_2 has to be a controllable transition. Its firing will be determined with the marking of the control place P_{c1}.

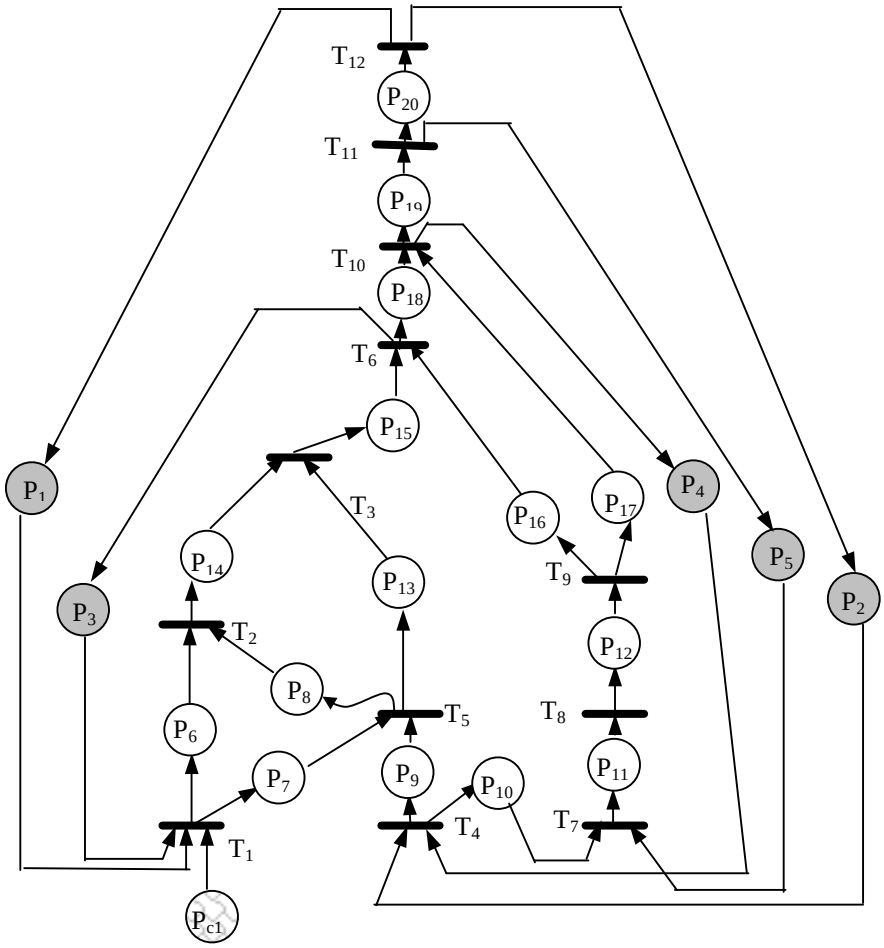

Figure 2. The controlled Petri net for an assembly workstation

The controlled Petri net is converted into *the automaton that generates the physically possible behaviour* of the assembly workstation.

Each state of the automaton is a reachable marking of the controlled Petri net. The events that generate transitions correspond to the firing of the transitions in the controlled Petri net.

The generator ***G*** of the physically possible behaviour of the assembly workstation can be defined as:

$$G = (Q, \Sigma, \delta, q_0, Q_m),$$

where:

- Q is the set of states, each state is a reachable marking in the controlled Petri net,
- $\Sigma = \Sigma_c \cup \Sigma_u$ is the set of events, each event is a transition of the controlled Petri net and the controllable events are corresponding to the controlled transitions,
- δ is the transition function defined in accordance with the transitions between the reachable markings of the controlled Petri net
- q_0 is the initial state corresponding to the initial marking,
- $\{Q_m\} = q_0$ is the single marked state which corresponds to the end of a job in the assembly workstation: in this state a product of the assembly workstation is accomplished.

For example, the generator for the assembly workstation corresponding to the Petri net in figure 2 is shown in figure 3.

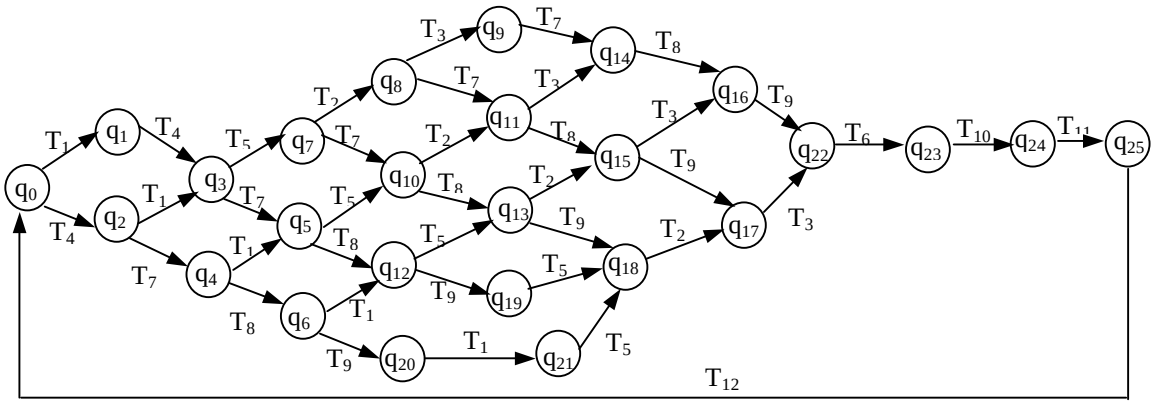

Figure 3 The transition graph for the generator of the considered assembly workstation

The systematic method presented above leads to an automaton, which is an adequate model to be used in the supervisory control problem.

For an automaton ***G*** modelling a system with cyclic working one can remark that the elements of the marked language, $L_m(\boldsymbol{G})$, are the strings of events which determine transitions from the marked state to the marked state. Such an element of the marked language is called *cyclic sequence*. A cyclic sequence containing every transition at most once it is called *minimal cyclic sequence*.

A minimal cyclic sequence for the generator of the behaviour of the assembly workstation contains all the events (transitions of the Petri net), the difference between two minimal cyclic sequences is the order of the occurrence of the events.

For example, the automaton shown in figure 3 has 55 minimal cyclic sequences.

The set of minimal cyclic sequences for an automaton with an unique marked state is:

$$S = \{s_i | \text{ i.e. } q_0 \xrightarrow{s_i} q_0 \}.$$

For the example considered above the marked language can be defined using the minimal cyclic sequences:

$$L_m(\boldsymbol{G}) = (s_1 + s_2 + \ldots + s_{55})^*.$$

A class of DES with cyclic working may be considered to have a single marked state and the initial state identical with the marked state. The generator ***G*** of the assembly workstation belongs to this class of DES.

We can define an automaton type called AWM (Assembly Workstation Model) to characterise all the discrete event systems which have the same characteristics defined above.

Definition 1

An automaton ***M*** defined by:

$$\boldsymbol{M} = (Q, \Sigma, \delta, q_0, Q_m),$$

is called *AWM* (Assembly Workstation Model) if:

- every state of the set Q is accessible and coaccessible (i.e. each state is accessible through a string which can be continued to the marked state);
- it has an unique marked state ($Q_m = \{q_m\}$) identical with the initial state ($q_m = q_0$);
- every minimal cyclic sequence contains all the events in Σ;

- any cycle contains the initial state.

The *AWM* type automaton is a model for an assembly workstation.

For a given assembly workstation ($L(\boldsymbol{G})$) usually the are some specifications to be met during the execution of the assembly process. These specifications lead to the admissible language ($L_a = K$) to be achieved by the controlled system.

There exists a supervisor to ensure the closed loop admissible behaviour described through the formal language $K \subset L_m(\boldsymbol{G})$ if and only if (as stated in the general existence theorem in Wonham and Ramadge, 1987):

- K is L_m-closed (i.e. $\overline{K} \cap L_m(\boldsymbol{G}) = K$, where $\overline{K}$ is the prefix closure of K)
- K is controllable (i.e. $\overline{K}\,\Sigma_u \cap L(\boldsymbol{G}) \subseteq \overline{K}$).

For an assembly workstation regarded like an *AWM* automaton (Mînzu and Cernega, 1999) was proposed a criterion for the L_m-closure.

If the closed loop desired language K is L_m-closed it only remains to verify its controllability. If K is not controllable it will be computed the greatest controllable language included in K which is called the *supremal controllable language* ($\sup\mathbf{C}(K)$) of K.

In the next section we propose an algorithm used for the computation of the supremal controllable language for *AWM* type models.

4. ALGORITHM FOR THE COMPUTATION OF THE SUPREMAL CONTROLLABLE LANGUAGE FOR AWM AUTOMATA

Let K_0 be the admissible closed loop behaviour for an *AWM* type automaton $\boldsymbol{G}$ with the possible behaviour $L(\boldsymbol{G})$ and the marked behaviour $L_m(\boldsymbol{G})$.

The computational algorithm proposed in this section uses the new concepts of *restriction of automaton in relation with another automaton* and *uncontrollable state* defined below.

Definition 2

The automaton $B = (\Sigma, X_B, \xi_B, x_0, X_m)$ is called the *restriction of the automaton* $A = (\Sigma, X_A, \xi_A, x_0, X_m)$, if the following two conditions are met:

i) $X_B \subset X_A$,

ii) $\xi_B(\sigma, x) = \xi_A(\sigma, x)$, $\forall\, x \in X_B$ and $\forall\, \sigma \in \Sigma$, for which $\xi_A(\sigma, x)$ is defined.

Remark that the automaton B is a restriction of the automaton A and the transition function ξ_B is a restriction of the transition function ξ_A.

Definition 3

Let the two automata $A = (\Sigma, X_A, \xi, x_0, X_m)$ and $B=(\Sigma, X_B, \xi, x_0, X_m)$, so that the automaton B is a restriction of the automaton A. A state $x \in X_B$ from the automaton B is called *uncontrollable* state of the automaton B in relation with the automaton A if the following condition is met:

$$\exists\, u \in \Sigma_u \text{ a.î. } \xi_A(u, x) \in X_A - X_B.$$

If the automaton A represents the physically possible behaviour of the system and B represents the admissible behaviour, an uncontrollable state has the meaning of a state from which the admissible behaviour can be exceeded when an uncontrollable event occurs. Hence an uncontrollable state corresponds to the case in which the violation of the constraints can not be prevented from occurring using control action.

Let S_0 be the automaton identical with the generator $\boldsymbol{G}$ defined as follows:

$S_0 = (\Sigma, X_0, \xi_0, q_m, \{q_m\})$,
where $X_0 = Q$,
$\xi_0 \equiv \delta$.

Algorithm SCAW (Supremal Controllable for Assembly Workstation)

Step1. It is constructed the recogniser S_1 for the language $K_0 \subset L_m(\boldsymbol{G})$ defined by:

$S_1 = (\Sigma, X_1, \xi_1, q_m, \{q_m\})$.

Step 2. i = 1

Step 3. It is computed $\overline{C_i}$ the set of uncontrollable states of S_i in relation with S_{i-1}.

If $\overline{C_i} \neq \varnothing$, then go to *Step4*.

else $S := S_i$ and STOP.

Step 4. It is constructed the automaton S_{i+1} by removing from S_i the uncontrollable states and the transitions to them. The automaton S_{i+1} is defined by:

$S_{i+1} = (\Sigma, X_{i+1}, \xi_{i+1}, q_m, \{q_m\})$, where
$X_{i+1} := X_i - \overline{C_i}$.

The automaton S_{i+1} is a restriction of S_i.

Step 5. If $X_{i+1} \neq \varnothing$, then $i := i + 1$ and go to *Step 3*;
else STOP.

Remarks

1. If the language K_0 is controllable the algorithm stops at *Step 1*.
2. Due to the iterative work of the algorithm every automaton S_{i+1} is a restriction of the automaton S_i computed by the previous iteration, S_{i+1} is a restriction of S_0.
3. If the algorithm stops at *Step 5* there is no controllable language included in K_0.

The next theorem proves that the result of the proposed algorithm is the supremal controllable language.

Theorem 1

For the *AWM* type automaton ***G*** and a given language $K_0 \subset L_m(\boldsymbol{G})$ the automaton *S* witch is the result of the algorithm *SCAW* is the recogniser of the supremal controllable language of $K = \sup\mathbf{C}(K_0)$.
The proof of the theorem shows that *K* is a controllable language and there is no controllable language larger than *K* in K_0.

Let us consider the generator ***G*** for an assembly workstation presented in figure 3 where the controllable subset of events is $\Sigma_c = \{T_1, T_8\}$.

A supervisory control problem for this DES asks to design a supervisor to ensure that the states q_9 and q_{14} will be never reached.
Such a supervisor exists if the language of specifications K_0 is L_m-closed and controllable.

A generator S_1 for K_0 given above through the specifications is shown in figure 4.

The language K_0 is not controllable.The supremal controllable language of K_0 is computed with the algorithm *SCAW*.

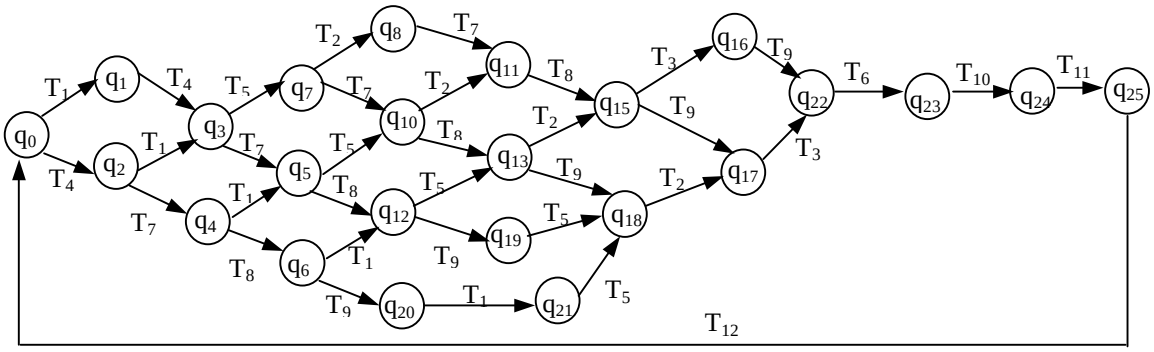

Figure 4. The transition graph of the automaton S_1

The result of the algorithm is the automaton S_3, shown in figure 5, which is the recogniser of the supremal controllable language of K_0.

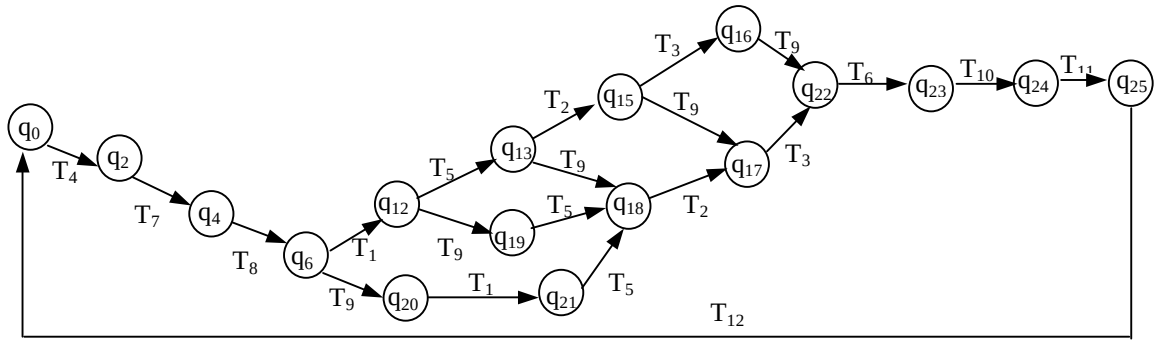

Figure 5. The automaton S_3

5. CONCLUSION

In this paper, a control problem for assembly workstations was presented. A systematic method to obtain an adequate model for the supervisory control problem for an assembly workstation was proposed. This method uses the assembly graph as the starting point and leads to the automaton or the formal language.

It was defined a class of discrete event systems with cyclic working, called *AWM* type automaton, which have the same characteristics with the assembly workstation.

The supervisor existence conditions for *AWM* type automata can be verified using specific methods for this class of DES.

A L_m-closure criterion for the admissible language for *AWM* type automata was proposed in a previous paper (Mînzu and Cernega, 1999).

An algorithm for the computation of the supremal controllable language of the admissible language was proposed in this paper. This algorithm can be used as a tool to verify the condition of controllability of the admissible language.

REFERENCES

Charbonnier F.(1996). Commande supervisée des systèmes à événements discrets. *Phd Thesis*, INPG, Grenoble, France.

Ramadge P. J., Wonham W. M. (1997). Supervisory Control of A Class of Discrete Event Processes. In *SIAM J. Control & Optimization*. **Vol.25, No.1**, pp.206-230.

Wonham W. M., Ramadge P. J. (1987). On the Supremal Controllable Language of a Given Language. In SIAM J. Control & Optimization. **Vol.25, No.3**, pp.637-659.

Ramadge P. J., Wonham W. M. (1989), The Control of Discrete Event Systems.In *Proceedings of the IEEE*. **Vol 77, no 1**.

Suzuki T., Kanehara T., Inaba A., Okuma S.(1993). On Algebraic and Graph Structural Properties of Assembly Petri Net. In *IEEE Proceedings International Conference on Robotics and Automatics*, pp 507 – 514.

Mînzu V., Henrioud J.M. (1993). Systematic Method for the Design of Flexible Assembly Systems. In *Proceedings of IEEE International Conference on Robotics and Automation*. Atlanta-USA.

V. Mînzu, J.M. Henrioud (1997). Approche systématique de structuration en postes des systèmes d'assemblage monoproduits; In *Journal Européen des Systèmes Automatisés*. **Vol.31**, **No.1/1997**, p57-78; HERMES ISSN 0296-1598.

Mînzu V., Cernega D.C. (1999). Supervisory Control Technique for Assembly Workstation. In *Proceedings of IEEE International Symposium on Assembly and Task Planning ISATP'99*, pp88 - 93 Porto, Portugal.