Automatic Design of Control Laws Based on Petri Net Formalism for Complex Discrete Event Systems

Deschamps Eric, Henry Sébastien and Zamaï Eric

Laboratoire d'Automatique de Grenoble – UMR CNRS 5528

Institut National Polytechnique de Grenoble

ENSIEG - Rue de la houille blanche, Domaine Universitaire

38402 Saint Martin d'Hères

FRANCE

Abstract: In an industrial context, the design of discrete control laws to drive a manufacturing system is usually assumed off line by several experts. Consequently, not only does the design of all the control laws mobilize many PLC program developers, but also, in the event of unexpected operating part failures, the reconfiguration process can be considered only from a manual point of view. This is mainly due to the lack of a generic method to model the controlled system abilities from which control laws can be automatically designed. The paper first presents a methodological approach to obtain all the required information on the controlled system for the purpose of modelling it. Second, an automatic generation of the Petri net model is provided. Finally, based on such a model, a control laws design algorithm used to automate the design of control laws is presented.

Keywords: Control laws design, Reconfiguration, Petri nets, Manufacturing automation, Discrete Event Systems.

M. Eric Deschamps was born in France in 1980. Since 2004, he is PhD student at the Grenoble Institute of Technology (INP-G), France. His research interests include supervision, monitoring, and control of discrete event systems. He received the master's degree in Automatic from the INPG and the engineer's degree from the Ecole Nationale Supérieure d'Ingénieurs Electriciens de Grenoble (ENSIEG; electrical engineering school), INPG in 2004.

Sébastien Henry was born in France in 1975. He received a Ph.D in automatic control engineering and computer-integrated manufacturing from the Grenoble Institute of Technology, France, in 2005. He is currently an associated professor at the Université Lyon 1 and does his research at the Laboratoire d'Informatique pour l'Entreprise et les Systèmes de Production (LIESP). His research interests include supervision, monitoring, and control of discrete events systems.

Eric Zamaï was born in France in 1971. He received a Ph.D in electrical engineering from the University of Toulouse, France, in 1997 and the ''Habilitation a` diriger des recherches'' diploma at the National Polytechnic Institute of Grenoble (INP-G), France, in 2006. He is currently an associated professor at the National Polytechnic Institute of Grenoble France (INPG) and does his research at the Laboratoire des Sciences pour la Conception, l'Optimisation et la Production (G-SCOP). His research interests include supervision, monitoring, and control of discrete events systems.

1. Introduction

This paper deals with the control of Automated Manufacturing Systems (AMS). The main function of the AMS is to transform a product flow to achieve certain objectives: provide products according to customers' specifications (surface roughness, colour, geometry, etc.) while minimising production cost and respecting security constraints. An AMS is composed of two parts, a controlled system and a control system (see Fig. 1). The controlled system is divided into an operative part including a set of resources (cylinders, conveyor belts, CNC, robots, etc.) acting on the product to give the added value. It is also composed of a sensors/pre-actuators interface to ensure the communication between the operative part and the control system. The control system integrates the software, the hardware and the information in charge of the process management (control laws using the IEC61131-3 standard [1] and controlled system model). To manage the complexity of a manufacturing system, the architecture of the control system can be split into five levels called the CIM architecture [2] as presented Fig. 2. Each module of this architecture integrates monitoring, supervision and control functions [3]. Each level has different knowledge granularity and abilities to carry out its particular function. The lowest level is decomposed into several modules that carry out the control of the resources.

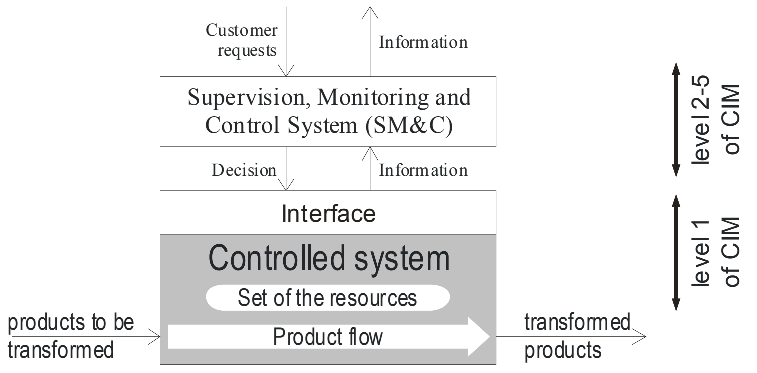


Figure 1. Diagram of an AMS

In an industrial context, the AMS is subject to failures in its operating part but also to disruptions in the customer requests (quantities, product specifications). To face shorter time to market, industry's only choice is to decrease the control laws design time and increase the reactivity of the AMS in order to guarantee the delivery lead time. However, today, control laws using the IEC61131-3 standard are designed by a team of automation engineers [4]. Consequently, the control laws design time is very long and indeterminate; the optimality of the solution is not ensured; a verification/validation step is required; and, if the automation engineer leaves the company, the knowledge about the controlled system abilities is lost. Thus, to improve the productivity and the reactivity of AMS, the design of control laws must be partially automated. So, it is first essential to provide a method to help engineers to capitalise on the knowledge of the controlled system's abilities in an adequate data structure (see Fig.3). Second, to use these data, a technique to generate a formal model must be provided. This modelling tool must allow not only the validation of properties, but also the design of optimal time control laws.

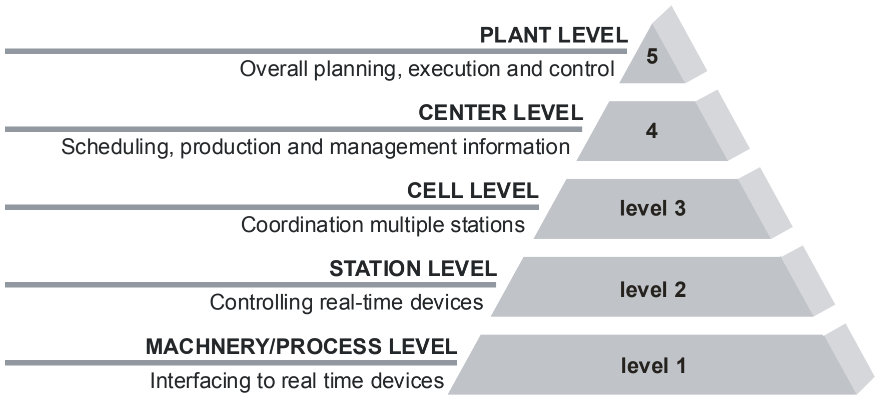

PLANT LEVEL
Overall planning, execution and control
5
CENTER LEVEL
Scheduling, production and management information
4
CELL LEVEL
Coordination multiple stations
level 3
STATION LEVEL
Controlling real-time devices
level 2
MACHNERY/PROCESS LEVEL
Interfacing to real time devices
level 1

Figure 2. The CIM Architecture

This paper is organized as follows. Section 2 gives a short review of the main works in the control laws design field of research. In section 3, the approach is situated in the context of the CIM architecture, and the requirements concerning the knowledge to be modelled are presented. Section 4 proposes a method to help engineers to capitalise on the formal knowledge of the controlled system model. From such a modelling tool, a translation into Petri nets is presented. Finally, an algorithm able to design optimised control laws according to the process shown in Fig. 3 is given. Section 5 gives an example to illustrate the proposed approach. The conclusion and future works end this paper.

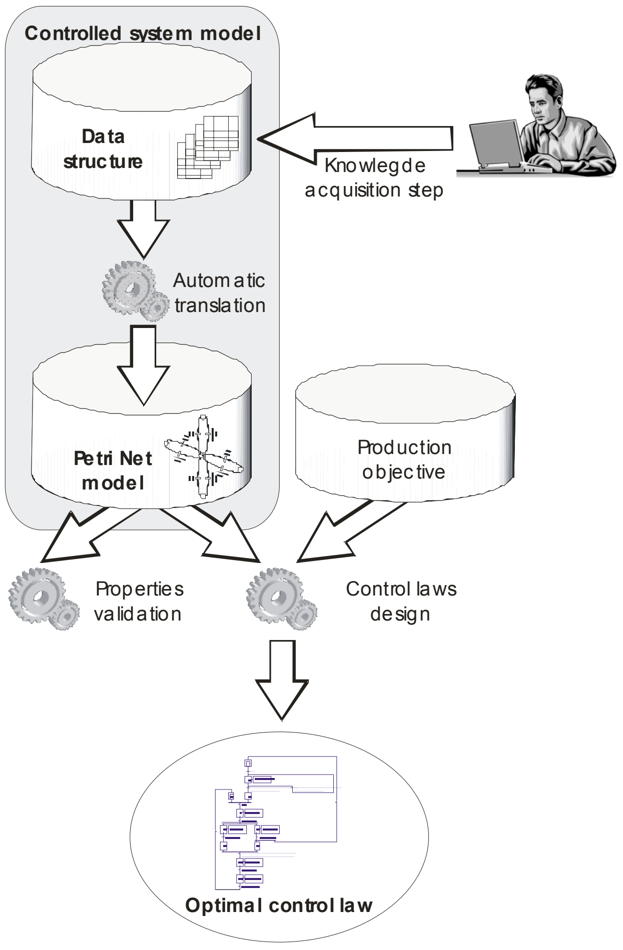


Figure 3. Principle of Control Laws Design

2. State of the Art

This paper focusses on the capitalisation of controlled system abilities and on their exploitation for automatic control laws design (see Fig. 3). To propose a solution to this problem, the main automatic control laws design approaches are presented. However, this paper is not a review, and for reason of concision, only analytical conclusions on these approaches are given. They can be classified into two kinds of approaches according to the CIM level (see Fig. 2) to which they can be applied.

The first two approaches [5] and [6] are used in the scheduling context [7] (see Fig. 2). These approaches allow the generation of operations sequences, and are based on models which represent accessibility and preceding relationships between operations which, but only those have effects on the product. In an uncertain context, these approaches are based on the capitalisation of knowledge about the possible transformations of the product which are essential at this level to guarantee the reactivity of the AMS. Indeed, in the event of failure synonymous with the loss of the abilities of the controlled system (machine breakdown), the control system can generate on line a new operation sequence (real time scheduling) which is dispatched to modules of the lower levels.

The approaches presented in [8], [9], [10] and [11] are used at levels 2 and 3 of the CIM architecture (see Fig. 2). The models are based on the all of the controlled resources with their own control laws. Then, to coordinate all these resources, the proposed automatic design method allows the generation of a control code which limits the system behaviour to the most permissive [12]. It should be noted that the control code is designed from the control specifications (depending on the physical constraints) but not from product specifications. At the lowest level, directly capitalising there adapted control laws is effective and sufficient. But to coordinate all these functional chains, the approaches presented do not capitalise the associated knowledge but directly the control code (representing all the possible control laws). It results in a limitation in terms of the complexity of the controlled systems that these approaches can manage.

Then, the literature proposes two kinds of approaches. The first kind, corresponding to the scheduling level, has a high level point of view which is only oriented to product transformation. The second kind corresponding to levels below the scheduling level has a point of view oriented only towards resources and constraints existing between them. However, the coordination level requires having both these points of view: one to coordinate the resources, one to act on the product flow depending on the objectives given by the scheduling level (see Fig. 4). In the next paragraph, distinctive features of the coordination level are clarified.

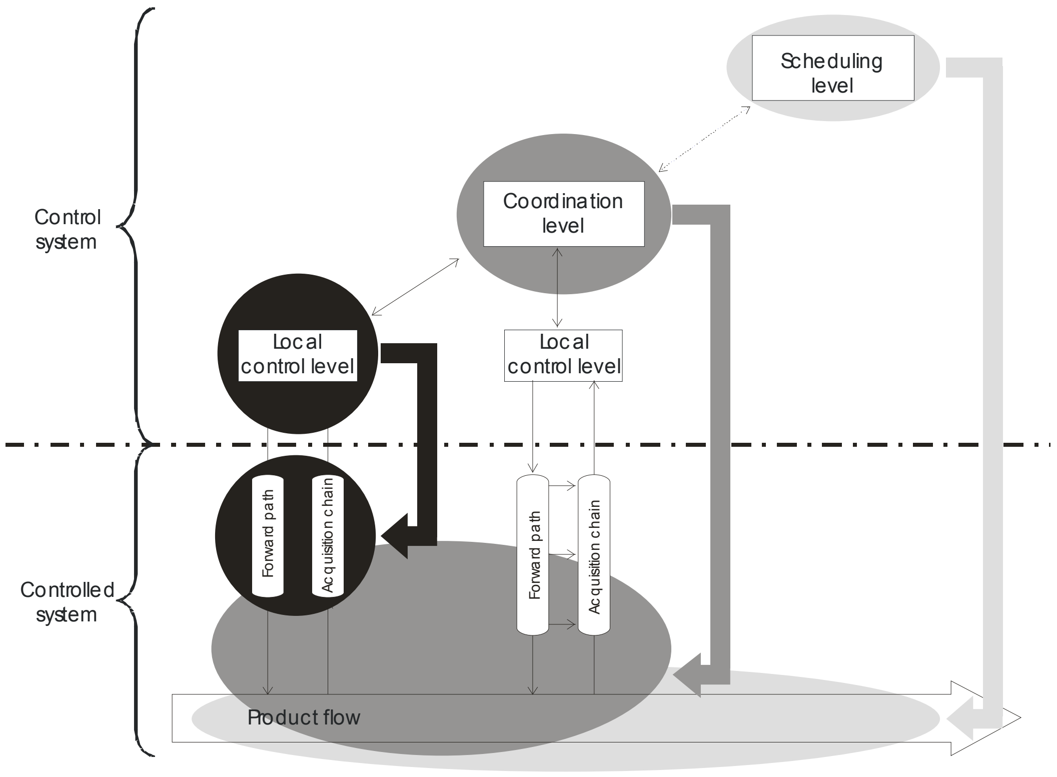


Figure 4. Main Goals of CIM Levels

3. Modelling Requirements

Considering classical CIM architecture (see Fig. 2), our approach is situated at the coordination level of the structure in the monitoring, supervision and control context [13]. The specificity of this level is to link orders sent by the scheduling level and the local control level split into several functional chain (FC) modules (see Fig. 5). In this case, this level of control has to model several types of knowledge. It is detailed in the next paragraph from two points of view.

First, the coordination level drives functional chains. Thus the relationships between the controlled functional chains as well as their direct physical environment must be modelled. Let us consider the example of two orthogonal cylinders which must not be extended simultaneously to avoid collision between these cylinders. Whether these two cylinders are controlled by the same coordination module or not, extending one cylinder means retracting the second. This constraint on the other controlled functional chain state or on the environment state must be modelled. To do this, the state variables characterising the operating part and the direct environment must also be modelled. Next, to act on these state variables, not only the operations proposed by functional chains (command quantities) must be modelled, but also the relationships with the evolutions of state variables characterising the operating part.

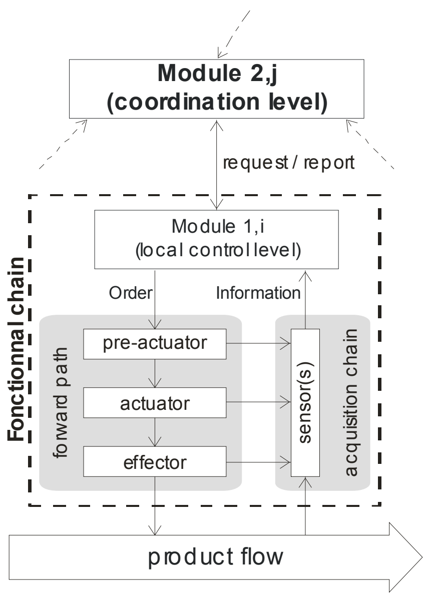


Figure 5. Proposed Architecture

Secondly, the goal of the coordination level is to act on products with the use of services offered by the functional chains that this level controls (see Fig. 4). So not only must the state variables characterising the product be modelled, but also the relationship between the operation and the evolution of these state variables. Indeed, launching an operation leads to an evolution in the operating part which can lead to an evolution in the state of the product. In the same way an operation can be constrained by the state of the product flow.

To conclude, all the knowledge presented above must be modelled at the coordination level. At the coordination level, the model must contain functional chain abilities and the link with the possible effects on the product flow (see Fig. 4).

The section below focusses on the coordination level of the control system. To show the global interest of the proposed approach at this level, it is important in this paper to present not only the modelling step, but also the automatic translation into Petri nets and the control laws design algorithm.

4. The Proposed Approach

Due to the profusion of information to be modelled, the model used at the coordination level cannot be a simple representation of possible effects on the product flow and the corresponding preceding constraints as in a scheduling context. However, in order to manage the size of the model, it cannot be the explicit representation of all the possible states of the controlled system as in the lowest level of the architecture CIM. In fact, even if the complexity can be managed by a computer, the modelling step must be a knowledge capitalisation step for designers. Then, the knowledge must be transformed into formalism appropriate to the requirements (validation, control laws design, etc.) as in Fig. 3.

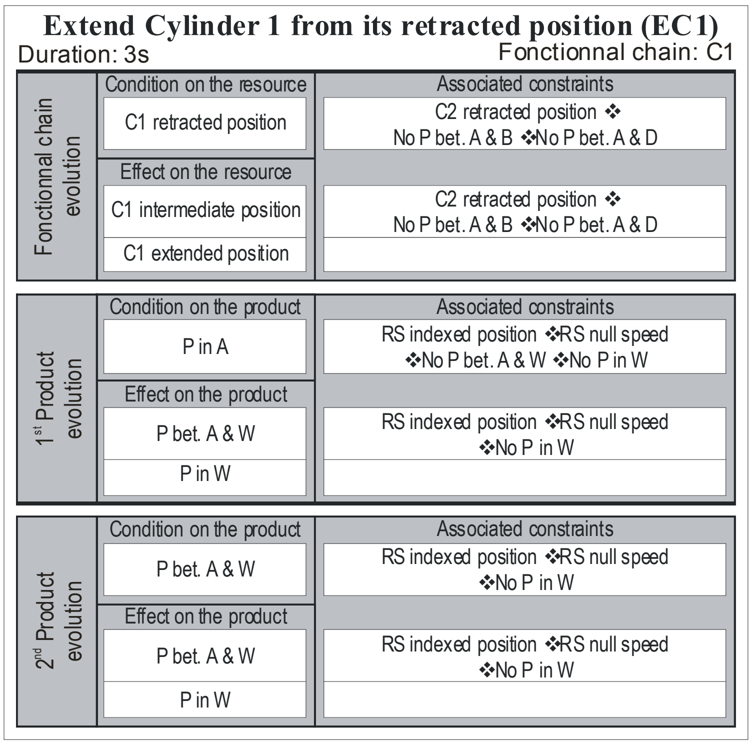
The next section presents the data structure, based on the principle described in section 3.

4.1. Data Structure

To design the control law, the model must contain all the information presented in section 3. Designers must be provided with a method to help them in the capitalisation step. The proposed method consists in determining all the services offered by each functional chain that makes up the controlled system. These services correspond to the possible action of each elementary FC (extend/retract a cylinder, run/stop a motor, etc.). The controlled system model is a collection of operations based on services. An operation specifies: the service provided by an FC, the service effects on the operating part, the potential effects on the product flow, the conditions necessary to obtain these effects, and the constraints to be satisfied when the conditions are true. This operation concept is closely related to the notion of action in the automated planning field [14] and especially the action model proposed in [15] and [16].

For one service of a FC, the methodology first proposes to consider the evolution of this FC without studying the possible product evolutions. This evolution is described by the initial state of the FC, its intermediate state which corresponds to its state following the beginning of the operation (resulting from a request at the coordination level) and its final state which corresponds to its state following the end of the operation (indicated by the reception of the lowest level report). All these states are defined as values of state variable FCs. But to allow the control system to request this operation and to launch this evolution, the constraints must be satisfied, before (pre-constraint) and during (constraint) this operation. These (pre-)constraints are expressed on the state variables of all FCs and of the product flow. This first part defines the FC evolution (see Fig. 6). To launch an operation its FC evolution at least must be launched. To do this, the FC must be in the defined initial state and the (pre-)constraints must be satisfied.

Then, in the second step, the possible evolutions of the product flow are considered. They define the Product Evolution (PE) of the corresponding operation (see Fig. 6). Similar to the FC evolution, to define the product evolution, the initial, intermediate and final states of the product are described. Next, (pre-)constraints must also be described. To launch a PE, the product flow must be in the defined initial state of this PE, and thus the associated constraints must be satisfied. The initial product state is called the condition of the PE. Finally, to optimise the control law, the quantifiable criteria are required, e.g. time cycle and the cost. To compute these criteria, the operation modelling must include the operation features such as the duration of the operation.



Extend Cylinder 1 from its retracted position (EC1)

Duration: 3s — Fonctionnal chain: C1

Fonctionnal chain evolution	Condition on the resource	Associated constraints
	C1 retracted position	C2 retracted position ❖ No P bet. A & B ❖No P bet. A & D
	Effect on the resource	
	C1 intermediate position	C2 retracted position ❖ No P bet. A & B ❖No P bet. A & D
	C1 extended position	
1st Product evolution	Condition on the product	Associated constraints
	P in A	RS indexed position ❖RS null speed ❖No P bet. A & W ❖No P in W
	Effect on the product	
	P bet. A & W	RS indexed position ❖RS null speed ❖No P in W
	P in W	
2nd Product evolution	Condition on the product	Associated constraints
	P bet. A & W	RS indexed position ❖RS null speed ❖No P in W
	Effect on the product	
	P bet. A & W	RS indexed position ❖RS null speed ❖No P in W
	P in W	

Figure 6. Extend Cylinder 1 Operation

4.2. Automatic Translation into Petri Nets

The data structure allows the capitalization of all the required data for the automatic control laws design. It does not use classical tools such as state transition systems or Petri nets so that designers can focus on the required information but not on constraints of the modelling tool. However, before designing control laws, several classical properties have to be checked like reversibility, boundedness, liveness and deadlock-freedom properties. In fact, it is necessary to check whether controlled system properties are preserved in the model, but it is not possible to guarantee that engineers have described all the capabilities of the controlled system. To do that, it will be useful to combine our approach with the simulation approach, such as work presented in [11]. To both validate properties and facilitate the automatic design of control laws, this section proposes a translation into Petri net formalism [17] (see Fig. 3). As explained in [18], Petri nets formalism is an adequate tool to model manufacturing systems. They allow the representation of effects, flow, constraints and states as shown in Fig. 7. More particularly, the physical characteristics of products (surface roughness, colour, geometry, etc.) will be translated as attributes of tokens, and then predicate transition Petri nets will be used [19]. The design of such a model is described below.

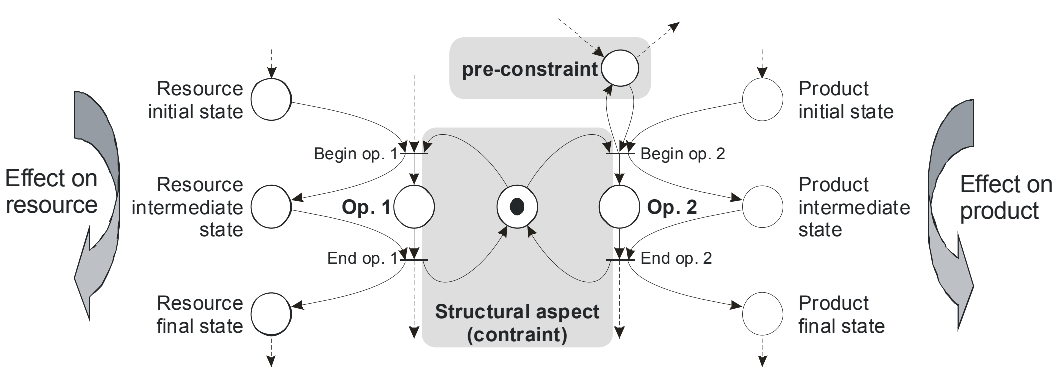


Figure 7. Modelling Capabilities with Petri Net Formalism

To facilitate understanding of the automatic translation into Petri nets, it is decomposed into six steps, each one giving the specification of a part of the complete Petri net model:

- The models of each FC state: the *FC state models* are obtained from the FC evolutions of the data structure. The set of values of state variables which characterize FCs are translated into a set of places. The evolutions of FCs in each operation are translated into two transitions: the evolution resulting from the beginning and from the end of an operation. The first transition links the place *initial state* and the place *intermediate state*. The second links the place *intermediate state* and the place *final state* (see Fig. 10). For a FC, the *state model* is obtained from all of its operations.

- The model of the product flow state: The *product flow state model* is obtained from the PEs of the data structure. The set of positions of product flow are translated into a set of places. The set of state variables corresponding to the physical characteristics of products are translated into a set of attributes of token. Each PE (product evolution) of each operation is translated into two transitions: "begin PE" and "end PE". The first transition links initial position place to the intermediate position place of the corresponding PE. The second links intermediate place position to final position place. If the PE modifies the physical characteristics of the product flow, the functions are respectively associated with the output arcs of the transitions "begin PE" and "end PE" of the corresponding PE, thereby giving the respective values of the intermediate and the final state of the product. The *FC state models* and the *product flow state model* resulting in the representation of the controlled system state, but the possible constraints to launch an operation must also be translated.

There are two kinds of constraints (and pre-constraints). The first kind corresponding to FC evolutions must always be satisfied. The second kind corresponding to PEs must be satisfied according to the evolutions of the product flow associated with the operation launched. The next step translates these different kinds of constraints (and pre-constraints) into two Petri net models.

- The *permanent pre-constraints model*: this model is obtained for each operation from the pre-constraints of the FC evolution of this operation. These pre-constraints must always be satisfied with or without effect on the product flow. To fire the transition "begin operation" (see Fig. 7), FC constraints and product flow constraints must be satisfied. For pre-constraints on the physical characteristics of the product flow (i.e. on attributes of token), predicates on transition "begin operation" are used.

- The *pre-constraints models*: this model is obtained for each operation from pre-constraints of PEs. Each PE is associated with pre-constraints which must be respected to fire the corresponding transition "begin PE".

The two previous steps consider only pre-constraints, which must be satisfied before the launching of the operation. But constraints which must be satisfied during the operation are not translated into Petri nets. The next two steps consider the Petri net translation of constraints which must be respected during the running of the operation.

- The *permanent constraints models*: They translate the constraints of FC evolution. These constraints must always be satisfied during the operation with or without effect on the product flow. For a specific operation, each operation (or PE of operations) that puts the controlled system in a state which does not respects the constraints (of the considered operation) must not be launched. Then, these operations (or PEs of operations) are set up in a mutual exclusion with the considered operation.

- The *constraints models:* They translate constraints associated with the effects on product flow (the constraints of PEs). They are constructed in the same way as the *permanent constraints models.* For a considered PE of an operation, each operation (or PE of operations) that puts the controlled system in a state which does not respects the constraints (of the considered PE) must not be launched.

All these Petri net models represent the controlled system model. They are built separately, but they are linked by places and by transitions. To validate the properties of the Petri net models, only one aggregated model is required. In this way, the same state places and the same transitions are combined between models. However, this is a little bit more complex, because transitions corresponding to "begin operation" and "end operation" (and respectively of PE) are duplicated and merged with transitions "begin operation behaviour" and "end operation behaviour" (respectively of operations behaviours containing the PE). Operation behaviour represents a possible combination of PEs for this operation. From this transformation, one Petri net model is obtained which represents the model of the controlled system capabilities.

However, in order to simplify the structure of the control laws design algorithm, Petri net models are used separately. Thus, the algorithm is split into steps according to different objectives. Because of these objectives, each step exploits a subset of the Petri net models.

4.3. Algorithm to Design Control Laws

From the controlled system model and production objectives (see Fig. 3), the problem of design of optimal time control laws is very complex. To solve this problem, it is necessary to generate the reachability graph of the aggregated Petri net model and to look for the optimal time sequence of transitions from the initial marking to the desired final marking. "But even for a simple Petri net, the reachability graph may be too large to generate in its totality" [20]. To confront this complexity, our first goal is to find admissible control laws. However, to optimize the duration of the designed sequence of transitions at each step of the algorithm, our work naturally turns to the operational research domain. In this section, it is shown that two algorithms can be used to solve problems in the control laws design. These problems are the Travelling Salesman Problem (TSP) with precedence constraints [21] (see section 4.3.3) and the shortest-path problem [22] (see section 4.3.4). Moreover, to design a control law, the algorithm decomposes into four steps the search for the sequence of transitions (see Fig. 8) which correspond to different kinds of operations and constraints. The first step proposes to search for a disorderly sequence of transitions (called *sequence 1*) corresponding to operations which transform the product according to the specifications. The second step defines the possible precedence constraints between these transitions from the *pre-constraints models* and the *permanent constraints models*. From *sequence 1*, the third step finds the sequence (called *sequence 2*) which satisfies all the pre-constraints on the product flow. Transitions added by this step correspond to product transport operations or product transformation operations required to carry out transformations according to product specifications. The fourth step adds transitions corresponding to set up operations in *sequence 2* to satisfy pre-constraints on the operating part. Finally, to optimize the duration of the resulting sequence (called *sequence 3*), the last step looks for parallelisms between transitions.

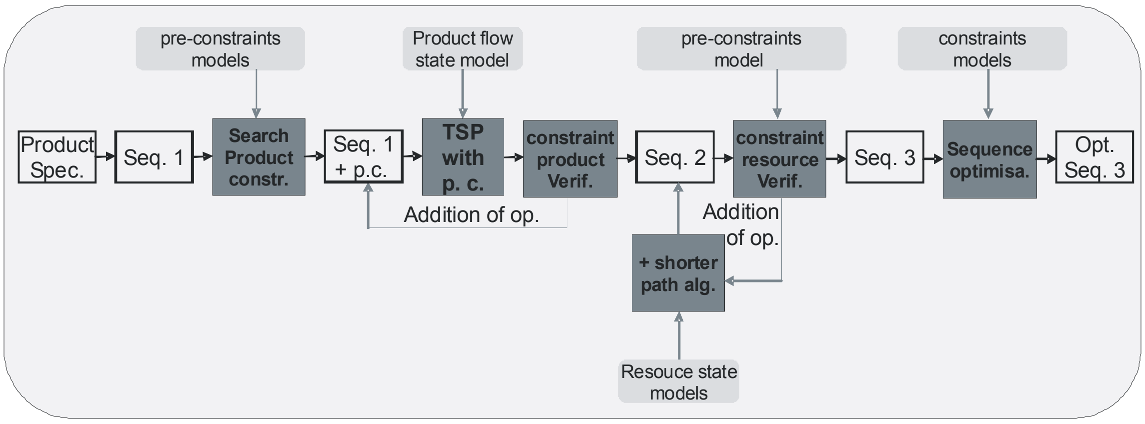


Figure 8. Algorithm of Control Law Design

4.3.1. 1st step: translation of the objective

From the product specifications, which are a set of product characteristics, this first step generates *sequence 1* (see Fig. 8). Product characteristics are given to the product by effects which are produced by PEs of operations. The ends of these PEs are modelled by transitions "begin PE" in the *product flow state model*, and thus must be added to sequence 1. Only for the final product position, is a transition added which has an arc directed from this transition to the place corresponding to the desired position. At the end of this step, *sequence 1* is generated (see Fig. 8).

The next step will specify the possible precedence constraints between operations of this sequence.

4.3.2. 2nd step: precedence constraints of the first sequence

Sequence 1 is disorderly; nevertheless constraints may exist between operations of this sequence (an operation which must be launched before another one). To obtain these precedence constraints, for each transition "end PE" of *sequence 1*, the algorithm verifies the predicate associated with the corresponding transition "begin operation" (an operation which contains this PE) in the *permanent pre-constraint models* and the corresponding transition "begin PE" in the *pre-constraint models*. If there is a pre-constraint on an attribute corresponding to another transition of *sequence 1*, a precedence constraint between these transitions is obtained, and so on. After this step, all the partial precedence constraints between transitions of *sequence 1* are defined.

4.3.3. 3rd step: find all the product evolutions

The next stage of the algorithm consists in the search of a sequence of transitions in *the product flow state model*, which contains all the transitions of *sequence 1* and satisfies the precedence constraints. This problem is solved as the TSP with precedence constraints [21]. In this way, the reachability graph of the *product flow state model* is generated and each arc corresponding to transitions of *sequence 1* are transformed into an arc, a vertex and an arc, and so precedence constraints are expressed on these new vertices. In the construction of the reachability graph, it should be noted be noticed that a transition "end PE" must not be fired before the corresponding transition "begin PE", and a transition "begin PE" cannot be fired during other PEs of the same operation. The weights of arcs corresponding to "begin PE" represent the durations of the operations; the others are weighted zero. Thus, the problem is equivalent to finding the shortest path, while respecting precedence constraints, from the vertex corresponding to the initial state of the product which will pass through all the introduced vertices. This problem can be solved by the heuristic proposed in [21].

The resulting sequence satisfies the precedence constraints, but not all the pre-constraints on the product flow, such as the product position to launch an operation, for example. Then, all product flow pre-constraints must be checked. In the order of the resulting sequence, for each transition "begin PE" pre-constraints on the product flow must be checked in the *pre-constraint models.* There are checked in the predicate for pre-constraints on physical characteristics, and the marking of the upstream places is checked for the pre-constraints on product positions. This is also true for the corresponding "begin operation" (operation which contains this PE) in the *permanent pre-constraint models.* If a pre-constraint is not satisfied, validation requires firing a transition "begin PE" before the considered operation. Then, this transition is added to *sequence 1* with the associate precedence constraint, and the algorithm jumps to the beginning of step 3. If all the pre-constraints are checked for the considered transition, it is fired, and the pre-constraints must be checked for the following transition of *sequence 1*. Thus, the algorithm jumps to the beginning of step 3. If pre-constraints are checked for all transitions of *sequence 1*, *sequence 2* is obtained (see Fig. 8). It satisfies all the pre-constraints on the product flow; however, the pre-constraints on FCs must also be satisfied; this is the aim of step 4.

4.3.4. 4th step: pre-constraints on the FC

For each transition "begin PE" of *sequence 2,* pre-constraints on FCs must be checked (the marking of the upstream places) in the *pre-constraint models* and for the corresponding "begin operation" (operation which contains this PE) in the *permanent pre-constraint models.* If a pre-constraint is not satisfied, the corresponding FC must be placed in a compatible state with this pre-constraint. This problem can be solved by a shortest path algorithm. To do this, the reachability graph of the considered *FC state model* is generated. The weights of arcs corresponding to "begin operation" are the durations of operations; the others are weighted zero. Thus, the problem is equivalent to finding the shortest path, from the vertex corresponding to the initial state of the FC to the expected final state. In our approach, we have applied the algorithm proposed in [22]. The resulting sequence of transitions is added to *sequence 3* just before the transition for which the pre-constraint was not satisfied. Then, the algorithm returns to the validation of FC pre-constraints. It should be noted that pre-constraints on the added transition "begin operation" must also be checked but only in the *permanent pre-constraint models.* At the end of this step, *sequence 3,* which respects all the pre-constraints, is generated (see Fig. 8). Moreover, the objective given for designing the control law can contain the expected final state of the operating part. This can be considered as a pre-constraint which must be checked at the end of the sequence obtained in 4th step. Thus, operations needed to place the operating part in the expected final state are automatically added.

4.3.5. Optimization of sequence duration

After the fourth step, the algorithm gives an admissible sequence which is locally optimized and satisfies all the constraints. The *permanent constraints models* and the *constraints models* can also be used to optimise the duration of the sequence. Depending on the constraints, several operations can be launched at the same time. The optimization step takes each transition "begin PE" (or "begin operation") and tries to move this transition forward in the full sequence. For a transition "begin PE" (or "begin operation"): to be able to move it before a transition "end PE", there must be no mutual exclusion with these PEs; to be able to move it before a transition "end operation", there must be no mutual exclusion with this operation; to move it before a transition "begin PE", this PE must not modify its pre-constraints; to move it before a transition "begin operation", this operation must not modify its pre-constraints. After this step, operations can be launched as early as is possible without violating constraints. To use this resulting sequence, all the transitions "begin PE" and "end PE" are replaced respectively by the corresponding "begin operation" and "end operation". The operation, in fact, infers its PE according to the state of the controlled system. Each "begin operation" carries out this operation, and the corresponding "end operation" waits for the end report of this operation. If the operation is already finished, the next request is launched directly.

5. Application Example

This section presents an application example based on a loading system (see Fig. 9) of the research platform Saphir at the Institut National Polytechnique de Grenoble, France. This platform is designed for the assembly of camshafts. A four-place rotating storage area is used to receive up to six different kinds of products. These products are identified by a weighing system. Once a product has been identified, it is drive by conveyor belt to a sorting device. A human operator is in charge of filling up the rotating storage and emptying the assembly station.

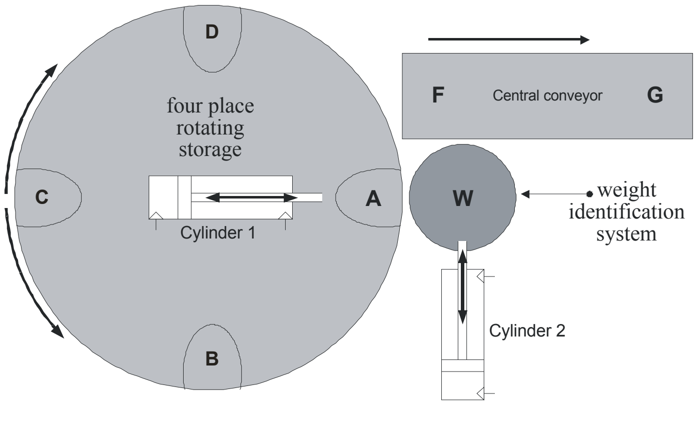
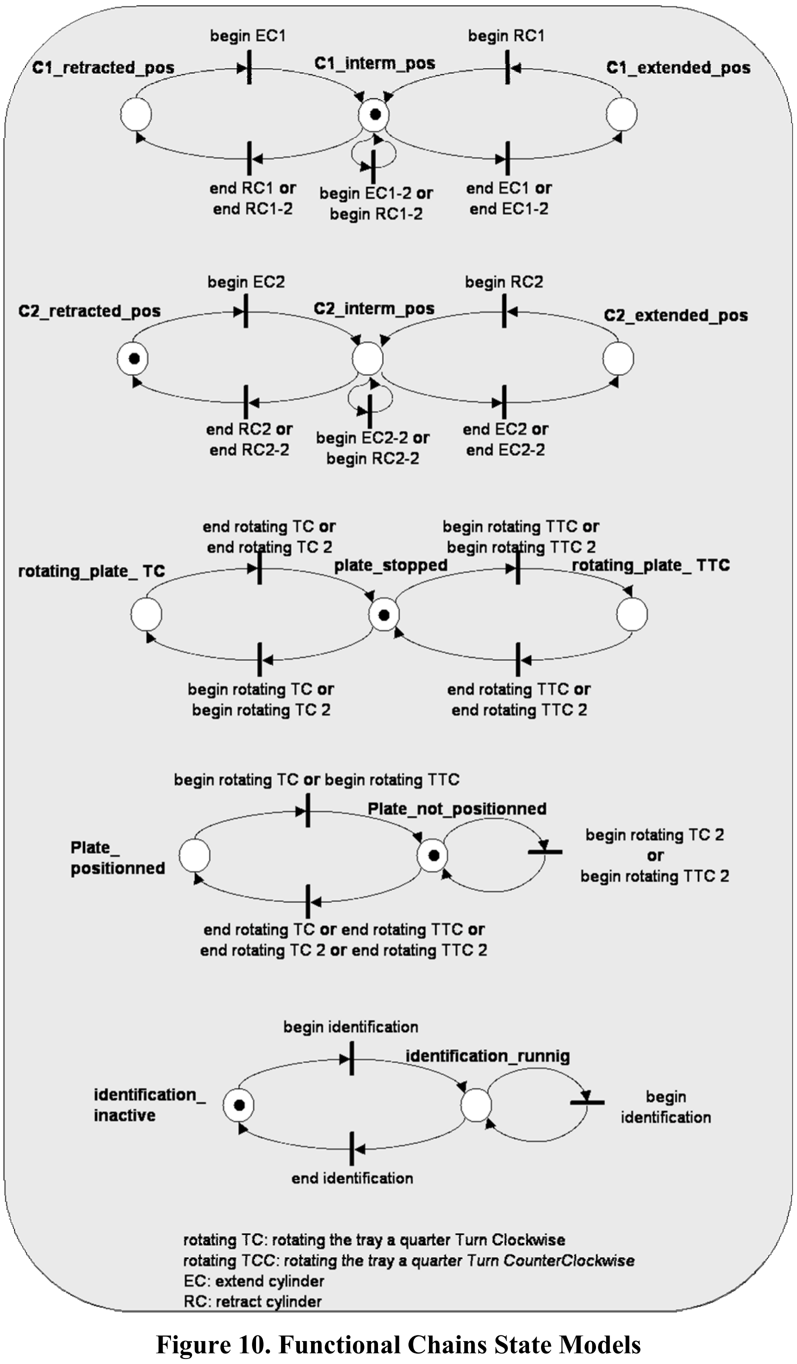


Figure 9. Loading System of Saphir

To explain the modelling steps, the *extend cylinder 1* (EC1) operation is considered: the FC goes (i.e. C1 evolution) from the retracted position to the intermediate position after the launching of the operation, and at the end to the extended position. To make this cylinder 2 (C2) must be retracted before (pre-constraint) and during (constraint) the operation to avoid a collision. For the same reason there should not be a product between A and B and between A and D. All this information is capitalised in the FC evolution (see Fig. 6). Then, the possible evolutions of the product flow are considered. If there is a product in front of the cylinder (in A), the launching of the operation will place the product between A and W, and at the end in W. This time, the rotating storage must be stopped and positioned and the product that is transferred must not collide with another product. This defines the first PE (see Fig. 6). But the initial state of the product could be a product between A and W, without considering how the product is put in this position. So, a second PE must be described for this operation. Thus, the complete description of the operation *extend cylinder 1* is presented in Fig. 6.

The models previously defined are generated from all the operations of the controlled system, after application of the automatic translation into Petri nets. Presenting all the Petri net models in this paper would take too much space, and thus only models used to illustrate the control laws design algorithm are given. Models showing the state of the loading system (see Fig. 3) are presented in Fig. 10 (*FCs state models*) and in Fig. 11 (*product flow state model*). The resulting *permanent pre-constraints model* of the translation of the EC2 operation is presented in Fig. 12. In this figure, places surrounded by a circle shown as a dotted line are the places of the *FCs state models* and the *product flow state model*. Fig. 13 presents the *pre-constraints model* corresponding to the operation *extend cylinder 1*, and Fig. 14 presents the *pre-constraints model* corresponding to the operation *rotating the storage a quarter Turn Clockwise* (TTC). Finally, *permanent constraints models* and *constraints models* are illustrated in Figs. 15 and 16, corresponding to EC1 operation.



begin EC1
begin RC1
C1_retracted_pos
C1_interm_pos
C1_extended_pos
end RC1 or end RC1-2
begin EC1-2 or begin RC1-2
end EC1 or end EC1-2
begin EC2
begin RC2
C2_retracted_pos
C2_interm_pos
C2_extended_pos
end RC2 or end RC2-2
begin EC2-2 or begin RC2-2
end EC2 or end EC2-2
end rotating TC or end rotating TC 2
begin rotating TTC or begin rotating TTC 2
rotating_plate_ TC
plate_stopped
rotating_plate_ TTC
begin rotating TC or begin rotating TC 2
end rotating TTC or end rotating TTC 2
begin rotating TC or begin rotating TTC
Plate_not_positionned
Plate_ positionned
begin rotating TC 2 or begin rotating TTC 2
end rotating TC or end rotating TTC or end rotating TC 2 or end rotating TTC 2
begin identification
identification_runnig
identification_ inactive
begin identification
end identification
rotating TC: rotating the tray a quarter Turn Clockwise
rotating TCC: rotating the tray a quarter Turn CounterClockwise
EC: extend cylinder
RC: retract cylinder

Figure 10. Functional Chains State Models

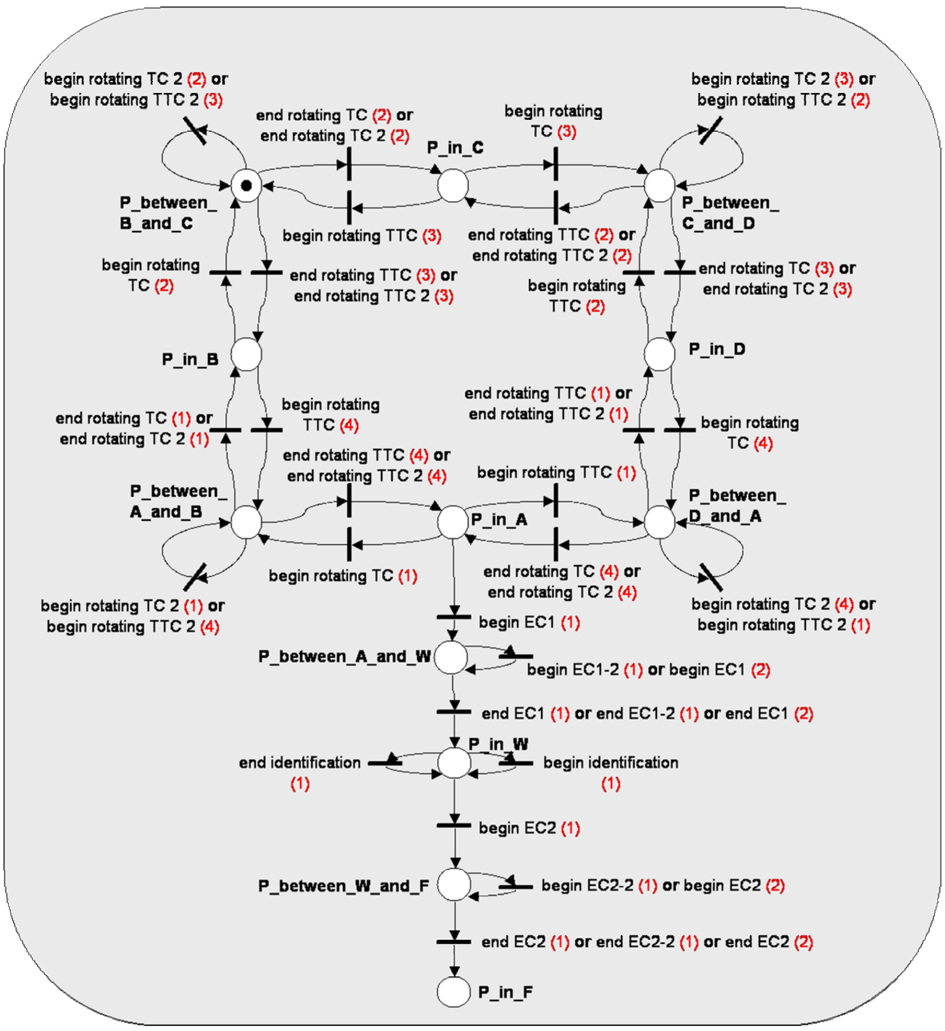

begin rotating TC 2 (2) or begin rotating TTC 2 (3)
end rotating TC (2) or end rotating TC 2 (2)
P_in_C
begin rotating TC (3)
begin rotating TC 2 (3) or begin rotating TTC 2 (2)
P_between_ B_and_C
begin rotating TTC (3)
end rotating TTC (2) or end rotating TTC 2 (2)
P_between_ C_and_D
begin rotating TC (2)
end rotating TTC (3) or end rotating TTC 2 (3)
begin rotating TTC (2)
end rotating TC (3) or end rotating TC 2 (3)
P_in_B
P_in_D
begin rotating TTC (4)
end rotating TTC (1) or end rotating TTC 2 (1)
end rotating TC (1) or end rotating TC 2 (1)
begin rotating TC (4)
end rotating TTC (4) or end rotating TTC 2 (4)
begin rotating TTC (1)
P_between_ A_and_B
P_in_A
P_between_ D_and_A
begin rotating TC (1)
end rotating TC (4) or end rotating TC 2 (4)
begin rotating TC 2 (1) or begin rotating TTC 2 (4)
begin rotating TC 2 (4) or begin rotating TTC 2 (1)
begin EC1 (1)
P_between_A_and_W
begin EC1-2 (1) or begin EC1 (2)
end EC1 (1) or end EC1-2 (1) or end EC1 (2)
P_in_W
end identification (1)
begin identification (1)
begin EC2 (1)
P_between_W_and_F
begin EC2-2 (1) or begin EC2 (2)
end EC2 (1) or end EC2-2 (1) or end EC2 (2)
P_in_F

Figure 11. Product Flow State Model

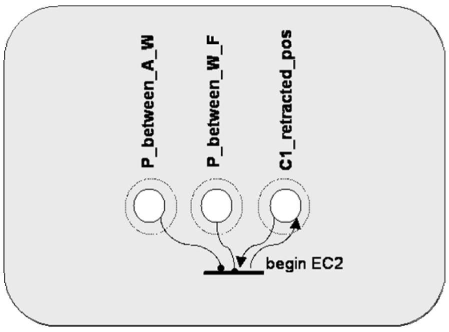


Figure 12. Permanent Pre-Constraints Model of EC2

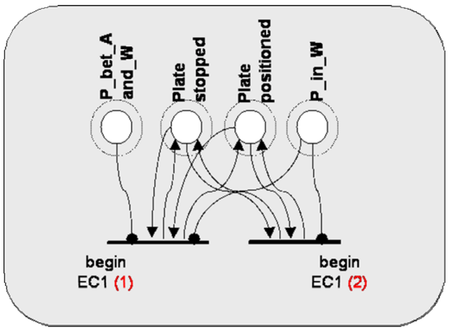


Figure 13. Pre-Constraints Model of EC1

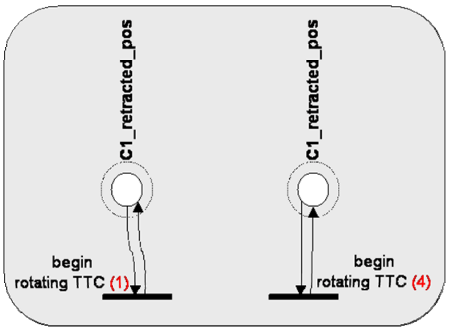


Figure 14. Pre-Constraints Model of “Rotating TCC”

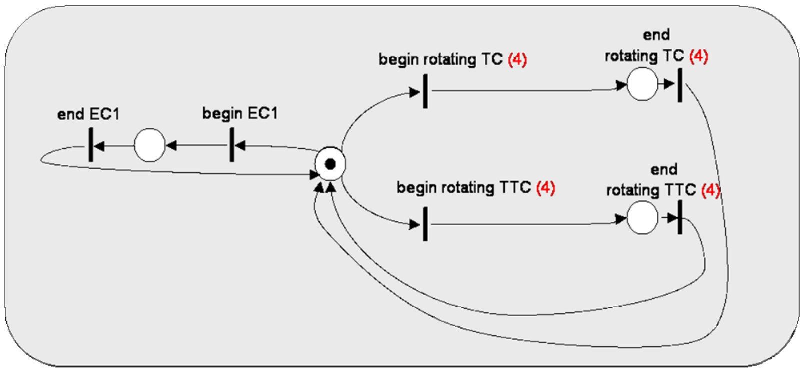


Figure 15. Permanent Pre-Constraints Model of EC1

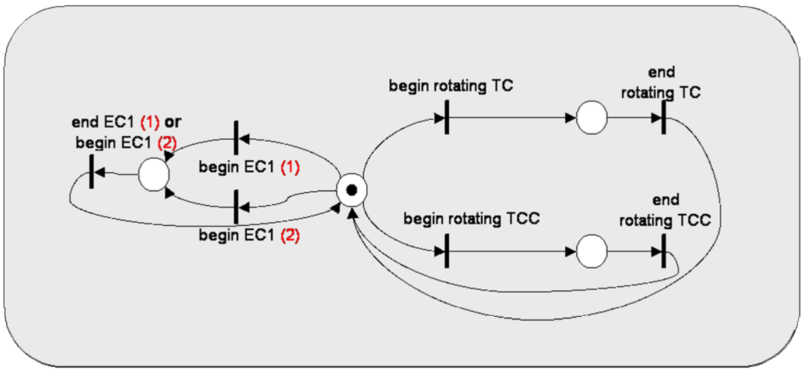


Figure 16. Constraints Model of EC1

To illustrate the control laws design algorithm, the following objectives of the control law are considered. The considered operating part of the Saphir platform is stopped. Let us assume that the state of the controlled system is: a product is located between B and C, C1 is extended and C2 is retracted (after a failure, for instance). The production objectives are an identified product on the conveyor belt (in F). The requested final state of the operating part (corresponding to the initial state in normal running) is: cylinders in retracted position, and the rotating storage stopped and positioned.

Then, the first sequence must be defined. The product specifications are an identified product on the conveyor belt (in F). So the first sequence is composed of the two transitions. One corresponds to the transition which modifies from *not identified* to *identified* the attribute of the product token. The second, corresponding to the final location of the product, is the input transition of the place *product in F*. Consequently, sequence 1 is composed of two PEs: *[End identification(1), End EC2(1)]*. In the example, no precedence constraint is obtained. However, when there is more complex transformation of the product, for example for a tapping, the drilling operation must be launched before the tapping operation. To generate the full sequence corresponding to expected product evolutions, the algorithm proposed by [21] applied to the modifying reachability graph of the *product flow state model* (see Fig. 11) gives the following sequence 2: *[Begin rotating TTC 2 (), End rotating TTC 2 (), Begin rotating TTC (), End rotating TTC (), Begin EC1 (), End EC1 (), Begin identification (), End identification (), Begin EC2 (), End EC2 ()]*. The operation *rotating TTC 2* is similar to operation *rotating TTC* except for the initial state of the FC. In this sequence, there is only one product, so all constraints on product flow are respected. Thus the next step is the checking of FC constraints. In the example, if C1 is in extended position and if there is a product in B, the *pre-constraints model* presented in Fig. 14 prevents the firing of *begin rotating TTC*. So, before the second rotation, C1 must be retracted. To extend C2 (see Fig. 12), C1 must also be retracted. In these two cases, the algorithm adds the transitions *Begin RC1* and *End RC1* before these two operations. These transitions are obtained with the algorithm proposed by [22] applied to the reachability graph of the *FC state model* of C1 (see Fig. 10). Finally, the last constraint on FCs is their expected final states, given in the objectives. Thus, the expected state of the controlled system does not correspond to the final state after the firing of the sequence of transitions. C2 must be retracted, thus the algorithm adds the transitions *Begin RC2* and *End RC2* at the end of the sequence. Finally, the resulting sequence 3 is: *[Begin rotating TTC 2(), End rotating TTC 2(), Begin RC1, End RC1, Begin rotating TTC(), End rotating TTC(), Begin EC1(), End EC1(), Begin identification(), End identification(), Begin RC1, End RC1,Begin EC2(), End EC2(), Begin RC2, End RC2]*. The last step optimizes the sequence; the algorithm moves forward the two transitions *Begin RC1*. After having replaced PE by the corresponding operation, the final sequence is:

[Begin RC1, Begin rotating TTC, End rotating TTC, End RC1, Begin rotating TTC, End rotating TTC, Begin EC1, End EC1, Begin RC1, Begin identification, End identification, End RC1, Begin EC2, End EC2, Begin RC2, End RC2]

Thus, the first operation *RC1* is launched at the same time as the operation *rotating TTC*, and the operation *identification* at the same time as the operation *RC1*.

6. Conclusion and Future Works

In this paper, an approach to control laws design for discrete event system has been presented. An innovative aspect of this approach is to solve the global problem, from the controlled system modelling to the automated control laws design. A data structure for the knowledge acquisition is provided to help engineers give all the required data concerning the controlled system. It is easily updated if there is a modification of the controlled system. To validate the model but also to facilitate the control laws design, the data structure is translated into a formal model using Petri net formalism. And thus, from this model, optimized control laws are computed. However, it should be noted that the proposed control laws design algorithm does not yet guarantee an optimal solution, so only local optimisations are proposed during the design of the control laws.

For this reason, our future works will first focus on the improvement of the proposed control laws design algorithm. It should also be improved to obtain cyclical control laws using the IEC61131-3 standard. Secondly, a control laws design algorithm of this type has to be considered in a dynamic context, especially considering failure recovery.

REFERENCES

1. IEC-61131-3, Programming Languages - Providing the Basis, Swiss, 1993.

2. CIM, **A reference model for computer integrated manufacturing from the viewpoint of industrial automation,** International Journal of Computer Integrated Manufacturing, Vol. 2 (2), 1989.

3. COMBACAU, M., BERRUET P., ZAMAÏ E., CHARBONNAUD P., and KHATAB A., **Supervision and monitoring of production systems**, Proc. IFAC 2nd Conference on Management and Control of Production and Logistics, Grenoble, France, 2000.

4. ZAYTOON, J., **On the recent advances in grafcet**, Production Planning and Control, Vol. 13 (1), 2002.

5. GOUYON, D., PETIN J. F., and GOUIN A., **Pragmatic approach for modular control synthesis and implementation**, International Journal of Production Research, Vol. 42 (14), 2004.

6. TOGUYENI, A. K. A., BERRUET P., and CRAYE E., **Models and algorithms for failure diagnosis and recovery in FMSs**, International Journal of Flexible Manufacturing Systems, Vol. 15, 2003.

7. CHRETIENNE, P., COFFMAN E. G., LENSTA J. K., and LIU Z., **Scheduling theory and its applications**, John Wiley and Sons (Eds.), 1997.

8. QIU, R., WYSK R., and XU Q., **Extend structured adaptative supervisory control model of shop floor controls for an e-manufacturing system**, International Journal of Production Research, Vol. 41 (8), 2003.

9. CHARBONNIER, F., ALLA H., and DAVID R., **The supervised control of discrete-event dynamic systems**, IEEE Transactions on Control Systems Technology, Vol. 7 (2), 1999.

10. HOLLOWAY, L. E., GUAN X., SUNDARAVADIVELU R., and ASHLEY JR. J., **Automated synthesis and composition of task blocks for control of manufacturing systems**, IEEE Trans. On Systems, Man and Cybernetics, Part. B, Vol. 30 (5), 2000.

11. BERRUET, P., MOUCHARD J.S., PHILIPPE J.L., and GUYOMAR J.P., **Modeling and validation of transitic systems based on reusable components**, Proc. 5th Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA, 2001.

12. RAMADGE, P. J. and WONHAM, W. M**., Supervisory control of a class of discreet event processes**, Journal of Control and Optimization, Vol. 25 (1), 1987.

13. ZAMAI, E., SUBIAS A., and COMBACAU M., **An architecture for control and monitoring of discrete events systems**, Computers in Industry, Vol. 36 (1, 2), 1998.

14. GHALLAB, M., NAU D., and TRAVERSO P., **Automated Planning**, Theory and Practice, Elsevier, 2004.

15. SANDEWALL, E. and RONNQUIST R., **A representation of action structures**. Proc. National Conference on Artificial Intelligence, 1986.

16. CASTILLO, L., FDEZ-OLIVERAS J., and GONZALES A., **Automatic generation of control sequences for manufacturing systems based on partial order planning techniques**, Artificial Intelligence in Engineering, Vol. 14 (1), 2000.

17. GIRAULT, C. and VALK R., **Petri Nets for Systems Engineering**: A Guide to Modeling, Verification, and Applications, Springer-Verlag, 2002.

18. MENGCHU, Z., DICESARE F., **Petri net synthesis for discrete event control of manufacturing systems**, Kluwer Academic Publishers, 1993.

19. DAVID, R. and ALLA H., **Petri nets and grafcet: tools for modelling discrete event systems**, Prentice Hall, 1992.

20. LEE, D.Y. and DICESARE F., **Scheduling Flexible Manufacturing Systems Using Petri Nets and Heuristic Search** , IEEE Trans. On Robotics and Automation, Vol. 10 (2), 1994.

21. BIANCO, L., MINGOZZI A., RICCIARDELLI S., **Dynamic programming strategies and reduction techniques for the travelling salesman problem with time windows and precedence constrints**, Operations Research, Vol. 45 (3), 1997.

22. DIJKSTRA, E.W., **Note on two problems in connection with graphs**, Numerische Mathematik, Vol. 1, 1959.