Reactive control system design using the Supervisory Control Theory: evaluation of possibilities and limits

Matteo Cantarelli, Jean-Marc Roussel

▶ To cite this version:

Matteo Cantarelli, Jean-Marc Roussel. Reactive control system design using the Supervisory Control Theory: evaluation of possibilities and limits. 9th International Workshop On Discrete Event Systems (WODES'08), May 2008, Göteborg, Sweden. pp. 200-205. hal-00353636

HAL Id: hal-00353636

https://hal.science/hal-00353636

Submitted on 15 Jan 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reactive control system design using the Supervisory Control Theory: evaluation of possibilities and limits

Matteo Cantarelli and Jean-Marc Roussel

Abstract— **This paper presents a method to obtain programs to control an industrial equipment using the Supervisory Control Theory (SCT). The presented approach is based on high detailed Process models. A model of the interactions between the Control Unit and the Process is also included in the Plant model to facilitate the extraction of the Controller behavior from the Supervisor. An approach to obtain from the Supervisor a Controller straightforwardly implementable and then to automatically generate the Control Unit program is provided. The method presented in this paper has been tested on several case studies (from Specifications to test with a real Process). A dimensional summary of two of them is presented in the last section.**

I. Introduction

This paper deals with the design of programs for a Control Unit using the Supervisory Control Theory (SCT). The SCT is an approach based on formal languages that has influenced many researches on discrete event systems since the publication of the seminal works [1]. Different theoretical results appeared since then and a basis for a technical transfer to the industry was also expected. Unfortunately though, today there is still a big gap between the theory and its application in the industry in spite of what the discrete event community would had expected fifteen years ago. In [2] the authors show how at the present the majority of industrial control programs for manufacturing system are obtained manually, relying on the reuse of hand written logic from previous projects, due to the similarity of the equipments to control and their behavioral requirements. In spite of the currently competitiveness, this strategy is not going to be compatible with the incoming safety requirements for the industry installations. One of the main limits of this practice is that the quality and safety of the resulting code relies totally on the expertise of developers of programs for Programmable Logic Controller (PLC). As the SCT is a mathematical framework capable of automatically calculate the maximal permissive behavior of a system guaranteeing the respect of the specifications, it seems ideal to be used to obtain programs. The possible behavior of a system is given by a set of Deterministic Finite Automata (DFA) named *Plant model*, while the allowed behavior is given by a set of DFA named *Specifications*. From these two entries the SCT calculates the *Supervisor*, the DFA which represents the maximal permissive behavior for the system. For further details about the SCT the reader is referred to [3]. As in the SCT the Supervisor is obtained as result of a mathematic computation its quality stands totally upon the quality of its inputs. Also, as the behavior to implement in the Control Unit is a subset of the behavior represented by the Supervisor it is needed a way to extract it. In this work a systematic approach to exploit the Supervisor and to obtain the PLC code is presented. This approach is based on highly detailed input models proposed to take into account the technology used in the Process and to obtain as consequence a suitable Supervisor.

The approach we propose is presented in section II. Section III deals with the Plant model and section IV deals with the Specifications. The two next sections show how to obtain the PLC code from the Supervisor. In section VII the results of the achieved experiments are presented (a full detailed report for two of them is available at the website http://www.lurpa.ens-cachan.fr/isa/asc/sct/). The final section deals with the conclusion and the perspectives of this work.

II. Proposed Approach

The proposed method is described in Figure 1. The inputs of the SCT synthesis are the *Plant model* and the *Specifications*, the output is the *Supervisor*. From the *Supervisor* is extracted a *Controller* suitable to obtain the *PLC Code*.

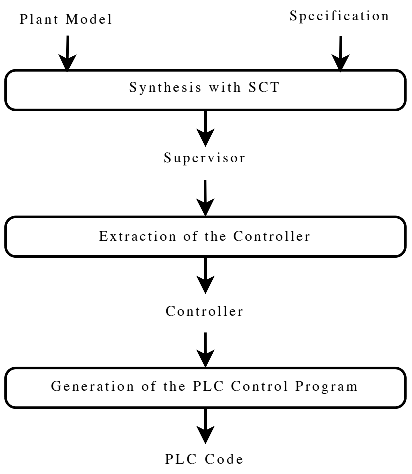

Fig. 1. Proposed approach based on SCT

In approaches found in literature the step for the extraction of the behavior to implement is mixed with the code generation. In this work, the two steps are kept separate to clearly face the problems of the extraction stage. The

M. Cantarelli is Electronic Engineer at DIEE, University of Cagliari, Piazza D'Armi, 09123, Cagliari, Italy. This work was undertaken during his MSc thesis. He is now employed at Pilz Ireland as Software Engineer. m.cantarelli@pilz.ie

J.M. Roussel is with LURPA, ENS de Cachan, UniverSud 61 Av. du président Wilson, F-94235 Cachan Cedex, France. jean-marc.roussel@lurpa.ens-cachan.fr

difficulties are due to both the different interpretations and objectives between a Supervisor and a Controller. The control of a system from a Supervisor is done by the interdiction of the controllable events emitted by the Plant model [1]. The Plant model is a "wide" event generator and the Supervisor, according to the Specifications, disables the controllable events that can put the system in an unsafe or undesired state. The problem arises when controlling a real system as it is often required to force some events to occur and not only to enable and disable some of them. To do this a proposal [4], used in different works, supposes that the Plant model only produces the uncontrollable events and that the controllable events are produced by the Supervisor (Figure 2). The proposed approach achieves instead to stand upon the

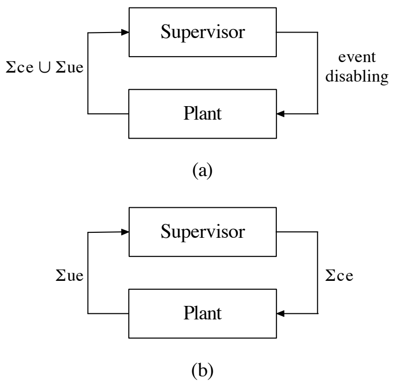

Fig. 2. (a) Original interpretation (b) Input/Output interpretation (proposed by Balemi)

original interpretation [1], permitting however to determine which are the controllable events that need to occur to fulfill the Specifications. This is made possible thanks to the model of the Control Unit introduced into the Plant Model. The second difference is due to the different required objectives between a Supervisor and a Controller. From the Supervisor

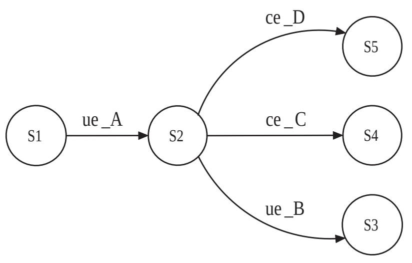

Fig. 3. A deterministic behavior for a Supervisor but an indeterministic behavior for a Controller

Figure 3 point of view, once that uncontrollable event `ue_a` has occurred, the only authorized behavior is the occurrence of `ue_B` or the occurrence of one of the Controllable events `ce_C` or `ce_D`. From the control point of view, once that uncontrollable event `ue_a` has occurred, in the `S2` state three different behaviors are possible:

- to wait the occurrence of uncontrollable event `ue_B`,
- to produce controllable event `ce_C`,
- to produce controllable event `ce_D`.

This behavior is non deterministic for a Controller. Does the system have to wait for the next uncontrollable event (`ue_B`) before producing the controllable one or this last is produced as first? And which one of the two different controllable events (`ce_C`, `ce_D`) has to be produced? While the Supervisor represents what the system is authorized to do, the Controller represents what the system has to do. This problem has been faced in different ways in the literature (a survey is present in [5]). For the choice between uncontrollable and controllable events a solution often proposed is to give the priority to the uncontrollable events [6][7][8], while in [9] the Supervisor is rejected if it is does not correspond to a Controller. The SCT is purposely reinterpreted in [10] to let the Supervisor acting like an events trigger. For the choice between controllable events [6] proposes the association of different priority levels or the manual intervention of the designer. In this work we propose to extract the Controller from the Supervisor selecting the path which minimizes the number of times that the PLC changes its outputs. The behavior of the obtained Controller is then expressed with a Mealy machine to minimize the size of the implemented state machine. The proposed form permits to obtain compact PLC code.

Section III deals with the creation of the Plant model. A model of the Control Unit is added to the Plant model to introduce the concept of the PLC scan cycle, this allows to define the behavior of the logic Controller solving part of the concurrence problems shown in figure 3. Section IV presents a way to write the Specifications that need to force some events to occur. Section V shows how to obtain the Controller from the Supervisor, while Section VI shows how this Controller is exploited to obtain the PLC code.

III. Design of the Plant model

Note: to avoid confusion between the term "Plant" used in SCT and the term "Plant" used in industry, in the context of industrial equipment to control we will refer to it as "Process to control", while keeping the term "Plant" for the SCT.

The design of the Plant model is a difficult step and deserves the maximum care. The Plant model needs to be representative of the whole possible behavior of the system, i.e. whatever thing the system is physically capable to do should be also allowed by the model. From a technical point of view, the overall system could be partitioned into two parts: the Process to control and the Control Unit (Figure 4).

The boundary between the two parts is imposed by the technology, and more precisely, by the choice of inputs and outputs of the industrial Controller. These inputs and outputs of the Control Unit are electrical signals (which can be modelled by Booleans) from sensors or for the actuators. As our objective is the synthesis of the program to implement int the Control Unit, we consider as Plant all the other parts of the system. Also, the Plant model we propose is composed of a model of the Process to control and of complementary

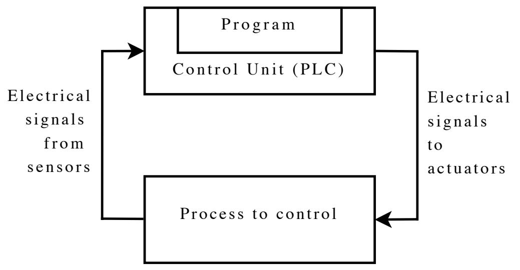
Program
Control Unit (PLC)
Electrical signals from sensors
Electrical signals to actuators
Process to control

Fig. 4. Fig: Technical decomposition of a controlled system

models added to describe how the Process and the Control Unit interact. These complementary models are generic as they describe the behavior of the inputs and outputs of Control Unit and its own behavior. This strategy is also convenient as it helps to precisely define the right level of abstraction needed to produce the system control laws. The separation of each basic Process model is essential to minimize the errors and to promote a reuse orientation. In the examples presented in the literature the level of abstraction is often too high to allow a correct definition of the control code. We now present all the different models introduced by this approach in the Plant Model.

A. The inputs and outputs models of the Control Unit

The first models which compose the Plant model describe the behavior of the inputs and outputs of the Control Unit. These models map the variation of the electric signals in the Control Unit to events. Two events are associated to each electric signal [11]: one for the change from value 0 to value 1 (rising edge) and one for the opposite change (falling edge). These events will be uncontrollable for the inputs (the values are imposed by the Process) and controllable for the outputs (the values are imposed by the Control Unit). The proposed models precise that the two events associated to each electrical signal alternatively occur. The models for inputs and outputs (Figure 5) are different in order to take in account the problem of the initial value. As the initial value of input is not always the same and to have generic models, the proposed one for inputs has three states. For the outputs a model with two states is sufficient as the initial state can be considered, without lack of generality, as zero. These models are generic for all the systems and can be reused across different projects.

B. The Control Unit model

The second model added to the Plant Model describes the behavior of the Control Unit itself. Modeling this behavior explicitly assures no room left for any kind of ambiguity or handmade choice during the later determination of the Controller. The interactions between the Process and the Control Unit represent the cornerstone of this method. The synchronization of the Control Unit with the Process produces a separation between variations of inputs and outputs belonging to different PLC cycles. This aspect is described in the Control Unit model. In this paper two models are proposed for the Control Unit. In both of them the Control

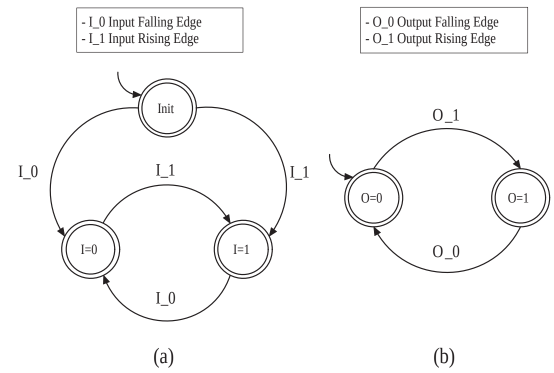

Fig. 5. (a) Input model (b) Output Model

Unit is considered as a reactive system i.e. it can distinguish input changes and can always calculate the consequent output changes. This is possible if the scan cycle time of the Control Unit is short enough in comparison to the time constants of the Process. This is a usual and appropriate assumption.

1) Single Input Multiple Output (SIMO): The SIMO model describes an infinitevely reactive Control Unit which can distinct all inputs and can build an appropriate reaction with, if necessary, several changes of outputs. The proposed model (Figure 6) has two states: `Wait` and `Exe`. When an uncontrollable event occurs (evolution from state Wait to state Exe), it is possible to write several outputs or to go back to state Wait. The `Exe` state represents the execution of the user program, during which the Control Unit changes its outputs. In the `Wait` state the outputs are kept fix. The controllable event `End` represents the end of the PLC cycle. After the synchronization each cycle is separated by the `End` event. This controllable event allows to write Specifications that requires to force a particular controllable event to occur (see Section IV).

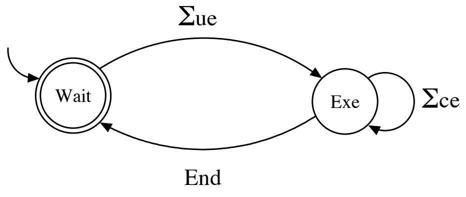

Fig. 6. SIMO automaton model

2) Multiple Input Multiple Output (MIMO): As in real system several inputs may change during the same PLC scan cycle a MIMO model is proposed too. The automaton (Figure 7) has two states: `Wait` and `Exe`. The uncontrollable event `Read` has been added as to indicate the conclusion of the reading operations. As for the SIMO model the `End` event will separate different PLC cycle, while the `Read` event will separate, within a cycle, the reading from the execution steps.

3) Impact of SIMO and MIMO models: The choice between SIMO and MIMO models, in agreement with the system requirements, is up to the designer. The experimentations have shown that the chosen model affects the Process

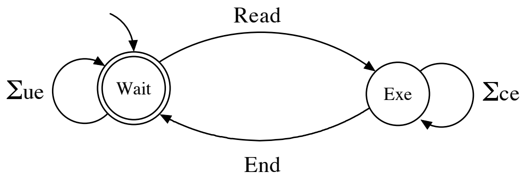

Fig. 7. MIMO automaton model

models to take in account and the obtained Supervisor. We have noted that realistic Process models are more simpler to obtain for the SIMO model. If a MIMO model is needed to precisely model the Control Unit, the Process models need to be adequately detailed; in this paper we only consider the behavior of the Control Unit as described by the SIMO model.

4) Properties resulting from the Control Unit model: Because the Plant Model contains the Control Unit it can benefits from the following characteristics:

- The states can be partitioned in two sets: `Ss1` the states for which the output events are only uncontrollable events, `Ss2` the states for which the output events are only controllable events. Only states in `Ss1` can be marked.
- Using the SIMO model between the occurrence of two uncontrollable events is always present an occurrence of the `End` event.
- All the sequences of output changes (controllable events) is terminated with the `End` event.

These properties are exploited for the design of the Specifications and in the extraction of the Controller from the Supervisor.

C. The Process models

Once the inputs and the outputs of the system are fixed, the automata models to define the behavior of the Process are added to the Plant Model. The main difficulty is to propose models with the right level of abstraction. As our objective is to obtain control programs for a PLC, we need very detailed models of the Process with the following requierements:

- these models must not describe only the expected behavior: all the possible behaviors, dangerous and not, have to be modeled. Necessary complementary Specifications need to be added to constraints this behaviors to what expected. If this aspect is not take into account, the obtained control program would not include appropriate reactions to avoid the dangerous situation in the real Process.
- these models must be coherent with the technology of the Process components. For instance, the code to control a pneumatic cylinder includes some parts to take into account the type of cylinder (single or double acting), if these details are not take into account when creating the Process model, the final code could be wrong. To avoid these errors, the Process models could be obtained from the composition of each elementary component as proposed in [12].

By experience, we have noted that the design of the Process model is a complicated step and deserves the maximum care. As the Process model needs to be representative of the whole possible behavior of the real Process, these models are very difficult to create. To be coherent, these models need to be designed by experts with mechanical and electrical competences (to analyse the Process) an also competence in SCT to be able to express this behavior.

IV. Design of the Specifications

The writing of the Specifications has always represented a common problem in the community, whichever modeling class is used. The main issue is to translate something usually expressed in natural language into a formal language used as control model.

The first issue is the choice of the marked states of the Specifications. The real impact of this choice is clearly presented in [6]. In our approach, we propose to mark the states in which the system has accomplished the desired task. As the SCT disables the events that do not lead the model to a marked state, all the routes that do not allow the system to complete its task are automatically suppressed. The same behavior is commonly obtained by not marking some states in the Plant Model. In our approach this is avoided to clearly separate the Specifications aspects from the Process ones.

The second issue is to force a controllable event to occur. This need was clearly presented in [8][10] and was the object of several scientific works. In our method, we propose a way to force the Control Unit to react when an uncontrollable event occurs (for instance, to stop a conveyor at the end of its movement) without extending the SCT. To obtain this, thanks to the Control Unit model introduced, it is enough to forbid the `End` event to occur as long as the controllable event to force is not occurred. This shows from a different perspective the importance of the Control Unit model introduced in the Plant model.

In Figure 8 an example of a Specification shows the ideas presented. The objective of this Specification is to force the `Output` event if the `Input` event is read. As the `End` event is forbidden from the state `S2` the only way to end the current PLC cycle is to first produce the `Output` event.

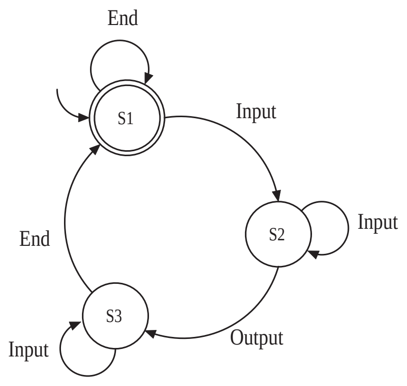

Fig. 8. A way to force a controllable event to occur.

V. From Supervisor to Controller

The output of the SCT synthesis is a Supervisor which can be minimized by language equivalence to reduce its number of states. This Supervisor describes the maximal possible behavior of the system which satisfies the specifications. Due to the presence of the Control Unit model in the Plant Model, all the states in the Supervisor belong either to `Ss1` or `Ss2` (see III-B.4). When in a state belonging to `Ss2` it is available more than one controllable event it is necessary to precise the expected behavior for the controller therefore it must select just one of the available events. The criterion we propose to do this consists in selecting the path which permits to reach the most rapidly the states for which the system has accomplished the desired tasks (theses states are the marked states of the Supervisor). The proposed criterion therefore is to minimize the number of times that the PLC changes its outputs. This is obtained selecting among all the possible paths the one with the smallest number of `End` events. To do this a weight is associated to every state. This weight represents the minimum number of `End` events that need to be traversed to reach another marked state. Once this weight is calculated, to select the controllable event from a state, it is enough to analyze the weight of all the reachable states and select the one with the minimum weight. The other transitions are suppressed from the Supervisor. In case several states present the same weight, we propose to select the controllable event according to a priority level given by the designer. When all the states are processed, the Controller is obtained by retaining only the reachable state from the initial state. Figure 9 shows an example of Controller obtained when using a SIMO Control Unit model.

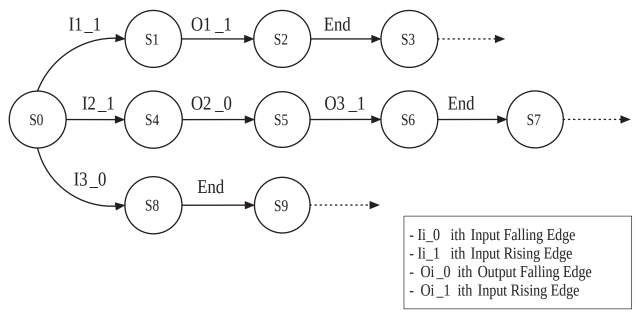
I1_1
S1
O1_1
S2
End
S3
S0
I2_1
S4
O2_0
S5
O3_1
S6
End
S7
I3_0
S8
End
S9
- Ii_0 ith Input Falling Edge
- Ii_1 ith Input Rising Edge
- Oi_0 ith Output Falling Edge
- Oi_1 ith Input Rising Edge

Fig. 9. Example of DFA Controller

The pattern obtained with the SIMO model has sequences of events composed by an uncontrollable event, a list of controllable events and closed by the `End` event. Each sequence corresponds to an execution of the PLC program during a PLC cycle. In the state `S0` the Controller is at the beginning of the PLC cycle. If the event `I1_1` is read then the Controller has to set to one the output 1 producing the event `O1_1`. The event `End` states the end of the current cycle and the next PLC cycle will begin from the state `S3`. This same behavior can be represented in a more compact way by a Mealy machine thanks to the following properties:

- only the final PLC state is important for the code,
- each sequence can be represented by a single transition,
- if several outputs are written, their order is not important as they will be processed during the same PLC cycle.

During the conversion to the Mealy machine the `End` event is suppressed as from now on each cycle is going to be represented by a single transition. The Mealy machine obtained from the DFA Controller in Figure 9 is presented in Figure 10.

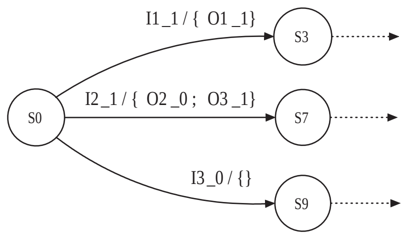

Fig. 10. Corresponding Mealy machine of the Controller presented in Figure 9

VI. From Controller to Code

The last step of our method is the generation of the PLC code from the Mealy machine. This Mealy machine is a generic model which can be coded in several languages. For the case studies we dealt with, we chosen to work with the Structured Text (ST) language defined in the IEC 61131-3 standard [13] as, thanks to its textual nature, it is easier to automatically generate the PLC code to export to the PLC development tool suite. Figure 11 presents the extract of code, corresponding to the Mealy machine given in Figure 10.

```
PROGRAM

IF (STATE=0) THEN
   IF (I1) THEN
      SET O1;
      NEXTSTATE:=3;
   ELSIF (I2) THEN
      RESET O2;
      SET O3;
      NEXTSTATE:=7;
   ELSIF (NOT I3) THEN
      NEXTSTATE:=9;
   ELSE
      NEXTSTATE:=STATE;
   END_IF
END_IF;

IF (STATE=3) THEN
      [...]
END_IF;

IF (STATE=7) THEN
      [...]
END_IF;

IF (STATE=9) THEN
      [...]
END_IF;

      [...]
STATE:=NEXTSTATE;
END_PROGRAM
```

Fig. 11. An extract of the generated PLC code in ST language

It can be noted that only two integer variables are required to code the Mealy machine: `STATE` for the value of the current state at the beginning of the PLC cycle and `NEXTSTATE` for the value of the state at the end of the PLC cycle. This compact notation is possible as the Mealy machine has only one active state. The use of these two variables avoid several changes of state during the same PLC cycle, as it is clearly presented in [14]. The next state of the Mealy machine is determined from the current state of the Mealy machine. At the end of program, the value of variable NEXTSTATE is copied into variable STATE to prepare the next PLC cycle. The generated code is composed of `IF` statements: the extern `IF` selects the current state (variable `STATE`), while the second level select the changed inputs. The test of an input, to determine the right outputs and the next state, is made simple comparing its value with 0 or 1. The rising and falling input edges, represented by the events in the Mealy machine, don't need to be take into account as the information of the previous value of the variable is implicitly embedded in the state. The output changes are produced using the functions `SET` and `RESET`.

VII. Experimental feedbacks

The approach presented in this paper has been tested on several applications to be proven valid. Table I presents a dimensional summary for two of them. The first presented case of study consider the control of an automatic car gate, the second one deals with the control of a pick and place station. The SCT synthesis was made with the Supremica tool developped at Chalmers [15]. The different algorithms used to obtain the Controller and the Mealy machine were coded in Python. The generated ST code has been downloaded into a PLC (TSX Premium from Schneider Electric) and successfully used to control the processes. The reader can consult the website previously given to obtain all details about these case studies: Process models, Control Unit Model, Inputs/Outputs models, Specifications, Supervisor, Controller, Mealy Machine and PLC code. Even if the state explosion is an important factor (the size of the resulting synchronized automaton of the Plant model and Specifications (Table I) is quite big), the proposed approach allows to obtain a Mealy machine Controller of reduced size.

TABLE I
Treated use cases

	Car gate	**Pick&Place Station**
Plant Model	11 Automata:	17 Automata:
	4 for Inputs(*)	6 for Inputs(*)
	2 Outputs(*)	3 Outputs(*)
	1 Control Unit(*)	1 Control Unit(*)
	4 Process models	7 Process models
	(*)Generic models	
Resulting automaton:	481 States	62721 States
	1330 Transitions	237890 Transitions
Specifications	11 Automata	9 Automata
Resulting automaton:	276 States	416 States
	1215 Transitions	2616 Transitions
Supervisor	368 States	173 States
	646 Transitions	251 Transitions
After minimization:	110 States	108 States
	194 Transitions	160 Transitions
Controller	45 States	45 States
	70 Transitions	73 Transitions
Mealy Machine:	15 States	30 States
	40 Transitions	58 Transitions

VIII. Conclusions

The approach proposed in this paper allows to use the SCT to obtain control programs for industrial equipements with high detailed Plant model. The Plant model we propose is composed of a model of the Control Unit, of models of its inputs and outputs and of models of the Process to control. We shown how the Control Unit model can be used to facilitate the design of Specifications and how to exploit it to extract the Controller from the Supervisor. These added models are generic and can be reused, also, as their role has been clearly defined in this paper, other models could now be proposed and future works could research their impact on the obtained Supervisor. The case studies we dealt with show how the size of the Supervisor increases rapidely when the Plant model is very detailed. As this level of description is mandatory to obtain programs able to control real processes, to use the SCT in more complex systems, e.g. large scale production lines, a decomposition in sub-parts would be needed. The SCT could then be used again to synchronize these parts to work together.

References

[1] P. Ramadge and W. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[2] M. Lucas and D. Tilbury, "A study of current logic design practices in the automotive manufacturing industry," *International Journal of Human-Computer Studies*, vol. 59, no. 5, pp. 725–753, 2003.

[3] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[4] S. Balemi, "Control of Discrete Event Systems: Theory and Application," Ph.D. dissertation, PhD thesis, Swiss Federal Institute of Technology Zurich, 1992.

[5] F. Basile and P. Chiacchio, "On the Implementation of Supervised Control of Discrete Event Systems," *Control Systems Technology, IEEE Transactions on*, vol. 15, no. 4, pp. 725–739, 2007.

[6] D. Gouyon, J. Petin, and A. Gouin, "Pragmatic approach for modular control synthesis and implementation," *International Journal of Production Research*, vol. 42, no. 14, pp. 2839–2858, 2004.

[7] P. Malik, "Generating Controllers from Discrete-Event Models," *Proceedings of MOVEP'2002*, 2002.

[8] P. Dietrich, R. Malik, W. Wonham, and B. Brandin, "Implementation considerations in supervisory control," *Synthesis and Control of Discrete Event Systems*, vol. 185201, 2002.

[9] J.-M. Roussel and A. Giua, "Designing dependable logic controllers using the supervisory control theory," *16th IFAC World Congress (Prague, Czech Republic)*, 2005.

[10] S. Balemi, G. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. Franklin, "Supervisory control of a rapid thermal multiprocessor," *Automatic Control, IEEE Transactions on*, vol. 38, no. 7, pp. 1040–1059, 1993.

[11] J. Zaytoon and V. Carre-Menetrier, "Synthesis of control implementation for discrete manufacturing systems," *International Journal of Production Research*, vol. 39, no. 2, pp. 329–345, 2001.

[12] V. C.-M. B. Rohee, B. Riera and J.-M. Roussel, "A methodology to design and check a plant model," *3rd IFAC Workshop on Discrete-Event System Design, DESDes'06*, pp. 246–250, 2006.

[13] I. TC65 and I. WG, "61131-3: Programmable Controllers–Part 3: Programming Languages," *International Electrotechnical Commission IEC Std., Rev*, vol. 2, 2003.

[14] M. Fabian and A. Hellgren, "PLC-based implementation of supervisory control for discrete eventsystems," *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, vol. 3, 1998.

[15] K. Akesson, M. Fabian, H. Flordal, and A. Vahidi, "Supremica: a tool for verification and synthesis of discrete event supervisors," *Proc. of the 11th Mediterranean Conference on Control and Automation*, 2003.