A methodology to design and check a plant model

Benoit Rohée, Bernard Riera, Véronique Carré-Ménétrier, Jean-Marc Roussel

A METHODOLOGY TO DESIGN AND CHECK A PLANT MODEL

B. Rohee, B. Riera, V. Carré-Ménétrier, J-M. Roussel

B. Rohee, B. Riera, V. Carré-Ménétrier
CReSTIC, Moulin de la Housse, BP 1039, 51687 Reims
Cedex 2 France,
Ph: +33 (0) 3.26.91.32.26, Fax: +33 (0) 3.26.91.31.06
benoit.rohee, bernard.riera ,veronique.carre
@univ-reims.fr

J-M. Roussel
LURPA, 61, avenue du Président Wilson, 94235 Cachan
Cedex France
Ph: +33 (0) 1 47 40 22 15, Fax: +33 (0) 1 47 40 22 20
jean-marc.roussel@lurpa.ens-cachan.fr

Abstract: Applications on manufacturing systems (diagnosis, control, supervision…) often require a plant model. Obtaining the plant model is a difficult task because of systems complexity and deepth knowledge of physical materials. A necessary condition that must be verified by the plant model is its capability to respond to all the requests of the specification model. This paper ends by two complementary approaches a mathematical property is proposed to check formally and this capability and a methodology to design the plant model is proposed in order to guarantee the model capability according to the most permissive control model.

Keywords: Discrete event systems, Finite automata, Formal specification, Formal verification.

1. INTRODUCTION

In the field of the Discrete Event Systems (DES), many works (Carré-Ménétrier and Zaytoon, 2002), (Cimatti, *et al.*, 2003), (Gouyon, *et al.*, 2004), (Ndjab Hagbebell, 1999), (Roussel and Giua, 2005), (Tajer, 2005), (Trikki, 2005), (Zaytoon and Carré-Ménétrier, 2001), require a behavioral modeling of the plant. A plant model describes the asynchronous and nondeterministic evolutions of a physical system associating discrete evolutions and linear or not physical laws. A model is a representation of a system that depends on the observer and its point of view. A model can be formal, semi-formal, discrete, continuous, static, dynamic, deterministic, stochastic… For example, the model of a pneumatic cylinder can be a 2-states model (left side and right side), or a N-states model (the stem position is discretized in N distinct positions), or a continuous model (stem position represented by a real variable). In the case of stochastic approaches, probabilities characterize the events occurrence such as the breakdowns (Proth, 1997). The framework of this paper, only concerns formal, discrete, represented by means of automata in finite states models. Time is not taken into account. Moreover, we focus on the manufacturing systems characterized by a process part (plant) and a control part in interaction together. Consequently, in our work, modeling a plant is similar to describe all the coherent and possible occurrences of sensors events according to the commands.

In the case of DES, in a very general way, independently of the application (command synthesis, control command checking, supervisory control, diagnosis, supervision…), some theoretical approaches are based on the interactions between the plant model and a specification model (Ramadge and Woham, 1987), (Balemi, 1992), (Akesson, *et al.*, 2003), (Meftah, *et al.*, 2006). Consequently, independently of the application, it is important to make sure the plant model is able to respond to all requests relating to it coming from the specification model. Some works propose examples of plant model. Hasdemir, *et al.*, (2003) shows a model for a pneumatic cylinder. This model is empiric and there is no possibility to check its quality with regard to another various model. Within the framework of this paper, two complementary approaches are proposed.

- a mathematical property is proposed to formally and simply check an unspecified plant model is able to respond to all the requests from a specification model addressed to it.
- a methodology of designing plant models is proposed in order to guarantee it is complete with the most permissive control model. Within this framework, modelling the system consists of describing all the occurrences of possible sensors events according to the orders coming from the PLC. The obtained plant model is particularly adapted to control synthesis applications.

The first part of the paper details the framework in which the modelling step is placed and the assumptions that are carried out. The second part deals with the finite state automata, which are the selected modelling tool. The third part of the paper is about the checking problem of the plant model that consists in confronting it with a specification model. The fourth part of the paper presents the methodology to design the plant model. The suggested approach wants to be modular and close to the structure of plant (sensors, pre-actuators, and actuators). The last part illustrates the methodology and the checking of plant models by means of a simple but realistic application example.

2. CONTEXT AND ASSUMPTIONS

A plant model of a manufacturing system represented by a finite state automaton reveals events that can be observable (or measured) or unobservable. In addition, the plant model must necessarily respect various dynamic and static constraints:

- Static constraints are defined by the plant structure (Thistle, 1996). For example, two sensors measuring the cylinder position (left side or right side positions) cannot have their active states at the same time.
- Dynamic constraints characterize the possible evolutions sequences. For example, for a cylinder modeled by a 3-states automaton (left side, intermediate position, right side), way from the left side state to the right side state must be thought the intermediate position state.

A manufacturing system is composed of several actuators in physical interaction. In other words, some movements generate dangerous behaviors. It is the case for example of two cylinders sharing a common workspace. The designer of the plant can choose to integrate or not these characteristics in the plant model. If the plant model tolerates all the events including those that can lead to dangerous situations, it is called a "free" plant model. In the contrary case, it is called a constrained plant model. For example, in (Kumar, *et al,* 2001), (Chandra and Kumar, 2001), from the constraints enumeration and a "global and free" plant model, the constrained plant model is generated automatically. The main problem of this approach is to be exhaustive in the description of the constraints.

As seen in the introduction, work in DES considers the specification and process models as input data for the formal methods. It should be noted that these methods are less or not interested in getting the plant model. However, it is a difficult task because:

- It requires the knowledge of the technologies used in the process and the modelling tools skill.
- In manufacturing systems, plant is a distributed system whose many components are in interaction together.
- If the plant model is used in industrial applications, it is necessary to go down on a detail level taking into account all the possible events of sensors and commands.

Within the framework of our work, we supposed that:

- Studied systems are asynchronous (non-simultaneity of events).
- Each material element of plant is modeled by a discrete model with a finite state number of states.
- Time only makes possible to know the chronology between the events.
- The product and its following are not taken into account

Plant is modeled and represented by a finite state automaton. We chose the Balemi's interpretation (1992) (Fig. 1) where controllable events represent process inputs (commands) and uncontrollable events represent process outputs (sensors responses).

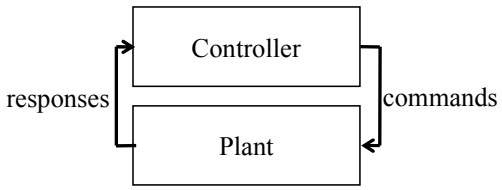

Fig. 1. Interpretation according to Balemi (1992).

The control part and the plant exchange electrical signals. We assume that they are Boolean (Fig. 2). They can take the logical values low level and high level. The behavior of each signal can be described by an automaton with 2 states and 2 transitions. The transitions correspond to the state changes of these boolean signals. As illustrated on Figure2, the states are represented by circles and the transitions by arrows. The initial state is selected arbitrary and is represented by an entering arrow. Two events are associated to each logical signal S: $\uparrow$S and $\downarrow$S. $\uparrow$S corresponds to the trigger of the signal S from low level to high level of the signal S. $\downarrow$S corresponds to the trigger of the signal S from the high level to the low level. These are the only possible events on the logical signals and there is always alternation of these 2 events.

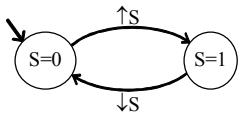

Fig. 2. Behavior of the signals.

More generally, we consider that a controllable event corresponds either to the rising edge $\uparrow$C or to the falling edge $\downarrow$C of a command. An uncontrollable event is associated either to the rising edge $\uparrow$U or the falling edge $\downarrow$U of a sensor response. The sets Σ_C and Σ_U are defined as following: $\Sigma_C = \uparrow C \cup \downarrow C$ and $\Sigma_U = \uparrow U \cup \downarrow U$. Events sets $\uparrow$C and $\downarrow$C, correspond to set and reset of commands. Events sets $\uparrow$E and $\downarrow$E correspond to set and reset of the responses.

The following paragraph deals with the used modelling tool.

3. MODELLING TOOL

A finite state automaton is expressed by the following 4-tuple:

$A = \left\langle Q^A, \Sigma^A, \delta^A, {q_0}^A \right\rangle$ in which, ${Q}^A$ represents the finite set of states, ${\Sigma}^A$ represents the finite set of events, ${\delta}^A$ represents the transition function $\delta^A : Q^A \times \Sigma^A \to Q^A$ and ${q_0}^A$ represents the initial state.

Operations used on automata are full and broadcast synchronizations of two automata and restriction of language. Full synchronization, broadcast synchronization and restricted language are respectively noted A||B, A+B and $A\big|_{\Sigma^B}$. The Supremica software (Akesson, *et al.*, 2003) makes possible to calculate and defines formally these operations.

4. CHECKING OF A PLANT MODEL WITH RESPECT TO A SPECIFICATION MODEL

The confrontation of a plant model with a specification model consists in checking that the plant model is always able to accept events sent to it. If the plant model, noted P, and the specification model noted S are represented by two finite state automata, that means that the language of S reduced to the P-alphabet is included in the language of P reduced to the S-alphabet. The following equivalence makes it possible to check this condition in a simple and formal way:

(S+P)||S ≡ S||P (equation 1)

The demonstration, which is not the paper subject, is done by checking equivalence for all the possible events occurrences. The property can be simply explained. The automaton obtained by carrying out the composition S||P includes events occurring in S and not belonging to the P-alphabet, events occurring in P and not belonging to the S-alphabet and events occurring simultaneously in P and S.

The obtained automaton by carrying out composition (S+P)||S includes events occurring in S, and events occurring in P and not belonging to the S-alphabet.
No event common to the S-alphabet and the P-alphabet can occur in S and not in P if and only if (equation 1) is verified. If it is the case, this means that all the events of the specification S addressed to P are well recognized by P. The property is equivalent to $S\big|_{\Sigma^P} \subseteq P\big|_{\Sigma^S}$. This equivalence can be applied without assumptions on plant and specification models. The advantage of the formula is that only the common events to the two models are considered to check if P is complete compared to S.
As example, let us consider the following plant model (P) and specification model (S) of the Fig. 3:

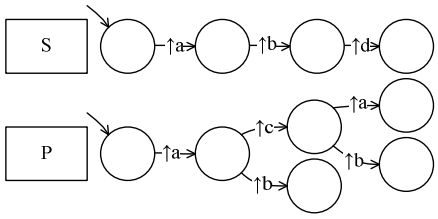

Fig. 3. Illustration of the property on an example.

The language of S is {ø, ↑a, ↑a↑b, ↑a↑b↑d}
The language of $S\big|_{\Sigma^P}$, is {ø, ↑a, ↑a↑b}
The language of P is {ø, ↑a, ↑a↑b, ↑a↑c, ↑a↑c↑a, ↑a↑c↑b}
The language of $P\big|_{\Sigma^S}$ is {ø, ↑a, ↑a↑b, ↑a↑a}
We can notice that $S\big|_{\Sigma^P} \subseteq P\big|_{\Sigma^S}$. Consequently, the property must be true. Composition S+P gives the following automaton on Fig. 4.

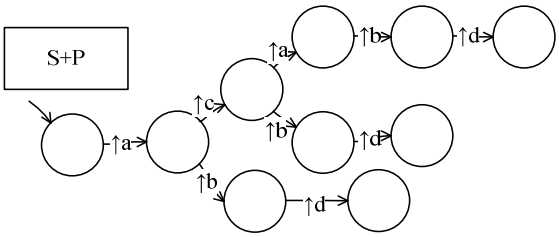

Fig. 4. Resulting automaton by broadcast composition between S and P.

The property is true in this example because all the sequences of S-events reduced to the P-alphabet are included in the language of P reduced to the S-alphabet (Fig. 5):

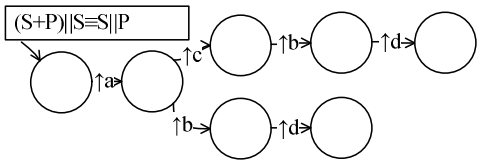

Fig. 5. Property response applied on the example.

Note that the property can be used to confront a specification model with a constrained plant model. The following paragraph proposes a methodology to design a non-constrained plant model. That is to say, the property will be verified on the plant model and the most permissive control model.

5. PROPOSAL OF A MODULAR METHODOLOGY OF MODELLING

Firstly, the manufacturing system has to be cut in modular parts (Fig. 6). Each module includes an actuator and its pre-actuators, and the sensors measuring the actuator situations. Then, each module is modeled. Finally, all the module models are composed to get the “global and free” plant model. This plant model comprises all the possible situations of commands and responses (from sensors). The constrained plant model only includes “correct” situations for the designer.

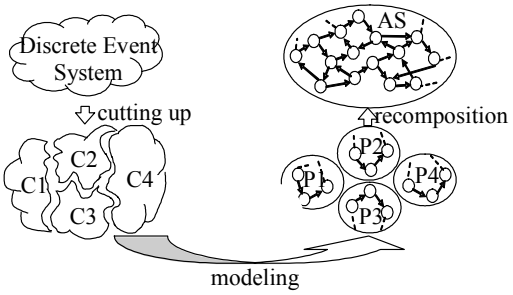

Fig. 6. Modular step of modelling.

4 steps are necessary to model a module:
1) The discretization of material elements requires the determining states the actuator, the pre-actuators and the sensors can be in.
2) A finite state automaton, named Apec, describes the pre-actuator behavior according to controllable events (Figure7). We can note the pre-actuators states are non-observable. The Apec model allows to rebuild the pre-actuator state taking into account the controllable events sequence. The state of pre-actuators is not always a combinatory function of the inputs. Indeed, in a more general case, it is a sequential problem. Consequently, this function is represented by an automaton called Apec. Each Apec model state contains its corresponding pre-actuators state. The initial state is supposed to be known.
3) Sequences describe the sensors behavior according to the current state of the pre-actuators (Fig. 7). It is thus necessary to define, as many sequences as pre-actuators number of states. These sequences are called Acp(pre-actuator_state). They describe the sensors states changes. These are non-controllable events sequences for each pre-actuator state. For that, at first, it is necessary to know the pre-actuators effects on the actuator and then, the actuator effects on the sensors. In other words, the pre-actuators state induces an evolution sequence of sensors state, via the actuator. This method includes the classical case where each vector of sensors state characterizes one and only one actuator state.

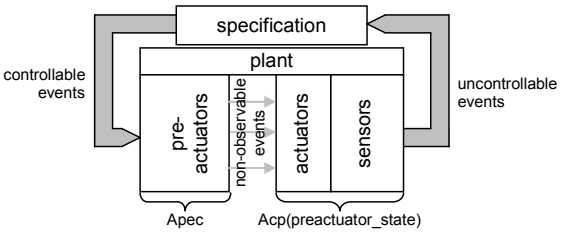

Fig. 7. Modelling of a module.

4) The finite state automaton of the module i, named Api, is achieved by using the finite state automaton Apec and the sequences named Acp(pre-actuator_state). It aims to represent the coherent uncontrollable events evolution (sensors) induced by the controllable events evolution.

The plant model noted Apo is obtained by reconstructing the module models with the broadcast product between all module models. This model is thus the "free and global" plant model.

By this construction method, the obtained plant model is able to respond to all the controllable events and, so, respects the most permissive control model. The suggested methodology is particularly adapted to controller synthesis applications (Tajer, 2005). As specified at the beginning of the paper, manufacturing system applications are various (diagnosis, synthesis, product following…). Generally, applications are based on a confrontation between a plant model and a specification model. The next paragraph illustrates the property through a model proposed by Hasdemir, *et al.*, (2003) and the methodology to design the plant model.

6. EXAMPLES

To illustrate the synthesis and checking steps applied to plant models, we took an interest in the control problem of "Tapiris" system subset (*Teaching model Tapiris Schneider Electric: www.schneider-electric.fr/tapiris-M166-R297-A262.htm*) (Fig. 8).

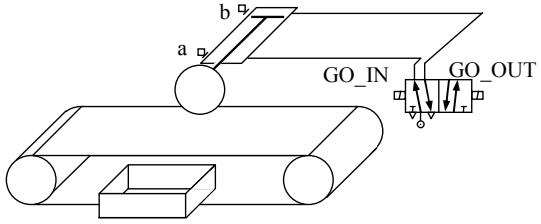

Fig. 8. Tapiris subsystem.

The subset makes possible to move parts on a conveyor to evacuate them in a box. In this example, we only consider the double-acting cylinder controlled by a bistable electro-valve with two positions and two sensors (a and b) detecting the extreme positions of the cylinder.

a) Illustration of the property

To highlight the property interest, we propose to apply it to a plant model (P) of the Tapiris system based on the empirical model proposed by Hasdemir, *et al.* (2003) (Fig. 9).

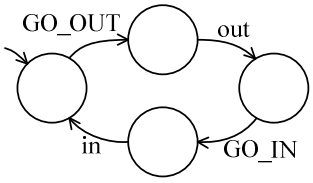

Fig. 9. Empirical plant model.

This model is confronted with the most permissive control specification model (S) (Fig. 10). The 2 models are represented by 2 finite states automata that have the same alphabet in this example.

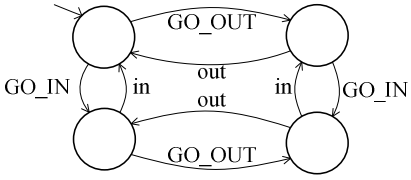

Fig. 10. Proposal for a non-satisfied specification by the plant model.

The property is used to check that the plant model is able to respond to the commands. The result indicates that it is not the case. The plant model does not authorize to change the movement direction before arriving in the final position. This plant model does not respect the most permissive command model. It cannot be used for some applications like in (Roussel and Giua, 2005).

b) Modeling methodology

We propose now to apply the proposed plant modelling methodology to this subset.
As written before, the studied system is composed of one module. The 4 stages of modeling are now detailed.
1) The valve is discretized by a 2 unobservable states model corresponding to the two stable states of the valve (state1 and state2) and receiving the 2 commands (GO_IN and GO_OUT) sent by the controller. Taking into account the cylinder instrumentation, it is discretized in a 3 states system (left side, intermediate position, right side). Lastly, each sensors (a, b) is modeled by 2 states representing the 2 logical boolean states they can get in. We supposed the initial situation is known: GO_IN=0, GO_OUT=0, state=state1, a=0 and b=1.
2) The valve behavior according to the commands can be represented by a truth table. However, determining the valve state is not a combinatory problem. Indeed, in the case of the 2-positions, when the two commands are not activated, the valve drawer preserves its position. The truth table of the studied valve, presented Table 1, shows the function of the valve state from commands.

Table 1 Valve state according to the commands

GO_OUT	GO_IN	valve state (n)
0	0	valve state (n-1)
0	1	state 1
1	0	state 2
1	1	valve state (n-1)

3) The truth table is easy to translate in a finite state automaton noted Apec (Fig. 11). Each state corresponds to a vector made up of the commands and valve states. With the assumption of the initial situation (command and valve state), the initial state of Apec is identified.

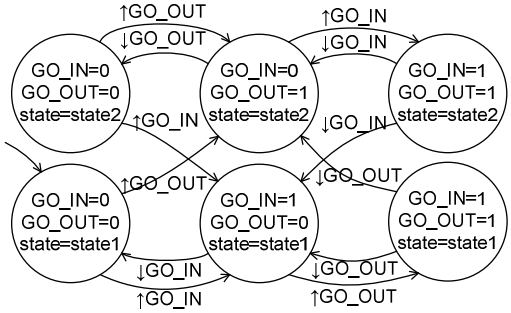

Fig. 11. Automaton Apec of the studied subsystem.

4) Then, for each state of the valve, the possible evolutions of the actuator are determined. For the Tapiris subsystem application example, the following sequences are obtained (Fig. 12).

state 1
right side → intermediate position → left side

state 2
left side → intermediate position → right side

Fig. 12. Cylinder evolution diagram depending on the valve state Automaton Apec of the studied subsystem.

For each state of the actuator, the corresponding sensors state is associated to it (Fig. 13).

left side:	a=0, b=1
right side:	a=1, b=0
intermediate position:	a=0, b=0

Fig. 13. Association between the actuator states and the sensors state.

Sensors evolutions sequences, named Acp(pre-actuator_state), are easily obtained by using the sequences of evolution and association between the pre-actuator states and the configurations measured by the sensors (Fig. 14).

A_{CP}(state1)

a=1 → a=0 → a=0
b=0 → b=0 → b=1

A_{CP}(state2)

a=0 → a=0 → a=1
b=1 → b=0 → b=0

Fig. 14. Sensors evolution sequences according to the valve.

5) Knowing the initial state, the valve state (according to the commands) and the evolution sequences of sensors (according to the valve state), it is possible to generate the plant model. The global and free model of the Tapiris example is presented Fig. 15.

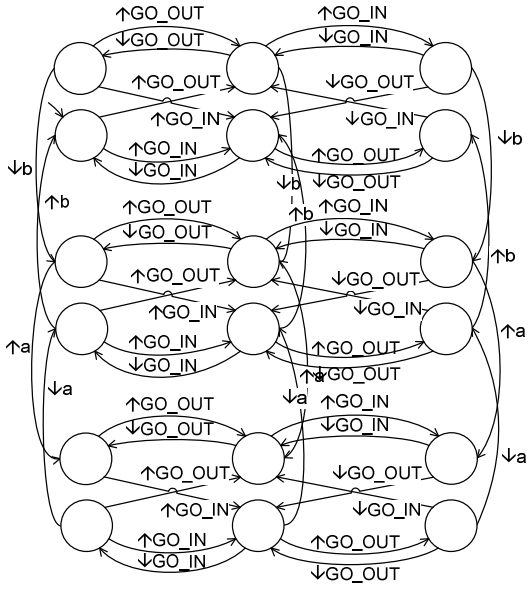

Fig. 15. Plant finite state automaton APO.

This plant module model is complete with respect to the most permissive controller. This means that the

plant model accepts every command sent by the controller.
In this example, the system is composed of only one module. Consequently, in this case, the module model is the plant model. In a real application, when all the modules are performed, they should be recomposed to obtain the model of "global and free" plant. According to applications, the global model can be restricted taking into account only normal situations.

7. CONCLUSION AND FUTURE WORK

Applications in the manufacturing systems field (diagnosis, control design, supervision…) are often based on methods using a plant model. A necessary condition to guarantee the plant model validity is to be sure that it is able to respond to all the specification model requests. Within the framework of this paper, two complementary approaches were proposed:
- A property was proposed to check formally and simply that an unspecified plant model verifies this condition.
- A methodology was proposed to design the plant model. The obtained plant model can accept all commands sent by the controller. The proposed methodology is based on a description of all possible occurrences of the sensors. The proposed modelling approach is modular and based on the manufacturing system physical structure (sensors, pre-actuators, actuators).

Concerning the suggested property, it seems to be interesting to apply it to large-scale systems in the control and supervision fields.
In addition, future work concerning the methodology to design the plant model can deal with:
- modules library development having various technologies (electric, pneumatic, hydraulic energies) usually used in manufacturing systems.
- formalization of safety constraints. The objective is to obtain a methodology to design the constrained plant model, particularly adapted for control synthesis applications (Carré-Ménétrier and Zaytoon, 2002), (Zaytoon and Carré-Ménétrier, 2001).
- the taking into account and the follow-up of the product in the plant model for supervision applications.

REFERENCES

K. Akesson, M. Fabian, H. Flordal A. Vahidi (2003). Supremica – a tool for verification and synthesis of discrete event supervisors *Procedings of the 11th Mediterranean Conference on Control and Automation*

S. Balemi.(1992). Input/output discrete event processes and system modeling. In Balemi et al. [8], pages 15--27. *Procedings. of the Joint Workshop on Discrete Event Systems (WODES'92), Prague, Czechoslovakia*

V. Carré-Ménétrier, J. Zaytoon (2002). Grafcet, behavioural issues and control synthesis. *European Journal Of Control synthesis Article* **Vol. 8/4** p. 375-401

V. Chandra, R. Kumar (2001). A new modelling formalism and automata model generator for a class of discrete of discrete event systems. *Mathematical and Computer modelling of Dynamical systems*

A. Cimatti, C. Pecheur, R. Cavada (2003). Formal verification of diagnosability via symbolic model checking. *18th International Joint Conference on Artificial Intelligence)*

D. Gouyon, J-F Petin, A. Gouin (2004). A pragmatic approach for modular control synthesis and implementation. *International Journal of Production Research,* **Vol. 42**, n° 14,

I. Tolga Hasdemir, S. Kurtulan, L. Gören (2003). Supervisory control of a pneumatic system using PLC. *International Conference on Electrical and electronics Engineering*

R. Kumar, V. Chandra, S. Mohanty (2000) Automated control synthesis for an assembly line using discrete evenet system control theory. *IEEE Transactions on Robotics and Automation*

T. Meftah, H. Gueguen, N. Bouteille, V. Boutin (2006). Spécification par contraintes de la commande des systèmes automatisés, *JDMACS-JNMACS 06*

C. Ndjab Hagbebell (1999). Synthèse de la commande des systèmes à événements discrets par grafcet *Diplôme de doctorat* Université de Reims

J-M Proth (1997). Petri nets for modelling and evaluating deterministic and stochastic manufacturing systems. *6th International Workshop on Petri Nets and Performance Models*

P. J. Ramadge, W. M. Wonham (1987). Supervisory control of a class of discrete event processes *SIAM Journal of control and optimisation*,

A. Tajer (2005). Contribution aux approches formelles de synthèse de commande spécifié par grafcet. *Diplôme de doctorat* Université de Reims.

J.G. Thistle (1996). Supervisory control of discrete event systems, *Mathematic Computing Modelling,* **Vol. 23**, p. 25-53, 1996

S. Trikki (2005). Contribution à la conception d'outils de supervision « active » pour les procédés continus et manufacturiers. *Diplôme de doctorat.* Université de Valenciennes et du Hainaut-Cambresis.

JM Roussel, A. Giua (2005). Design dependable logic controllers using the supervisory control theory, *16th IFAC World Congress* Prague Czech Republic

J. Zaytoon, V. Carré-Ménétrier (2001) Synthesis of a correct control implementation. *International Journal of Production Research,* **Vol 39/2**,