A pragmatic approach for modular control synthesis and implementation

David Gouyon, Jean-François Pétin, Alexia Gouin

A pragmatic approach for modular control synthesis and implementation

D. GOUYON†*, J.F. PETIN† and A. GOUIN‡

Abstract:
Within the framework of Supervisory Control Theory, synthesis algorithms enable automatic generation of control rules from behavioural models of the process to be controlled and of goals to be achieved. This paper talks about a pragmatic use of these algorithms within an automation engineering context and focuses on two key issues: process and goals modelling and supervisory controller implementation. In this way, the approach presented in this paper combines the synthesis techniques and algorithms with an object-oriented automation method that supplies guidelines for the analysis, the design and the implementation of a modular control system. This pragmatic approach is applied to the synthesis and implementation of the control of an assembly station. It highlights the difficulties of the modelling step and its great influence on the synthesis result itself but also demonstrates the feasibility of our synthesis approach through a successful implementation in conformance with the IEC 61131-3 standard.

Keywords:
Logic controllers, modular control synthesis, supervisory control, automation engineering, control specification, control implementation

1. Introduction

Automatic synthesis of control systems, as proposed by (Fusaoka *et al.* 1983), consists in defining the (unknown) control rules of the (known) dynamics of a physical system, starting from the behavioural goals (known) to be met, while satisfying the following condition: *Control Rules* $\wedge$ *Dynamics* $\supset$ *Goal*. In this way, the Supervisory Control Theory (SCT) (Ramadge and Wonham, 1987) defines a formal framework aiming to analyse Discrete Event Systems (DES) and provides algorithms that enable an automatic synthesis of supervisory controllers in such a way that the controlled plant behaves according to some given goals. Even if these various algorithms (Wonham and Ramadge 1987, Kumar *et al.* 1991) have demonstrated their efficiency, industrial applications of synthesis techniques in the area of automated manufacturing systems remain very limited.

† Centre de Recherche en Automatique de Nancy, UMR 7039, CNRS – Université H. Poincaré – INPL, Campus scientifique - BP239, 54506 Vandoeuvre-lès-Nancy Cedex, France
‡ Laboratoire d'Automatique de Grenoble, UMR 5528, CNRS – INPG – Université J. Fourier, ENSIEG - BP46, 38402 St Martin d'Hères, France
* To whom correspondence should be addressed. e-mail: david.gouyon@cran.uhp-nancy

To explain this situation, several reasons can be pointed out (Zaytoon and Carré-Ménétrier 2001): the synthesis algorithms demand correct models of the system and of its goals that are not easy to capture in an industrial context (Morel *et al.* 2001), synthesis may lead to supervisory controllers having an unrealistic size due to a possible explosion of the state space, the model interpretation given by SCT is more permissive than the one required to implement deterministic control systems.

In order to cope with these difficulties, the approach proposed in this paper takes advantages of combining SCT formal framework and synthesis algorithms with automation engineering methods based on the re-use of control software objects (Combacau and Courvoisier 1990, Elkhattabi *et al.* 1992, Lhoste and Morel 1996, Feliot and Staroswiecki 1998, Tiller 2001). It leads to a pragmatic approach where modular supervisory controllers are obtained by applying an iterative algorithm of synthesis and are implemented inside Programmable Logic Controllers (PLC) by translating them into functional control blocks in conformance with the IEC 61131-3 standard (IEC 1993) recommendations.

This paper is organised as follows. Issues about the process and goals modelling as well as implementation in the SCT framework are given in section 2. In section 3, an approach combining automation engineering methods with synthesis techniques is presented. In section 4, this approach is applied to a station of the AIP-Primeca Lorraine experimental assembly system. Results of this application, especially those dealing with the modelling step and its great influence on the synthesis result itself are discussed in section 5. At least, in section 6, some conclusions and future works are drawn.

2. Modelling and implementation issues within SCT framework

2.1. SCT framework

The supervisory control theory (Ramadge and Wonham 1987) provides a formal framework for DES analysis based on the models of the process to be automated (called *generator*) and the supervisory controller (called *supervisor*).

The process model is described by an automaton of the following form:

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

where Q is a set of states q, Σ is a non-empty set of event labels called an *alphabet*, δ is transition function described by states transitions, $Q_m \subseteq Q$ is the set of *marked* (terminal) *states* and $q_0 \in Q$ is the initial state.

It is assumed to 'generate' spontaneously controllable and uncontrollable events. Controllable events are those whose occurrence can be disabled while uncontrollable events can not be prevented and are permanently enabled.

The supervisor can affect the behaviour of the process model by enabling or disabling controllable events to maintain the process in a space of acceptable states for a given specification. It is described by an automaton of the following form:

$$S = (X, \Sigma, \xi, X_m, x_0)$$

where X is a set of states x, Σ is the alphabet used by G, ξ is transition function, X_m is the set of *marked states* and x_0 is the initial state.

In this framework, synthesis algorithms (Wonham & Ramadge 1987, Kumar *et al.* 1991) generate automatically the optimal supervisor that enables the maximum set of events that do not contradict the goal specifications. These specifications define the enabled sequences of events that belong to G alphabet. Considering the generator model as the physically possible sequences, and the goal specifications as the legal sequences, the synthesised supervisor is expected to inhibit the process behaviour so that only desirable sequences are generated. Benefit is that the generated supervisor is correct, by construction, with regards to the specifications and deadlock-free.

2.2. Modelling issues

Executing synthesis algorithms requires concrete and detailed models of the process behaviour and of the goal to achieve which are more or less considered as given inputs of the control design activities. However, practical automation of complex systems often relies on a progressive refinement of the models (Brandin *et al.* 2000, Hiraishi 2001) that are not considered within the engineering process as fixed points. Indeed, the modelling of systems of industrial complexity requires a preliminary analysis, based on the use of abstract and more or less formal representation formalisms, under a shape of function, set, … allowing to grasp, in a pragmatic, intuitive, even qualitative way, the global functioning of a system to be designed (Marikar *et al.* 1998).

If we consider the SCT formalisms and methods for process and goal modelling, it clearly appears that:

– more structured representations such as Petri Nets (Xie 1994), synchronous languages (Marchand *et al.* 2000) or predicate (Sanchez and Macchieto 1995) and temporal (Fusaoka *et al.* 1983) logics would be preferable to capture the behaviour of complex systems,

– the definition of a methodological framework, that could help the designer to progressively transform the end-user's requirements and the plant operations into formal specifications of the expected behaviour and process dynamics, should be very helpful to facilitate the use of synthesis techniques within an automation life cycle.

A well known answer to these needs consists in decreasing the complexity of the modelling phases by promoting a decomposition strategy that leads to introduce modularity and hierarchy within the SCT framework. In the field of supervisory control, the interest of modularity has been widely demonstrated through extensions of the theoretical frame that propose:

- distributed (Figure 1) (Fen and Wonham 1990) or hierarchical (Zhong and Wonham 1990, Gohari and Wonham 1998) decomposition of the supervisory controllers to be synthesised that implicitly requires a decomposition of the associated goals specifications,

- modular modelling of the process (*generator*) to be controlled (Yoo and Lafortune 2002) (Figure 2)

- mixed approaches where both process models and supervisory controllers are modelled in a modular way (Chafik and Niel 2000).

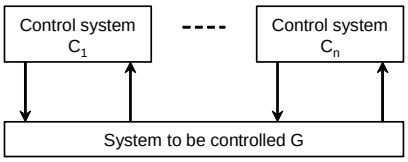


Figure 1: Distributed supervisory control (Fen and Wonham 1990)

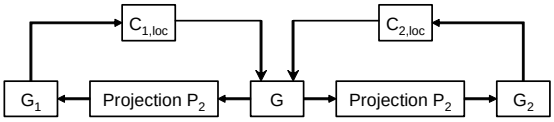


Figure 2: Decentralised supervisory control

Whatever the type of decomposition is, the objective is mainly to decrease the explosion of state space (Queiroz and Cury 2002) generated by the algorithms of synthesis, notably by processing these algorithms on smaller models that involves a reduced number of states and transitions. In this way, the chosen criteria for the decomposition of the models are those who lead to a set of automata that minimizes the intersections between their alphabets, in order to be able to synthesize, in an independent way, several supervisors of smaller size.

The benefit of these approaches is real to master the complexity of the modelling phase. However, focusing the decomposition rules on the only problem of state space size hide more or less the methodological aspects involved by modularity and hierarchy within control systems.

2.3. Implementation issues

Implementation issues must be addressed in order to fully use synthesis techniques within an automation life cycle. The main problem results from the different interpretations of the supervisor model given by SCT and by traditional automation methods for the design and implementation of real-time control systems.

Indeed, within the framework of the SCT, the process (generator) is supposed to generate events in a spontaneous way; the only way for the supervisor to affect the behaviour of the process is then to enable or to disable the controllable events. Moreover, the synthesis algorithms (Wonham and Ramadge 1987, Kumar *et al.* 1991) provide the maximal permissive supervisor that maintains the process behaviour in legal states and sequences for a given specification. So, the result can be described by automata where several sequences of controllable events are enabled from a given state.

Nevertheless, a reactive control system is expected to force some events to occur and not only to enable and disable some of them. It is often based on a set of predetermined evolution rules that calculate the appropriate actions (controllable events) to be applied on the process according to observations (uncontrollable events) of its current state. This computation is said deterministic in the sense that a given sequence of observations always generates the same sequence of actions.

These various interpretations of a what is called *supervisor* by SCT and *controller* by the forcing events approaches (Marikar *et al.* 1998) requires some additional features to make these two notions compliant with implementation issues.

A first solution considers that real systems require the addition of an external control agent that forces some events to occur. It consists in modifying the SCT theoretical framework by adding an intermediate model dedicated to the reactive and deterministic process control (Figure 3) (Charbonnier *et al.* 1999). In this case, process and control models are supposed to be known. Generated supervisor aims at ensuring that process associated with its control behaves according to given specifications. This approach is too restrictive for our initial objectives of synthesis and implementation.

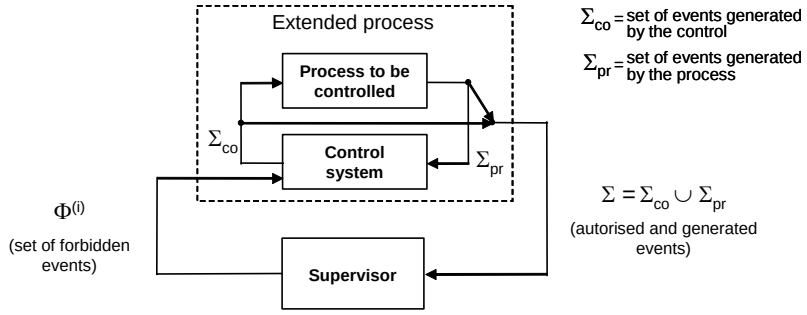


Figure 3: Supervisory control principle according to (Charbonnier *et al.* 1999)

Another approach consists in preserving the SCT theoretical framework while introducing an input-output interpretation of the SCT controllable and uncontrollable events (Balemi *et al.* 1993). This reactive interpretation requires that every supervisor sequences must have at least one controllable event between two uncontrollable events. Indeed, it means that the controller must react to a given input by emitting an output before a new input occurrence. Assuming this hypothesis is verified, this interpretation can be applied for implementation of SCT supervisors (Brandin 1996, Niel *et al.* 2001).

At least, more direct approaches that transform a SCT supervisor into a controller able to be implemented have been proposed (Sanchez and Macchieto 1995 (Figure 4), Marikar *et al.* 1998, Fabian and Hellgren 1998). This transformation requires removing the non deterministic choices contained in the supervisor and to translate the evolutions rules of a finite state automaton in terms of target languages such as the Ladder Diagram of the IEC 61131-3 standard (IEC 1993). The benefit is that SCT approach is kept without any additional interpretation constraints. However, in order to guarantee that the controller maintains the properties of the supervisor, such as controllability and deadlock-freeness, a formal translation mechanism is required. Most of the above mentioned approaches are based on the representation of automaton behaviour in terms of algebraic equations and a partial order relation between equations for their implementation.

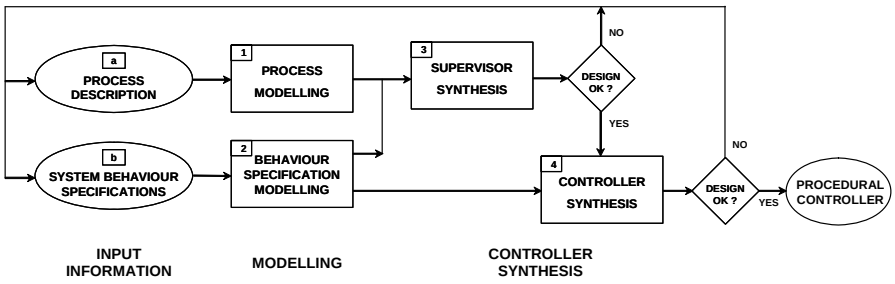


Figure 4: Controller synthesis procedure (Sanchez and Macchieto 1995)

The approach presented in this paper conserves the basic foundations of SCT. It aims at defining a methodological framework for using this theory within a real automation life cycle (Figure 4). More particularly, modelling and implementation issues will be addressed by combining existing automation methods that provide guidelines for a structured design and implementation of control systems and the SCT framework.

3. An approach for modular control synthesis and implementation

3.1. Automation engineering

During last decade, industrial practices in automation engineering have promoted the re-use of 'on-the-shelf' control components in the same way of generic hardware components that exist in Electronics. From a technical point of view, this is justified by the development of distributed control system (remote I/O, fieldbus, intelligent actuators and sensors) and by the introduction of the functional block concept in the programming languages such those supplied by the IEC61131-3 (IEC 1993) or IEC61499 (IEC 2000) standards.

Automation methods have followed the same evolution. Object oriented reasoning that included modularity and hierarchy in the design of control systems has been widely explored (Combacau and Courvoisier 1990, Elkhattabi *et al.* 1992, Lhoste and Morel 1996, Feliot and Staroswiecki 1998, Pétin *et al.* 1998). These methods are more or less based on bottom-up decomposition of a control system in terms of functional elementary modules that ensure the control and the monitoring of their associated resources. In these bottom-up approaches, structured design starts by the process modelling.

Manufacturing systems appear to be built with similar technological elements - cylinders, pumps, motors – involved in the various actuation and measurement processes. Most of the time, these field devices are standard and they can be associated with a logical behaviour independent from their context of use. It results that the behaviour of such a manufacturing system can be represented as an orderly network of interconnected elementary behaviours (Tiller 2001) that could be plugged from a library of standardised components.

An original application of this work considers that a field-device behavioural model can be used as model for low level control and monitoring. It modifies the classical distribution 'Control system / Process system' by adding a local control of the field devices acting as an interface between technology and functional features (Figure 5) (Vogrig *et al.* 1987) in order:

– to filter the functional requests submitted by the control system and to compute appropriate actions to be applied on the technological device in such a way that these actions are compliant with the device current state and status,

– to filter the observations supplied by the device sensors and to compute reports about device state and status to be sent to the control system in such a way that these reports are consistent with the expected behaviour.

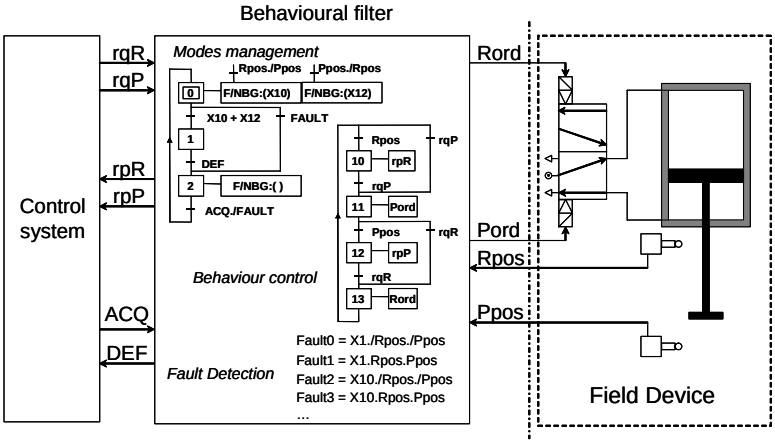


Figure 5: Device control placed as an interface

In other words, the 'Behavioural Filter' constitutes a step towards autonomous agents that are able to manage their mission according the current state of their resources, to monitor themselves and to elaborate a synthetic report about their functioning state. It could be implemented on supports such as PLC, distributed remote I/O or embedded electronics in the case of intelligent actuators and sensors. This clear distinction between what refers to functional control objectives of the application and what depends on the field devices technology is efficient to increase control flexibility and adaptation to changing needs and technology.

Generalization of this approach at less technological levels leads to iteratively aggregate basic behaviours to built more complex ones in a bottom-up way. The resulting hierarchical architecture involves a set of coordinated and/or cooperative modules having to control and monitor the execution of their own mission (Combacau and Courvoisier 1990, Elkhattabi *et al.* 1992, Lhoste and Morel 1996). These modules, called '*automation object'*, are able to accept or not the requests they receive, to chose the appropriate actions they will transmit to their resources (lower level modules), to monitor the execution of these actions and to report to the higher level modules.

3.2. Modular control synthesis

Our approach combines the iterative way of thinking proposed by the automation object-oriented methods and the modular synthesis techniques:

– criteria used for structuring the control system are not only driven by state-space explosion issues (Queiroz and Cury, 2002), but must be given by the structure of the physical process itself; it leads to an iterative bottom-up design starting from the field device models,

– synthesis algorithms (Wonham & Ramadge 1987) will be used to automatically generate a supervisor associated to a module (or *automation object*) of the control architecture; for a given module, the supervisor will be computed from specification describing the mission allocated to this module and from process model (*generator*) given by the lower level supervisors associated to its resource modules.

Like 'behavioural filter', these supervisors ensure the orderly occurrence of events into supervised modules by filtering functional requests and observations in order to separate functional aspects from technological ones and to guarantee consistency with an expected behaviour.

Let us consider the synthesis of the supervisors associated to the control of the field devices. These supervisors (noted S1 and S1' in Figure 6) are allocated to the level 1 in the control hierarchy scale. Two supervisors of the same level do not share events because each part of the system to be controlled has only one supervisor. Supervisors can be synthesised, in a classic way, from the elementary behavioural models of the devices (noted P1 and P1' in Figure 6) and from specifications describing the rules of filtering (Figure 5) presented in section 3.1.

A supervisor of higher levels (level n >1) in the control hierarchy scale can then be synthesised from the specification of its goals, which are of coordination of lower level resources, and from a process model given by:

– projection of the lower level supervisors (level n-1) considered as required resources in order to keep the only observable events from the given level (n),

– controllability status modification: kept events that were controllable in the lower level supervisors are seen as uncontrollable by the higher level and vice-versa,

– synchronous product of the above projections (they do not share events).

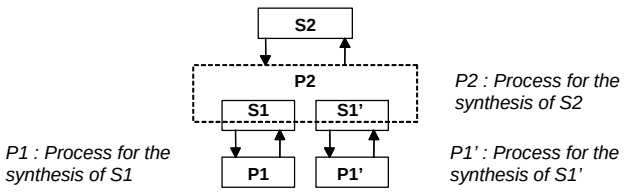


Figure 6: Iterative synthesis process

Deadlock freeness is preserved when synthesising supervisors at level n because:

– level n-1 supervisors are deadlock free by construction (property ensured by synthesis techniques),

– we assume that level n-1 supervisors have disjoint alphabets; consequently no deadlock problem may be introduced when coordinating these supervisors.

This last assumption is justified in the case of a hierarchical coordinated architecture where same level modules do not directly exchange information. However, it is not verified in the case of heterarchical architecture where both coordination and cooperation between same level modules are enabled. In this second case, preserving properties assume conditions, as studied in works on decentralized control (Jiang & Kumar 2000), such as prefix-closure of the alphabets.

Expected behaviour is defined through specification dedicated to each supervisor to be synthesised. Assuming the same hypothesis about the control architecture, these specifications are disjoint. Consequently, conformance of each supervisor at each hierarchical level with its own expected behaviour is preserved by synthesis techniques.

All the operations upon automata – synchronous product, projection, supervisor synthesis – are performed using the software tool TCT developed at the University of Toronto and downloadable at http://odin.control.toronto.edu/DES/.

3.3. Modular implementation of supervisors

Previous synthesis step results in a hierarchical set of supervisors. Current step aims at implementing these supervisors in a target language supported by most of the Programmable Logic Controllers (IEC 1993). As discussed in section 2.3 several ways can be used to achieve this implementation objective. Chosen approaches are the ones that provide translation mechanisms from supervisor to controller (Marikar *et al.* 1998, Fabian and Hellgren 1998) without modifying SCT framework (Charbonnier *et al.* 1999) and interpretation (Balemi *et al.* 1993).

In these approaches, two main steps have to be performed: translation from the most permissive supervisor into a deterministic controller and coding the controllers into a programmable language. Our proposal adapt the translation and coding rules proposed by Fabian and Hellgren (1998) in order to take into account the modular structure of our supervisors.

Translation rules have to simplify the supervisors in such way that the following property is satisfied: two transitions t1 and t2 may exit the same state if and only if the two events associated to both transitions are uncontrollable. This strong hypothesis is justified by the fact that:

– two controllable transitions from a same state mean that two actions are possible from a given situation while a reactive controller have to force one of them.

– two transitions, one controllable and the other uncontrollable, may generate non deterministic reaction depending on the sampling period where events are seen.

To avoid these kinds of situations, a mechanism of priority allocation must be applied. In the first case, priority will be given to the event that belongs to an alphabet of highest level supervisor (in Figure 7, *report* have higher priority than *action*); if the two events have the same hierarchical level, allocating priority refers to a designing choice. Whatever the choice is, the controller is nevertheless proved to maintain the process in enabled states. In the second case, priority depends on the controllability of the events: uncontrollable events have higher priority than the

controllable ones (in Figure 7, *observation* has higher priority than *action*), preserving the reactivity property and the orderly occurrence of events.

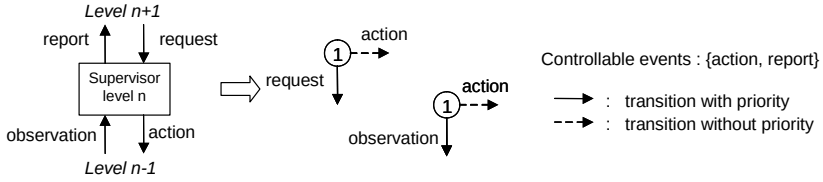


Figure 7: example presenting the priorities used

Coding rules of such deterministic controllers into the chosen target programmable language, which is Ladder Diagram of the IEC 61131-3, is based on algebraic equations that represent the synchronous activation (Ai_t) and deactivation (Di_t) of a state (Si_t) according to $Si_{t+1} = Ai_t \vee (Si_t \wedge \neg Di_t)$ (Figure 8).

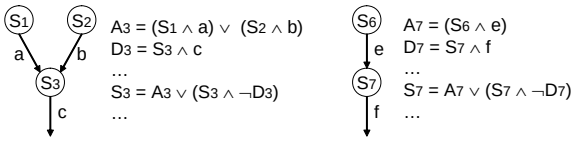


Figure 8: Algebraic translation by activation/deactivation

Implementation of these algebraic equations on a sequential machine requires an executing algorithm that evaluates 'activation' variables before 'state' variables in order to ensure non-blocking and reactivity properties of the implemented controller. Chosen algorithm is said '*without stability research*' and consists, for each sampling period of the PLC, in: reading external events (uncontrollable), evaluation of the new situation (Ai and Di calculation), deactivation and activation of states (updating Si), calculation of internal events and output (controllable) writing.

At least, the hierarchical structure of our synthesised supervisors and controllers is coded using the Function Blocks notation provided by the IEC 61131-3 standard.

3.4. Overview of the proposed approach

The approach here above presented is based upon an iterative way of reasoning that leads to a modular control synthesis. It can be summarized by the Figure 9 which shows the various stages of synthesis, projection, composition and coding of supervisory controllers. Next section demonstrates the feasibility of our proposal through a case study from initial steps of modelling until to the implantation stages.

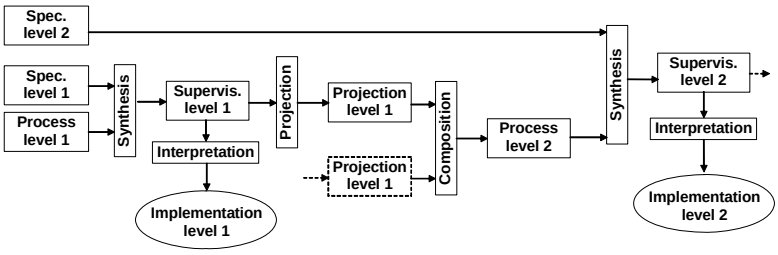


Figure 9: Overview of modular control synthesis

4. Application

Application focuses on the control of a pneumatic manipulator that is involved in an assembly station of the manufacturing cell of the 'Atelier Inter-Etablissement de Productique Lorrain' (AIP-PRIMECA Lorraine). It allows product (part) move from a picking post to a placing one following a 'U' cycle (Figure 10). The horizontal and vertical moves are performed by double-acting air cylinders provided with magnetic position detectors and respectively piloted by a 5/2 bi-stable and monostable solenoid valve. Holding system is carried out by a system of vacuum generator with Venturi effect. The control system is made up of a Siemens S7 PLC.

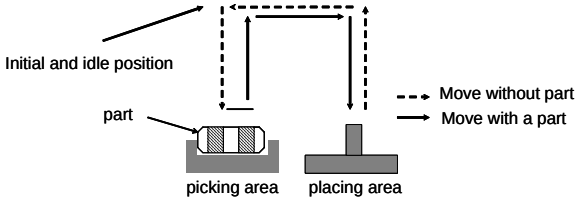
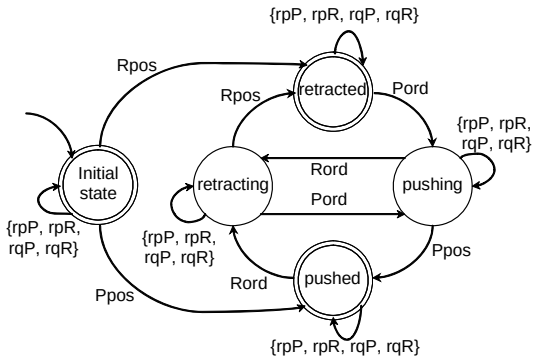


Figure 10: Overview of the pneumatic manipulator

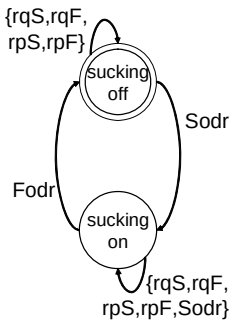
4.1. Synthesis of supervisors associated to field devices control

4.1.1. Process models

The double-acting air cylinders with their control valve are represented (Figure 11a) by two marked states, drawn by a double circle, in which the observable devices behaviour is steady (cylinder in a pushed or retracted position) and by two moving states which represent the evolution from a steady state to another one. The model of the holding system (vacuum generator) is described by two steady states: 'sucking on' or 'sucking off' (Figure 11b). The vacuum generator being not equipped with a held part detector, a part will be considered as being held as soon as the vacuum generator is in 'sucking on' state. A complete list of events is provided in Table 1.



a) air cylinder with a 5/2 bi-stable valves



b) holding system

Figure 11: Models of the field devices

Label	Significance	Controllabilty
rqR	Cylinder Retracting request	uncontrollable
rqP	Cylinder Pushing request	uncontrollable
rpR	Cylinder Retracted position report	controllable
rpP	Cylinder Pushed position report	controllable
Rord	Cylinder Retracting order	controllable
Pord	Cylinder Pushing order	controllable
Rpos	Cylinder in retracted position (given by detectors)	uncontrollable
Rpos	Cylinder in retracted position (given by detectors)	uncontrollable
rqS	Sucking request to vacuum cups	uncontrollable
rqF	Freeing request for vacuum cups	uncontrollable
rpS	Sucked Part report	controllable
rpF	Free Part report	controllable
Sord	Sucking order for vacuum cups	controllable
Ford	Freeing order for vacuum cups	controllable

Table 1: events for figures 11 – 15

4.1.2. Specifications of field devices control

The specification of the devices behaviour is done according to the basic filtering functions assigned to the ‘behaviour filters’ presented in section 3.1 without considering monitoring and fault detection aspects.

For the air cylinders, two automata describe rules that filter or validate some pushing and retracting requests according to the device current state. For example, a pushing request generate a pushing order if the cylinder is not already in ‘pushed state’ and if no ‘retracting request’ is active (Figure 12).

Two other automata describe the observations filtering: an event occurring from a position detector (‘retracted’ for example) may generate a report (‘retracted report’) only if it follows an order (pushed order) that has been computed and sent to the solenoid valve as a reaction to a validated request received by the device controller (Figure 12).

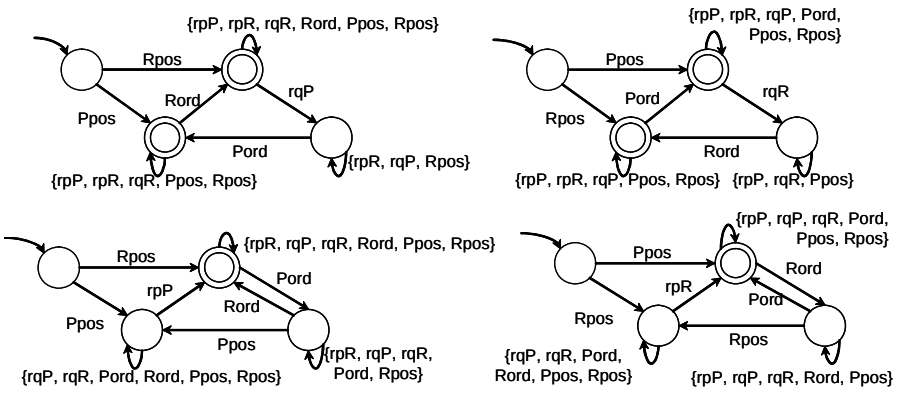


Figure 12: Specifications used for the filter of a cylinder and its bi-stable valve

The complete specification of the cylinder controller results from the synchronous product of these four automata (21 states, 68 transitions). Same approach is used for the specification of the vacuum system (8 states, 25 transitions).

4.1.3. Synthesis of field devices control

From the behavioural models of these devices and their specification, TCT tool is used to generate the largest (or 'supremal') controllable language that defines the most permissive supervisor (Figure 13) as usually done in a SCT synthesis framework.

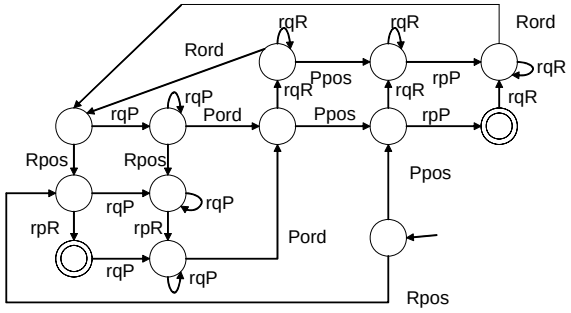


Figure 13: Supreme controllable of the air cylinder with 5/2 control valve

4.1.4. Implementation of the synthesized supervisors

Translation and coding rules presented in section 3.3 are then applied to implement three controllers associated with the two air cylinders and the vacuum generator. Algebraic equations are translated into Ladder networks (example Figure 14) that are implemented into three Functional Blocks of the S7 Siemens PLC.

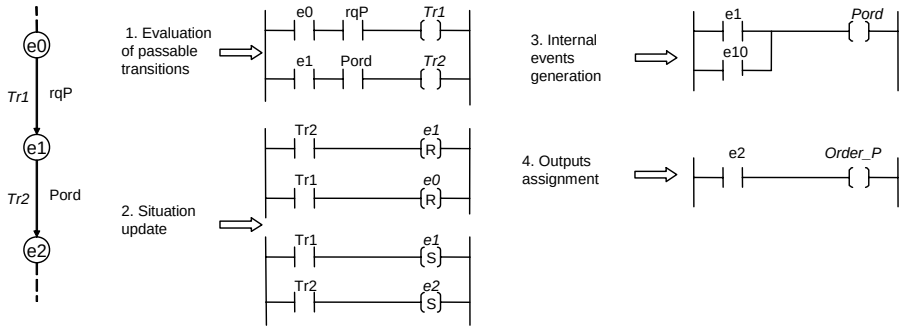


Figure 14: Example of translation into Ladder Diagram

Each controller has been independently tested on the field devices. The observed behaviour is in conformance with the expected one.

4.2. Coordination module Synthesis

Previous step aimed to synthesise generic controllers usable for a given class of field devices. Next step has to deal with the synthesis of a coordinating supervisor that schedules the requests sent to the controllers of the air cylinders and vacuum cups in such a way that a 'pick and place' move is realized. As said in section 3.2, synthesis of this coordinating supervisor is based on a process model that is automatically built from the local supervisors of the field devices. Figure 15 shows the partial process model built from the supervisor associated to the air cylinder that ensures the horizontal move. This partial process (noted *PPM1*) model results from:

– projection of the local supervisors by preserving the only events that are interacting with the superior level, and controllability status modification (*requests* are now seen as controllable while *reports* are now seen as uncontrollable)

– instantiation of the generic supervisor to each of the field devices; it is done by renaming the projected events (for example by adding an 'h' for the horizontal cylinder of Figure 15) to create a specific alphabet for each instances.

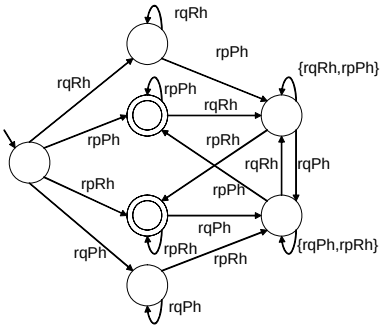


Figure 15: Supervisor projection for a cylinder and its 5/2 bi-stable valve

Same procedure is applied to build the other partial process (noted *PPM2* for vertical air cylinder and *PPM3* for vacuum system). The global process model (noted *GPM*) results from the synchronous product of *PPM1*, *PPM2* and *PPM3* using the SYNC procedure provided by TCT.

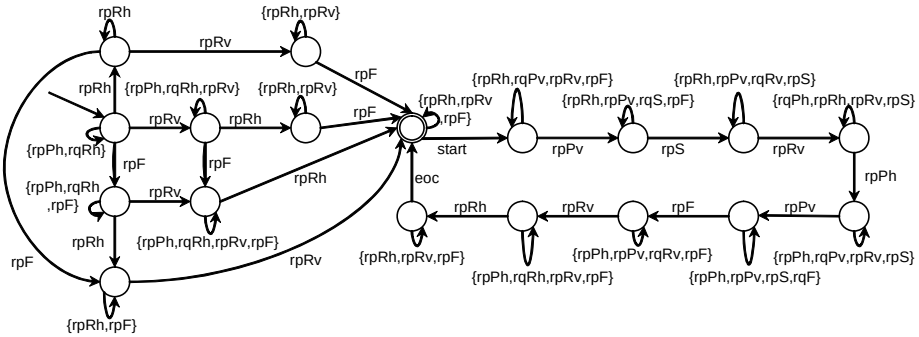


Figure 16: Coordination supervisory specification

The specification model (Figure 16) describes coordinating rules of the three field devices in order to realize a ‘pick an place’ cycle. In the corresponding automaton, two main parts can be identified: initialization steps toward the initial position, ‘pick and place’ cycle.

Synthesis of the coordinating supervisor is generated from this specification model and the above global process model *GPM*. It involves 38 states and 106 transitions: 18 states are devoted to the control of ‘pick and place’ cycle and 20 states refer to initialization. As previously done, the coordinating supervisor is translated and coded into Ladder Diagrams. They are implemented within a Function Block (FB) that calls the FB of section 4.1.4 (control of the field devices behaviour).

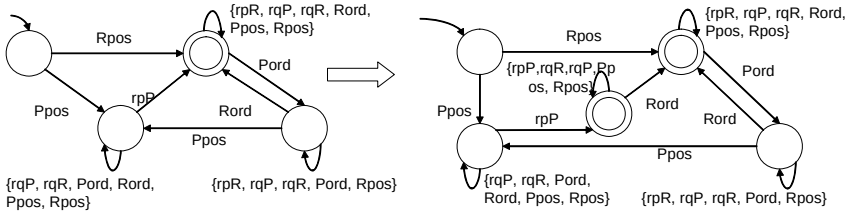
The whole program, including the different FB we have developed using our synthesis approach, has been successfully tested on the manipulator of our assembly cell. However, limitations clearly appeared and are discussed in the next section.

5. Discussion

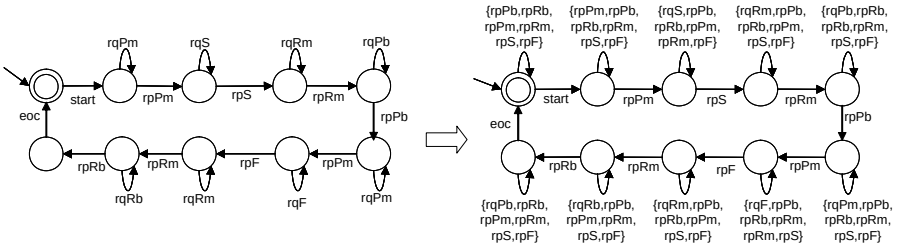
In the previous section, feasibility of our approach has been showed using a realistic case study. However, some modelling and implementation limits have still been encountered. In order to evaluate their impact on the synthesis result itself, several test cases are proposed on this example.

5.1. Modelling impact on the synthesis result

When establishing the models of process and specifications, it clearly appears that several solutions, which could more or less differ from each other, are enabled to capture the process behaviour and the user requirements. Among the various possible models, test cases are proposed in order to reflect two problem classes: identification of the marked states and identification of the significant states.



a) Device control specification: addition of a state (report persistence)



b) Coordination module specification: degree of permissiveness in the cycle

Figure 17: Various examples of test sets to the specifications models

Models that are involved in those test cases are the following:

– two different specifications (noted F1 and F2 in tables 2 and 3) are proposed for the synthesis of the cylinder controller; they differ by the signification of the involved states and the persistence of the *reports* (Figure 17a),

– four specifications (noted C1, C2, C3 and C4 in table 3) are proposed for the synthesis of the coordinating module; they mainly differ by their level of permissiveness and by their initialization (Figure 17b).

– for each of these specifications models, two different state markings have been evaluated.

5.1.1. Identification of the marked states

Marked states are defined in (Ramadge and Wonham 1987) as end of sequences belonging to the marked language L_m: « *$L_m(G) \subset L(G)$ is a distinguished subset of these sequences that may be « marked », or recorded, perhaps representing completed « tasks » (or sequences of tasks) carried out by the physical process that G is intended to model* ». This definition can be differently interpreted by every modeller with regards to its expectations about process behaviour.

For the model of the field devices, the problem is quite simple. Indeed, stable states of these elements can be clearly identified. For example, marked states of the air cylinder model correspond to a cylinder end of move, that is to say ‘pushed’ or ‘retracted’ states (Figure 11).

On the other hand, identification of the states that has to be marked is more difficult when modelling goal specifications. For example, which states should be marked in the specification of the ‘pick and place’ cycle: are they limited to the only state that characterise the initial or idle position of the manipulator, or are they

extended to the whole states that characterise the end of a manipulator move (horizontal and/or vertical one)?

As far as the synthesis algorithms (and more particularly the TCT procedures) depends on which are the marked states, the generated supervisors are far different from each other when a change, even if small, is introduced in the marks of the specification automata. The first two lines of the Table 2 present two different markings for the specifications of air cylinder supervisors:

– if all the states are marked (note *all marked*), the size of the generated supervisors is not realistic with regards to the given problem; however, even if most of the involved states have no meaning, the controller, that can be implemented from this supervisor, successfully works; main benefit is that finding the marked states is a systematic procedure,

– if the marked states correspond to end of a cylinder move (noted *normal*), the generated supervisors are more realistic and better correspond to the expected controller; they also require more intuitive interpretation from the modeller.

In table 2, notation [8,20,4] means that the automaton has 8 states, 20 transitions and 4 marked states. Note that the differences between devices controllers size, according to the different marks, become blurred when a projection is applied to give rise to the process model of a supervisory higher level (two last lines of the Table 2).

Devices control	Specification F1		Specification F2	
	Normal	*All marked*	*Normal*	*All marked*
bi-stable device	[13,26,2]	[33,80,33]	[13,30,2]	[41,102,41]
monostable device	[13,31,1]	[13,31,13]	[13,35,1]	[13,35,13]
Projection bi-stable	[9,24,2]	[11,32,11]	[7,20,2]	[9,28,9]
Projection monostable	[8,20,1]	[8,20,8]	[6,16,1]	[6,16,6]

Table 2: Influence of specifications marking on the result of synthesis;

5.1.2. Identification of significant states

The other problem is related to the various ways to write specification models. Let's take the example of the air cylinder specification. According to the 'behaviour filter' concept, it has to filter the actions and observations to/from the field device according to its internal current state. These filtering constraints can easily be written in natural language as predicates (Figure 18). Formalisation of these properties as algebraic equations is reachable and Roussel *et al.* (2004) showed that a synthesis procedure is applicable from them. On the other hand, modelling these properties as finite state automata is a more delicate task: how to capture the significant and marked states, how to define the required permissiveness given by self-loops?

P1. A 'pushing request' assigned to the 'behaviour filter' generate a 'pushing order' to be applied on the cylinder control valve if and only if the cylinder is not already in a 'pushed state' and no 'pushing request' is active.

P2. A 'pushed state' is reported if and only if the evolution of the position detector from 0 to 1 follows an exit order emitted beforehand.

…

Figure 18: Examples of predicates for the specifications

Considering the various possible answers to these questions, several models can be written to capture the specification of the expected behaviour. Test cases are applied to evaluate the sensitiveness of the synthesis procedure with regards to the specification panel (Figure 12 and Figure 16).

Considering device control, F1 specification differs from the F2 specification by the number of significant states. Indeed generation of a pushed or retracted report is considered as a fugitive event in one case and as a persistent state in the other. The influence of these different versions on the synthesis result (see Table 2) seems weak. In the case of '*normal'* marked states, the supervisor involves the same number of states (13) and few additional transitions (26 for F1 bi-stable instead of 30 for F2 bi-stable). These differences are justified as far as the additional transitions are self-loops that enable the persistent report. On the other hand, these different versions of device control specifications lead to very different coordinating supervisors as illustrated in Table 3. For example, specifications F1 and F2 lead to very different coordinating supervisors ([25,63,2] and [41,184,2]) for a same specification C2.

Among the variations introduced by C1, C2, C3 and C4, the degree of permissiveness seems to be the element that generates the major differences on the synthesis results (see Table 3). Permissiveness is characterised by the number of events that are enabled in the automata self-loops.

Self-loops of the C1 and C3 specifications involve a single controllable event (see first automaton on Figure 17b) that corresponds to the action that should be performed in a given state. C1 and C3 are then said the most permissive ones in the sense that all unexpected events are disabled in the self-loop. These specifications capture the accurate behaviour of the manipulator but do not allow the synthesis of a supervisor (see Table 3). Indeed, only a single uncontrollable event – the expected one – is enabled for exiting a state; that means that all states are considered as forbidden by the synthesis procedure and are consequently eliminated.

The opposite way of modelling (cf. the second automaton on Figure 17b) consists in enabling all the uncontrollable events in the different self loops. It allows the synthesis of a supervisor but the relevance of such a specification can be discussed. Indeed, for a given state of the specification, occurrence of any uncontrollable events is considered as normal behaviour of the process. This requires a strong hypothesis, which is not realistic, stating that any detectors failures can be avoided.

Defining an alternative solution (C2 and C4 specifications, see Figure 16), whose permissiveness is relevant for synthesis but nevertheless realistic, remains a uncertain task.

Coordination	Device specification F1		Device specification F2	
	Normal	*All marked*	*Normal*	*All marked*

Spec C1 : non permissive cycle	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]
Spec C2 : permissive cycle	[25,63,2]	[29,77,6]	[41,184,2]	[41,184,41]
Spec C3 : non permissive cycle + init	[0,0,0]	[0,0,0]	[0,0,0]	[0,0,0]
Spec C4 : permissive cycle + init	[38,106,1]	[38,106,8]	[38,164,1]	[38,164,38]

Table 3: Influence of the specifications structure on the synthesis result

At last, the results obtained using the C4 specification are homogeneous: only few differences concern the number of transitions. It is due to the self loops involving persistent *reports* that distinguish the F1 and F2 specifications.

More generally speaking, we must admit that our pragmatic way of modelling has been driven by the implicit knowledge we had about what the resulting supervisors have to look like. Indeed, specifications have been written again, through iterative attempts, until the synthesised supervisor, and consequently its controller, looks like the ones we have already implemented. In the same way than the use of formal abstract methods such as the B method allows to progressively eliciting the good invariants of a given problem, synthesis techniques may provide a help for defining the good models of specification.

5.2. Impact of supervisors size on implementation

If we look at the behaviour control of air cylinders, the simplest synthesised supervisor involves thirteen states and twenty-five transitions. Vogrig *et al.* (1987), have proposed an equivalent controller for the same device that requires only four states and six transitions of a Grafcet model. The increase of the size between our supervisor and this Grafcet model is still preserved when implementing its associated controller. Indeed, this controller is coded within fifty-five Ladder networks while only twelve Ladder networks are needed to code the Grafcet model of the controller.

In the same way, the control of the whole manipulator – three local controllers for the field devices and one for the coordination module – is described through an hundred of states and transitions and coded by more than four hundred Ladder networks. When comparing the scale of our case study to complex industrial systems, the size of the synthesised program can still be said as too big.

6. Conclusion & Future Work

This paper presents a modular approach for control synthesis within the SCT framework. It relies on the use of automation object-oriented methods that introduce a bottom-up hierarchical reasoning that gradually aggregates the control and process behaviours starting from the most technological levels. It mainly aims at providing methodological help for the preliminary modelling works as well as some coding rules for the implementation on industrial control architecture compliant with the IEC 61131-3 standard.

In this way, major benefits are the systematization of the modeller work and the modularity of the process and goal models. Indeed, process (or SCT *generator*) modelling takes a systematic character as far as the models are built either by instantiation from a library of generic field devices, either by projection and synchronous product of lower level supervisors. Specification modelling is simpler

because properties to be satisfied are distributed among the various levels of control. Projection mechanism is very useful because it reduces the number of handled states by keeping, at every step of the iteration, only a sub-part of the processed alphabet. At last, implementation is facilitated by the modular structure of our supervisors and controllers which is close to the architecture promoted by most of the development tools of the current Programmable Logic Controllers.

Application of the proposed approach using the case study available at the AIP-PRIMECA Lorraine has demonstrated its feasibility. Nevertheless, the writing of the specifications stays a difficult task having a real impact on the result of the synthesis procedures. Indeed, last section of the paper has shown that several accurate formalisations of the same specification could lead to very different synthesis results.

In the same way than the '*behaviour filter*' concept promotes tightly coupled control and monitoring systems, further work should take advantages from taking into account, within the control synthesis, the description of the normal behaviour of a system as well as its incidental one. Indeed, control synthesis including monitoring aspects should lead to clear distinction between what refers to the normal behaviour and what refers to a failure occurrence. This notably helps in reducing the problem linked to the definition of the permissiveness degree of a given specification.

At last, this approach is currently being extended to product-driven manufacturing systems. In this kind of modern architecture, the product is supposed to have an active role within the production scheduling. Synthesis techniques will then be used to automatically generate the different product possible trajectories (*supervisor*) among the available machines (*generator*) in such a way that the performed operations are compliant with the product definition (*specification*). The controllers will be embedded in the product information processing capabilities and coupled with the machine controllers that are developed according to the modular control synthesis presented in this paper.

References

BALEMI S., HOFFMANN G.J., GYUGYI P., WONG-TOI H., FRANKLIN G. F., 1993, Supervisory control of a Rapid Thermal Multiprocessor. *IEEE Transactions on Automatic Control*, 38, 7.

BRANDIN B.A., 1996, The real-time supervisory control of an experimental manufacturing cell. *IEEE Transactions of Robotics and Automation*, 12, 1, pp. 1-14.

BRANDIN B.A., MALIK R., DIETRICH P., 2000, Incremental system verification and synthesis of minimally restrictive behaviours. American Control Conference, ACC'00, Chicago, USA, pp. 4056-4061.

CHAFIK S., NIEL E., 2000, Hierarchical-decentralized solutions of supervisory control. 3th International Symposium on Mathematical Modelling, MATHMOD, Vienna, Austria.

CHARBONNIER F., ALLA H., DAVID R., 1999, The supervised control of discrete event systems. *IEEE Transactions on Control Systems Technology*, 7, 2.

COMBACAU M., COURVOISIER M., 1990, A hierarchical and modular structure for F.M.S. control and monitoring. 1st IEEE International conference on A.I., Simulation and Planning in High Autonomy systems, Tucson, USA, pp. 80-88.

ELKHATTABI S., CORBEEL D., GENTINA J.C., 1992, Integration of dependability in the conception of FMS. 7th IFAC/INCOM symposium, Toronto, Canada, pp. 169-174.

FABIAN M., HELLGREN A., 1998, PLC-based Implementation of Supervisory Control for Discrete Event Systems. 37th IEEE Conference on Decision and Control, Tampa, Florida.

FELIOT C., STAROSWIECKI M., 1998, A syntactic approach for functional modelling of physical systems. 9th Inter. Workshop on Principles of Diagnosis, Cape Cod, USA, pp. 174-181.

FEN L., WONHAM W. M., 1990, Decentralized control and coordination of discrete-event systems with partial observation, *IEEE Transactions on Automatic Control*, 35, 12.

FUSAOKA A., SEKI H., TAKAHASHI K., 1983, A description and reasoning of plant controllers in temporal logic. 8th Inter. Conf. on Artificial Intelligence, pp. 405-408.

GOHARI P., WONHAM, W.M., 1998, A linguistic framework for controlled hierarchical DES. Proceedings of the 4th IEEE Workshop On Discrete Event Systems, WODES 98, Calgary.

HIRAISHI K., 2001, Synthesis of supervisors using learning algorithm of regular languages. *Discrete Event Dynamic Systems: Theory and Applications*, 11, pp. 211-234.

International Electrotechnical Commission, 1993: *IEC 61131-3, Programmable controllers – Part 3: programming languages*, first edition, 2000: *IEC/PAS 61499-1, Function blocs for industrial measurement and control systems - Part 1: Architecture*.

JIANG S., KUMAR R., 2000, Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man and Cybernetics, part B*, 30, 5, pp. 653-660.

KUMAR R., GARG V., MARCUS S.L., 1991, On controllability and normality of discrete event dynamical systems. S*ystems & Control Letters*, 17, pp. 157-168.

LHOSTE P., MOREL G., 1996, From discrete event behavioural modelling to intelligent actuation and measurement modelling. ASI annual Conf. of the ICIMS-NOE, Toulouse.

MARCHAND H., BOURNAI P., LE BORGNE M., LE GUERNIC P., 2000, Synthesis of discrete-event controllers based on the Signal environment. *Discrete Event Dynamic Systems: Theory and Applications*, 10, pp. 325-346.

MARIKAR M.T., ROTSEIN G.E., MACCHIETTO S., 1998, An integrated environment for the design of procedural controllers. Volume II of 9th IFAC/INCOM symposium, Nancy.

MOREL G., PÉTIN J.F., LAMBOLEY P., 2001, Formal specification for manufacturing systems automation. 10th IFAC/INCOM symposium, Vienna, Austria.

NIEL E., PIETRAC L., REGIMBAL L., 2001, Advantages and drawbacks of the logic programm synthesis using supervisory control theory.10th IFAC/INCOM'01 symposium, Vienna, Austria.

PÉTIN J.F., IUNG B., MOREL G., 1998, Distributed intelligent actuation and measurement (IAM) system within an integrated shop-floor organisation. *Computers in Industry*, 37.

DE QUEIROZ M.H., CURY J.E.R., 2002, Synthesis and implementation of local modular supervisory control for a manufacturing cell. Proceedings of the 6th International Workshop on Discrete Event Systems, 2-4 October, Zaragoza, Spain, pp. 377-382.

RAMADGE P.J., WONHAM W.M., 1987, Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, Vol. 25, n° 1.

ROUSSEL J.M., FAURE J.M., LESAGE J.J., MEDINA A., 2004, An algebraic approach for dependable logic control systems design, *in this number/volume of IJPR*.

SANCHEZ A., MACCHIETO S., 1995, Design of procedural controllers for chemical processes, *Computers Chem. Engng*. 19, pp. S381-S386.

TILLER M., 2001, *Introduction to Physical Modelling With Modelica* (Kluwer Academic Publishers), ISBN 0792373677.

VOGRIG R., BARACOS P., LHOSTE P., MOREL G., SALZEMANN B., 1987, Flexible manufacturing shop. *Manufacturing Systems*, 16, 3.

WONHAM W.M., RAMADGE P.J., 1987, On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, vol.25, No.3.

XIE X., 1994, Some results on the integration of manufacturing systems using Petri nets. IEEE Conference on Systems, Man and Cybernetics, San Antonio, Texas.

YOO T.-S., LAFORTUNE S., 2002, A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 12.

ZAYTOON J., CARRE-MENETRIER V., 2001, Synthesis of a correct control implementation for manufacturing systems. *International Journal of Production Research*, 39, pp. 329-345.

ZHONG H., WONHAM W.M., 1990, On the consistency of hierarchical supervision in discrete event systems, *IEEE Transactions on Automatic Control*, 35, 10.