Robotics and Computer-Integrated Manufacturing 26 (2010) 361–369

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Synthesis of product-driven coordination controllers for a class of discrete-event manufacturing systems

A. Sanchez [a,*], E. Aranda-Bricaire [b], F. Jaimes [b], E. Hernandez [b], A. Nava [a]

[a] *Centro de Investigacion y Estudios Avanzados (Cinvestav), Unidad de Ingenieria Avanzada, A.P. 31-438, Zapopan 45091, Jalisco, Mexico*
[b] *Centro de Investigacion y Estudios Avanzados (Cinvestav), Seccion de Mecatronica, Unidad D.F., A.P. 14-740, D.F. 070000, Mexico*

ARTICLE INFO

Article history:
Received 3 October 2006
Received in revised form
11 November 2008
Accepted 23 November 2009

Keywords:
Automated manufacturing systems
Coordination control
Product manufacturing control
Discrete-event systems
Supervisory control

ABSTRACT

A synthesis approach is proposed for discrete-event coordination architectures applied to a class of automated manufacturing systems (AMS) in which a clear separation is established between equipment control activities and product manufacturing procedures. Manufacturing procedures are modeled by regular languages constructed with a class of control commands named *imperative*. Equipment controllers are synthesized as a standard discrete-event *supervisors* dealing only with operational and safety issues of equipment groupings. The control of equipment modules is carried out following the *imperative* control commands sequences. Conditions are established to guarantee that the manufacturing procedure of a given product can be achieved using the synthesized supervisors in a particular AMS. Therefore, equipment controllers are not needed to be modified to consider the manufacturing of different products, whilst the construction of achievable manufacturing procedures becomes an "ad hoc" simple process using a reduced set of procedural blocks. The approach is illustrated with an experimental AMS.

1. Introduction

Manufacturing systems (discrete, batch and continuous) have experienced in recent years a notorious increase of procedural complexity. Added to this factor, market conditions and customer awareness put considerable pressure on rising production and operational requirements related to environmental issues, safety and economy. In this context, discrete-event control is playing an increasingly important role. The academic community has responded to the challenge with the proposal of formal methods, techniques and tools exploring issues on designing, implementing and maintaining larger and more complex discrete-event control devices for manual or automatic operation. Among them, Supervisory Control Theory [17] has received considerable attention as a synthesis theory for a class of controllers. Its application and potential impact in manufacturing is widely documented. Most work concentrates on the synthesis of supervisors dealing with the operational behavior of processing equipment considering different control technologies (e.g. [1,2,14]) in modular-hierarchical architectures (e.g. [10,13,18]). Production rules for specific products are usually included as specifications in the supervisor synthesis procedure. This gives rise to rigid supervisory architectures for manufacturing products according to the established production rules. However, changing production rules is common practice (e.g. for manufacturing new products or introducing different production wheels). In many occasions this may require to update the supervisory controllers accordingly. Therefore, separating control activities related to equipment only from product manufacturing actions may help in designing better discrete-event supervisory architectures. A step in this direction is the synthesis of supervisors establishing feasible manufacturing routes for specific products (named product supervisors) in terms of equipment tasks-status models [5]. These supervisors relate to standard workstation supervisors by common sets of equipment task events. The approach gives rise to hierarchical-modular supervisory architectures in which for the synthesis of upper level product supervisors, projections of lower level workstation equipment behavior specifications are required. Therefore, changing supervisors at lower levels implies recalculating product supervisors. Moreover, the manufacturability of a given product is not guaranteed by the synthesized product supervisor and global controllability was not considered.

This paper explores the synthesis of discrete-event supervisory control architectures applied to a class of automated manufacturing systems (AMS) in which a clear separation between equipment control activities and product manufacturing control is established. Equipment supervisors dealing only with operational and safety issues of equipment groupings are synthesized in an standard fashion while generic product-manufacturing procedures, named recipe procedures [7], are modeled by languages constructed with a class of control commands named *imperative*. Conditions are established to guarantee that these recipe

* Corresponding author. Tel.: +52 3337773600; fax: +52 3337773609.
E-mail address: arturo@gdl.cinvestav.mx (A. Sanchez).

0736-5845/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2009.11.014

procedures are *achievable* (i.e. the product is manufacturable) by standard equipment supervisors restricting the AMS to controllable behavior generated by a *reliable* automaton. Therefore, the construction of recipe procedures becomes an “ad hoc” simple process using a reduced set of building blocks for which *achievability* has been previously verified. Section 2 initiates with a brief summary of the standard Supervisory Control Theory (SCT) introducing the event classification required for this work. The following section discusses the notions of reliable automaton and achievable language as well as sufficient conditions for their existence. These are the theoretical tools employed to build supervisors guaranteeing that a given manufacturing sequence can be realized in the AMS. An algorithm is proposed to verify whether a closed-loop behavior automaton is reliable. Other required calculations can be carried out with existing tools. The approach is illustrated with the design and implementation of a coordination control architecture for an experimental AMS described in Section 4. The industrial standards ISA-88 [7] and ISA-95 [8] are employed to build the equipment and procedural control models for a three-layer modular coordination architecture. The lowest layer deals with equipment control by dully executing *phases* whilst the top layer receives recipe procedures modeled as IDEF3 *flow models* [12] enforcing the *unit procedures* and *phase* sequencing. Supervisors are synthesized for each resulting module (i.e. unit procedures and phases). Section 5 describes the application of the standard supervisor synthesis procedure for phase and unit procedure exemplars. The generated closed-loop automata must be verified as reliable in order to guarantee the achievability of products. IDEF3 flow model steps are then translated, as discussed in Section 6, into automata accepting achievable languages employed to drive the AMS in a short-term schedule mode of operation. The paper closes with Section 7 discussing the advantages and limitations of the approach as well as commenting how the ideas presented in this paper can be used in conjunction with other work to strengthen the synthesis of this class of coordination controllers.

2. Fundamentals

An AMS is modeled as a discrete-event process in a SCT-standard fashion using an automaton $P=(X,\Sigma,\delta_P,x_0,X_m)$ where X, Σ, δ_P, x_0 and X_m denote, respectively, the state set, event set, transition function, initial state and set of marked states of the automaton. The event set is divided into controllable and uncontrollable events such that $\Sigma_u\bigcap\Sigma_c=\emptyset$. In this work, controllable events describing the initiation of a task (i.e. starting an operation) are labeled as *imperative*, that is $\Sigma_{imp}\subseteq\Sigma_c$. The rest of events are known as *informative* (i.e. $\Sigma_{inf}\subseteq\Sigma$) and $\Sigma_{imp}\bigcap\Sigma_{inf}=\emptyset$. The sequential behavior of the AMS is captured by the non-marked and marked *-languages $L(P)$ and $L_m(P)$ generated by the automaton P. The closed-loop behavior $L(V/P)$ is constructed in a standard fashion [17] and $L_m(V/P)=L_m(P)\cap L(V/P)$. In other words, the plant automaton P interacts with the control device V to restrict the set of feasible trajectories. In this work, *supervisors* are used as control devices. A supervisor is a map $V: L(P)\rightarrow\Gamma$ where each component of the control pattern $\gamma\in\Gamma$ contains at least all plant–enabled uncontrollable transitions. If the supervisor V is realized by the pair $\mathcal{S}:=(S,\phi)$ where S is an automaton $S=(Y,\Sigma,\delta_S,y_0,Y_m)$ and $\phi: X\rightarrow\Gamma$, then control action is effected by $\mathcal{S}(s)=\phi(\delta_S(y_0,s))$ for each $s\in L(P)$ maintaining the plant controllable whilst satisfying in a maximally permissive fashion the safety specifications established by the designer [17]. Decision mechanisms to choose among controllable events are not subject of this work. Efficient tools for supervisor synthesis and other calculations mentioned in this work can be found in the open literature (e.g. [14,16,17]).

3. Reliable automaton and language achievability

The construction of a supervisory controller guaranteeing the manufacturability of a given product is based on the notions of reliable automaton and achievable language whose definitions are given below. In the rest of the paper, $G=(X,\Sigma,\delta_G,x_0,X_m)$ is a standard automaton, $\|$ denotes the standard synchronous product between automata and $Proj:\Sigma^*\rightarrow\Sigma_{imp}^*$ is the canonical projection over Σ_{imp}.

Definition 1 (*Decision state*). For a given automaton G, a state $x\in X$ is said to be a decision state if, for every $\sigma\in\Sigma$ such that $\delta_G(x,\sigma)$ is defined, it holds that $\sigma\in\Sigma_c$ and there exists $\rho\in\Sigma_{imp}$ such that $\delta_G(x,\rho)$ is also defined.

In other words a state $x\in X$ is a decision state if all its enabled events are controllable and at least one of them is imperative.

Definition 2 (*Attainable states*). For a given automaton G, a state $\tilde{x}\in X$ is said to be attainable from $x\in X$ and denoted $x\dashrightarrow\tilde{x}$ if there exists $s\in\Sigma_{Inf}^*$ such that $\delta_G(x,s)=\tilde{x}$.

A state is thus attainable from another state if there exist at least one path from the origin state to the attainable state formed only with informative events.

Definition 3 (*Reliable automaton*). An automaton G is said to be reliable if all imperative events σ satisfy:

(1) for all $x\in X$ with $\delta_G(x,\sigma)$ defined, there exists a decision state $\tilde{x}$ which is attainable (i.e. $x\dashrightarrow\tilde{x}$),
(2) $\delta_G(\tilde{x},\sigma)$ is also defined.

Definition 3 means that, over a reliable automaton, all imperative events can eventually be executed. The following algorithm verifies whether an automaton G is reliable.

```
Let $f:\Sigma_{imp}\rightarrow 2^X$ s.t. $f(\sigma)$ is a set of decision states.
Let $g:\Sigma_{imp}\rightarrow 2^X$ s.t. $\exists\delta_G(f(\sigma),\rho)$ with $\rho\in\Sigma_{inf}$
For each $\sigma\in\Sigma_{imp}$ do
        If $f(\sigma)=\emptyset$ then $G$ not reliable, finish
        Else if $f(\sigma)\neq\emptyset$ and $g(\sigma)\neq\emptyset$ then
            Eliminate all imperative events from $G$
            If $g(\sigma)\notin Co(f(\sigma))$ then $G$ not reliable, finish
        Next
        Return $G$ reliable
```

Co(f) is the correachable set of f computed iteratively as the pre-image of f. It is easy to see that the synchronization of automata with disjoint event sets is also reliable.

Now consider an automaton $M=(Y,\Sigma_{imp},\delta_M,y_0,Y_m)$ whose languages describe the recipe procedures (i.e. the set of manufacturing commands sequences) for a particular product in terms of imperative events.

Definition 4 (*Achievable language*). For automata M and G, language $L(M)$ is said to be achievable over G if:

(1) $L(M)\subseteq Proj(L(G))$ and
(2) if $w_i\in L(G)$ and $w_i\in Proj^{-1}(s_i)$ with $s_i\notin L_m(M)$, then there exist a decision state $x_i=\delta_G(x_0,z_i)$ such that $\delta_G(x_0,w_i)\dashrightarrow x_i$ and $Proj(z_i)\xi\in L(M)$, where ξ is an enabled event at x_i.

If language $L(M)$ is achievable over closed-loop automata V/P, then all manufacturing sequences in $L(M)$ can be followed event by event over the closed-loop automaton V/P and thus all the required imperative events can be executed in a reliable way. By definition, an achievable language does not violate the safety and process specifications of the low level closed-loop behavior. The following result characterizes those languages that are achievable over a given automaton. The proof can be found in Appendix A.

Proposition 1. *Given a reliable automaton G, language $L(M)$ is achievable on G if $Proj(L(M\|G))$ is isomorphic to $L(M)$.*

The condition given in Proposition 1 can be verified carrying out the indicated operations. Moreover, a proper construction of a hierarchical control architecture can diminish the effort of modeling the recipe-procedure sequences, as it will be illustrated in the example.

4. Example

An schematic view of the AMS exemplar is shown in Fig. 1. Two dispatchers (D1 and D2) deliver feedstock-bricks A (black) and B (red) to two workstations (WS1 and WS2) using a rectangular conveyor belt. Each workstation is composed by a feeder, a processing machine and an unloading device for finished parts. Processing tasks are executed to manufacture products according to given recipes procedures. The number of feedstock-bricks in the conveyor belt and workstations can be restricted to a fixed amount for safety reasons. The AMS exemplar was built using Lego® components as shown in Fig. 2. The dispatcher mechanism is shown in Fig. 3. A retracting pusher powered by a motor pulls the incoming brick into the conveyor belt and then returns to its original position. Details of the workstation are shown in Fig. 4. The feeder assembly is placed above the conveyor belt. It contains a light sensor, a crank-shaft powered by a motor and a touch sensor. The light sensor detects the feedstock-brick of the desired color. Once the brick is detected, the crank-shaft pushes the brick into the processing machine. The number of turns of the crank-shaft must be determined experimentally depending on the speed of the conveyor belt and motors employed. The touch sensor detects the turns of the crank-shaft.

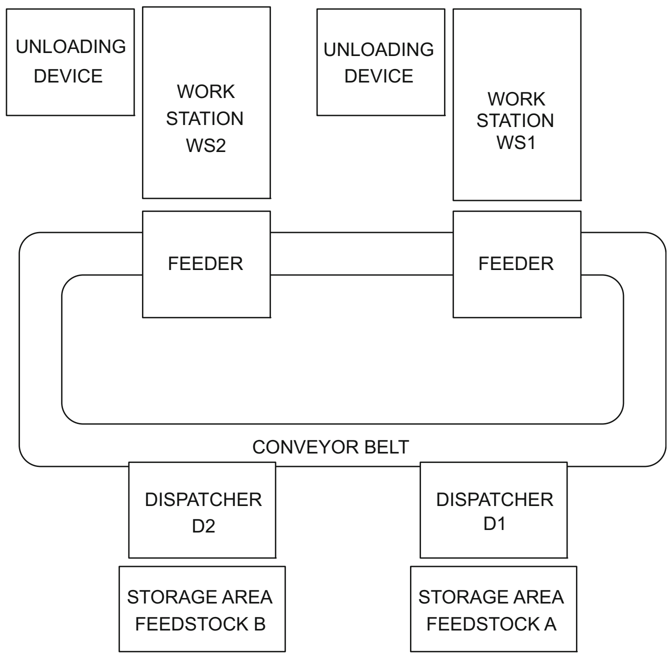

Fig. 1. Block diagram of AMS exemplar.

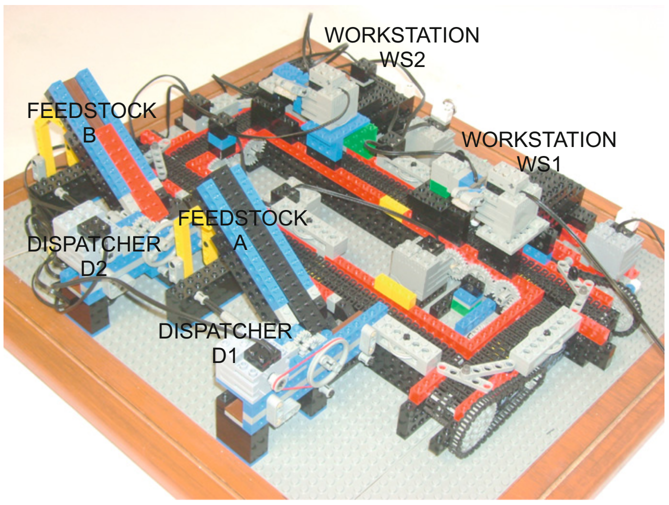

Fig. 2. AMS implementation with Lego®.

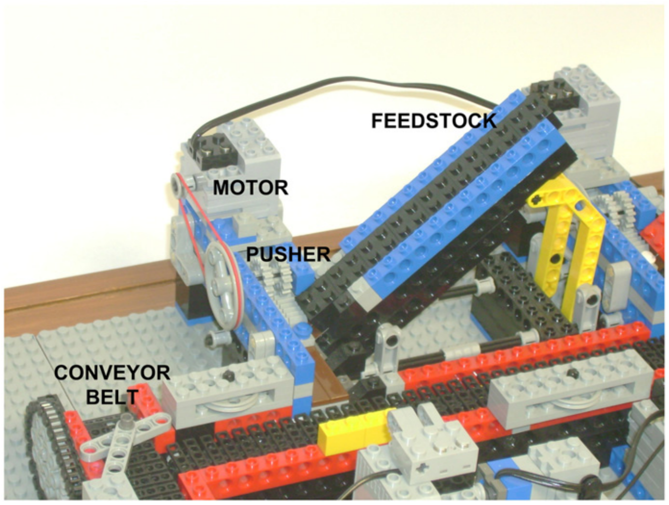

Fig. 3. Detail of feedstock dispatcher.

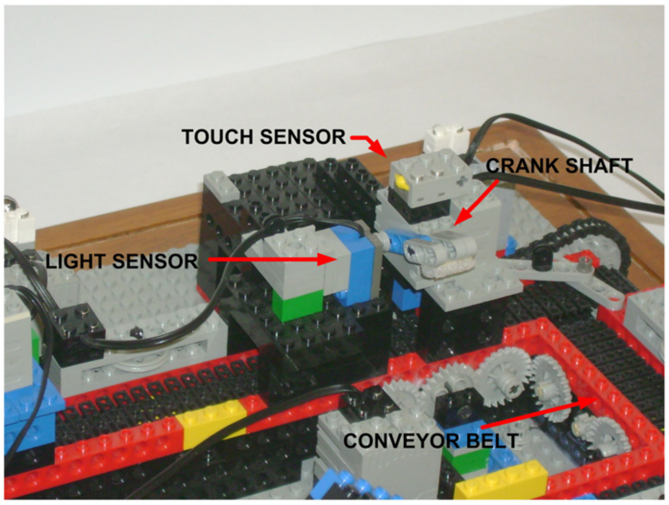

Fig. 4. Detail of workstation.

The processing machine is a receptacle from where the brick is removed once the processing time expires using a similar mechanism to the dispatcher pusher.

Tables 1 and 2 show the proposed ISA88 *equipment* and *procedural control* models [7] for the coordination layer of AMS exemplar. The design rationale was to keep this architecture simple in order to show in a graphic manner the use of the proposed approach for guaranteeing the manufacturability of products. Both *phase* and *unit procedure* control modules deal with equipment operation of the AMS. The *procedure* level contains the recipe procedures for fabricating specific products modeled as an ordered set of phase-status events which are employed as the linking mechanism among the different levels. The recipe procedures may be the result of a previous planning stage. Modules at each level (phase and unit procedure) do not share events thus avoiding SCT-conflict problems. Workstations WS1 and WS2 are considered as *equipment units* as well as both raw material dispatchers together with the conveyor belt, identified with label DRM. Equipment phases are associated to equipment modules. The resulting equipment modules and devices (both motors and sensors) are shown in their corresponding columns of Table 1. A graphical representation of the coordination layer is shown in Fig. 5. For space economy, broken line means that modules are repeated for each workstation. An input/output layer is introduced in the process model as to communicate the equipment phases with manufacturing equipment using *device drivers* and *state observers* [15]. A device driver translates control instructions from the equipment phase to the equipment and vice versa, calculating the discrete state of its associated manufacturing equipment using a discrete-event model. The state observers extract occurrences of discrete events from continuous measurements.

Table 1
ISA88 equipment model of the AMS example.

Cell	Eq. unit	Eq. module	Elementary eq. components
AMS	DRM	Dispatchers	Feeding motors
		Conveyor belt	Unit counter
	WS1	WS1 feeder	Motor, light and touch sensors
		WS1 machine	Delivering motor
	WS2	WS2 feeder	Motor, light and touch sensors
		WS2 machine	Delivering motor

Table 2
ISA88 procedural control model of the AMS example.

Proc.	Unit proc.	Equipment phase
Manufacture product	Dispatch raw material (UPDRM)	Dispatch raw material A or B (PhDRMAB)
	Process in WS 1 (UPPWS1)	Feed raw material A or B (PhFRMAB)
		Process and deliver (PhPD)
	Process in WS 2 (UPPWS2)	Feed raw material A or B (PhFRMAB)
		Process and deliver (PhPD)

5. Supervisor synthesis for equipment operation

Supervisors are associated to equipment phases and unit procedures according to the coordination layer model of Fig. 5. Both phase and unit procedure supervisors enforce safety

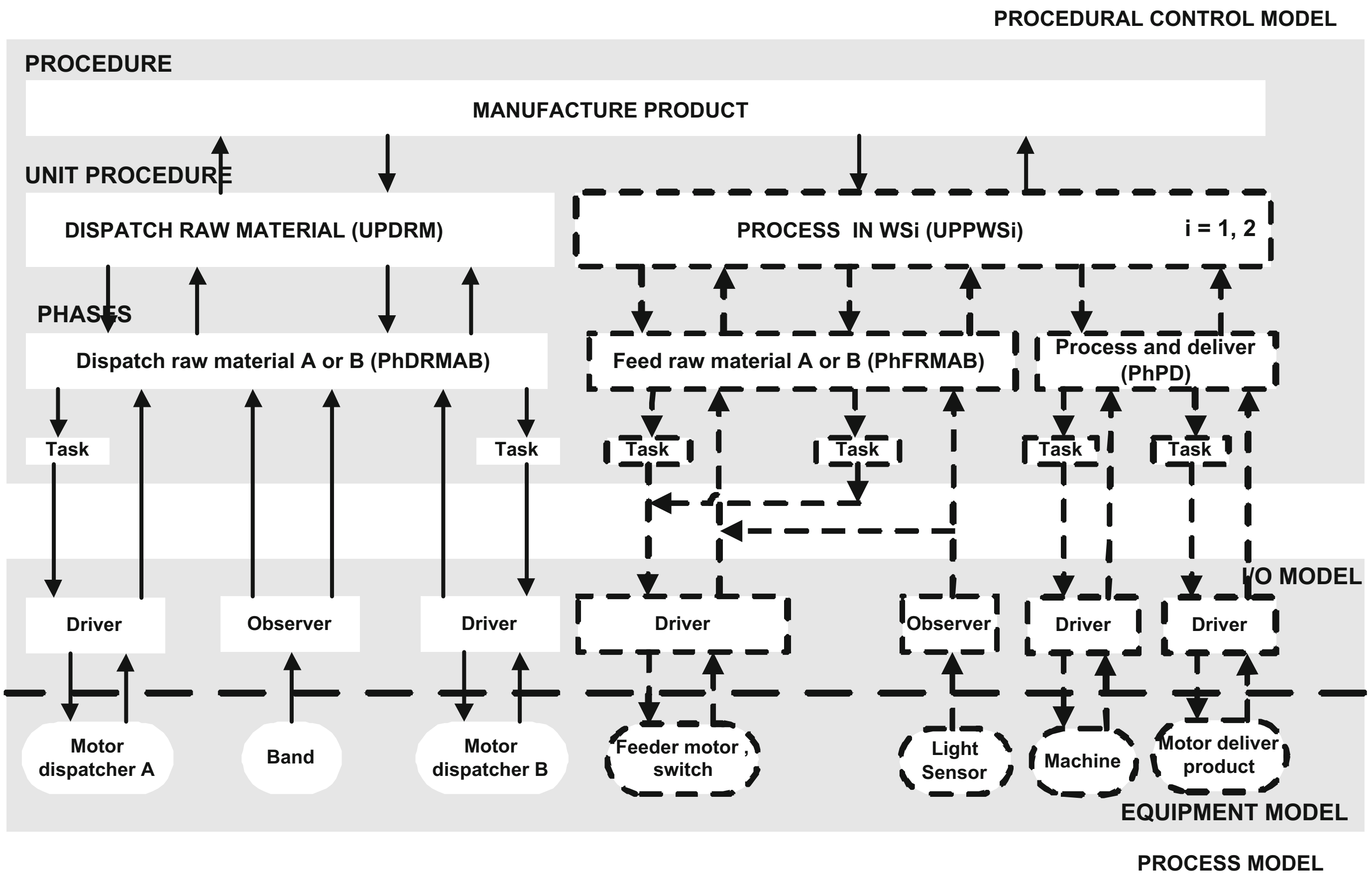

Fig. 5. Hybrid model of AMS exemplar.

specifications during operation of the AMS, leaving to the recipe procedures the phase sequencing required for manufacturing specific products. Phase supervisors deal with the associated components of the equipment unit whilst unit procedure supervisors does it for the safe execution of phases as requested by the production orders given in the recipe procedures. The synthesis can be carried out using standard procedures and tools. In this work, the synthesis procedure shown in Fig. 6 adapted from [14] was employed. Rectangles and ovals represent information entities and tasks, respectively. Controllable events starting-up phases are declared as imperative and are employed as the "linking" events between the recipe procedure and equipment supervisors (at unit procedure and phase levels). Therefore, an adequate building of elementary components as automata is of upmost relevance to facilitate the obtention of achievable languages on the closed-loop automata of unit operation and phase levels.

For the phase level, three different supervisors were synthesized: "Dispatch raw material A or B (PhDRMAB)", "Feed raw material A or B (PhFRMAB)" and "Process and deliver (PhPD). As an example, Fig. 7 shows the input and resulting automata for "Feed Raw Material A or B (PhFRMAB)" phase. The figure includes elementary component models (process equipment and status information) and specifications as well as the automaton representation of the closed-loop behavior FSM acting as supervisor. Table 3 shows the event set with its physical interpretation. The imperative events for this case are the phase start-up (35, 37). The plant model results from the synchronous product of elementary components plus the causal behavior automaton indicating that a phase cannot finish before motors are switched off. Five functional specification automata describe the desired operation. For instance, specification "Motor start-up" declares that the motor must be switched on only when raw material is detected. Specification "Feed raw material A" indicates that if material A is requested, no other material must be fed to the workstation. The monolithic specification automaton is obtained from the synchronization of the individual functional specifications. Reliability can be visually verified in the closed-loop behavior automaton since imperative events 35 and 37 are executed from state 0 as the only option. The other two supervisors are

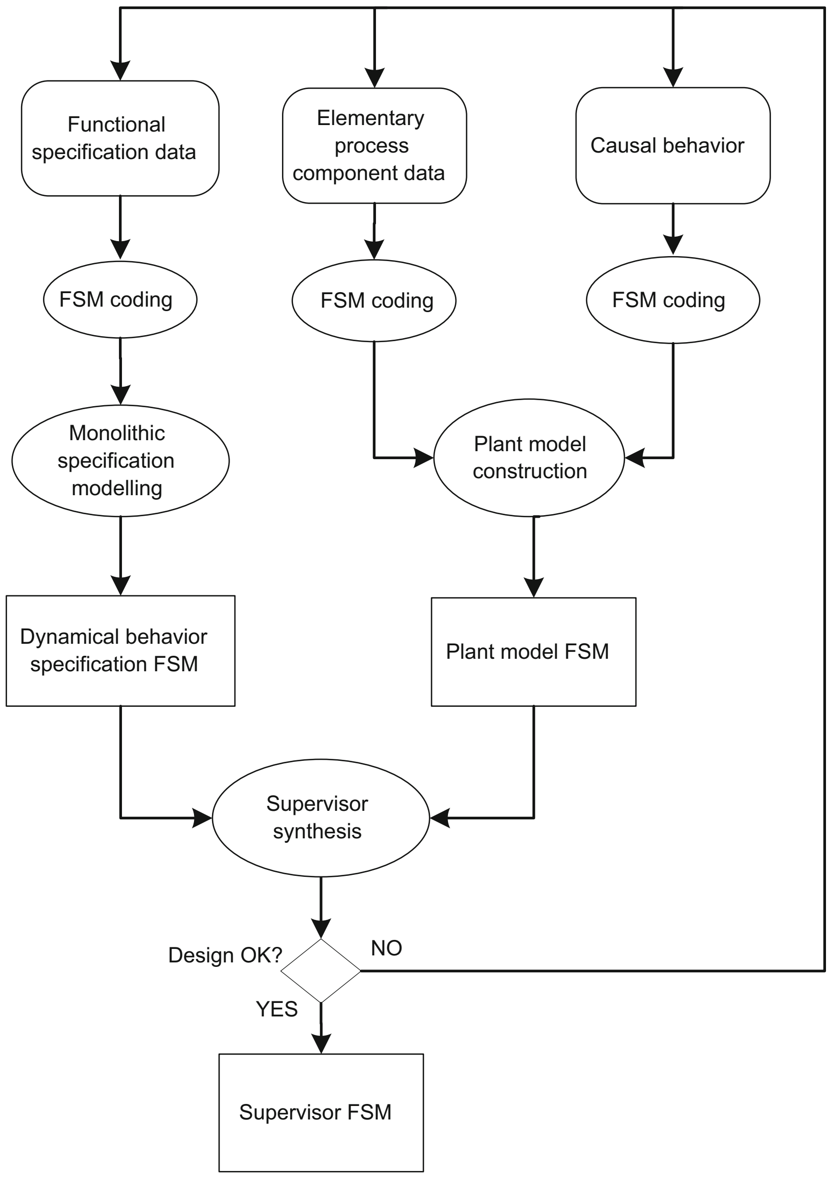

Fig. 6. Supervisory synthesis procedure after [14].

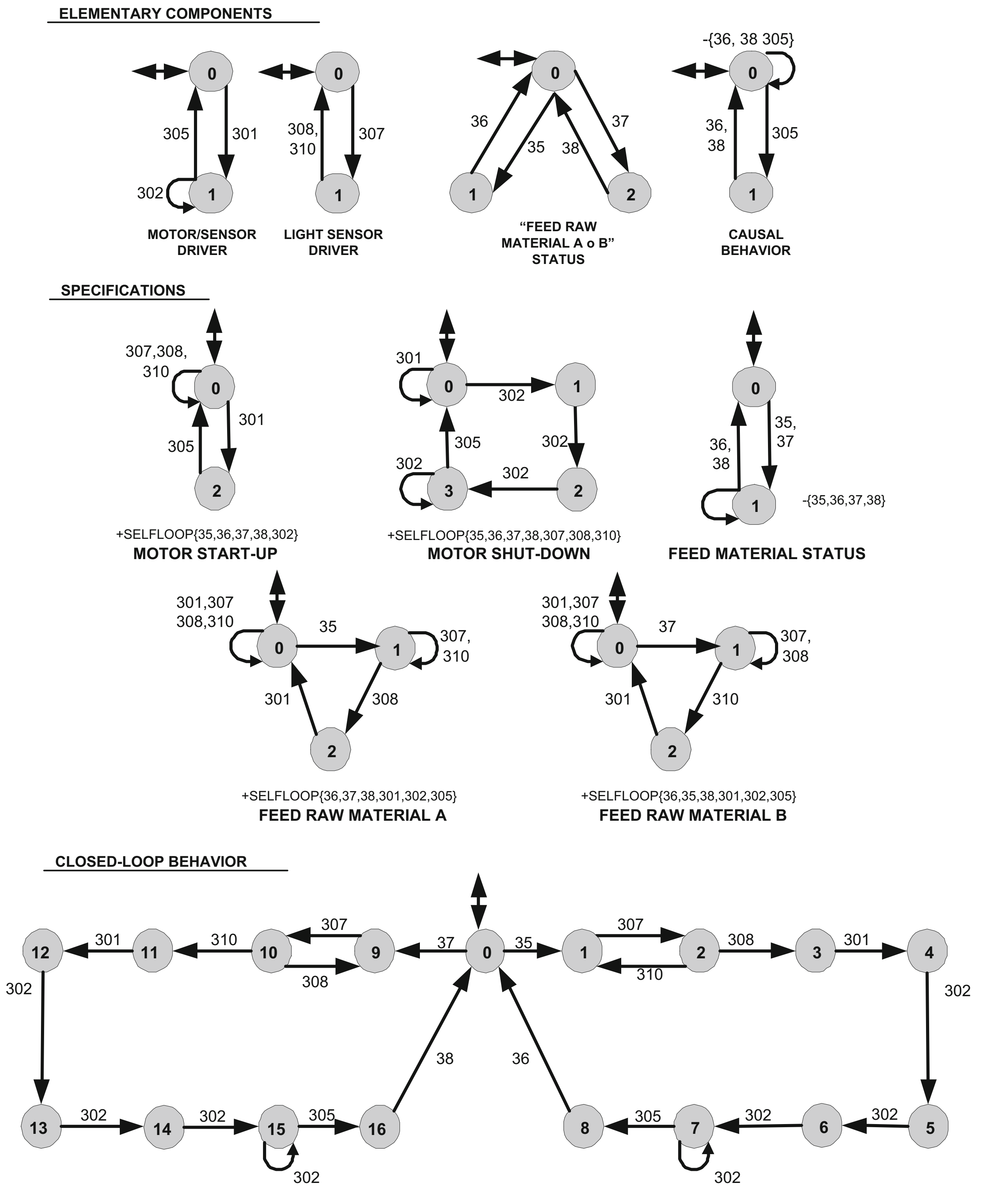

Fig. 7. Supervisor synthesis of "Feed Raw Material A or B in WS1" phase.

synthesized in the same fashion. Reliability is verified for each automaton and their synchronization.

As an example of unit procedure supervisor synthesis, Fig. 8 shows information of "Process in WS1 (UPPWS1)" procedure, including automata models for process components, specification and the synthesized closed-loop behavior. Note that elementary component automaton "Feed Raw Material A or B" is the same as in the phase supervisor synthesis. Specification automata describe the phase sequencing and maximum load of raw material allowed for WS1 (two blocks in this case) giving rise to a closed-loop automata of a manageable size capable of handling recipes of one or two feedstock blocks. The closed loop behavior automaton can be observed that is also reliable since imperative events 35, 37 and 51 are executed only from decision states. Similar to the phase supervisor synthesis, the "Dispatch Raw Material" supervisor gives rise to a reliable automation, thus the synchronization

Table 3
Event sets for example phase and procedure of AMS.

Evt	Meaning	Type
35	Phase “FRMA” start-up with raw material A	Imperative
36	Phase “FRMA” finishing with raw material A	Uncontrollable
37	Phase “FRMB” start-up with raw material B	Imperative
38	Phase “FRMB” finishing with raw material B	Uncontrollable
50	Phase “Op. WS” finishing	Uncontrollable
51	Phase “Op. WS” start-up	Imperative
301	Switch-on motor leftwise	Controllable
302	Touch sensor being activated	Uncontrollable
305	Switch-off motor	Controllable
307	Switch-on light sensor	Controllable
308	Light sensor detects RM A	Uncontrollable
310	Light sensor detects RM B	Uncontrollable

at this level is also reliable. The supervisor synthesis of “Process in WS2” is identical just changing event labels accordingly.

Automata sizes for all synthesized supervisors are included in Table 4. As can be observed, state spaces are rather small in all cases. Thus, calculations can be easily carried out.

6. Recipe procedures modeling and implementation

A recipe procedure can be modeled as an ordered set of process segments [8]. In this exercise, equipment allocation is resolved “ad hoc” for each recipe and IDEF3 *process flow models* [12] are employed to capture such procedures. As an example, Fig. 9 shows a recipe procedure for manufacturing a product labeled as “A5” fabricated with one unit of each raw material. The feeding

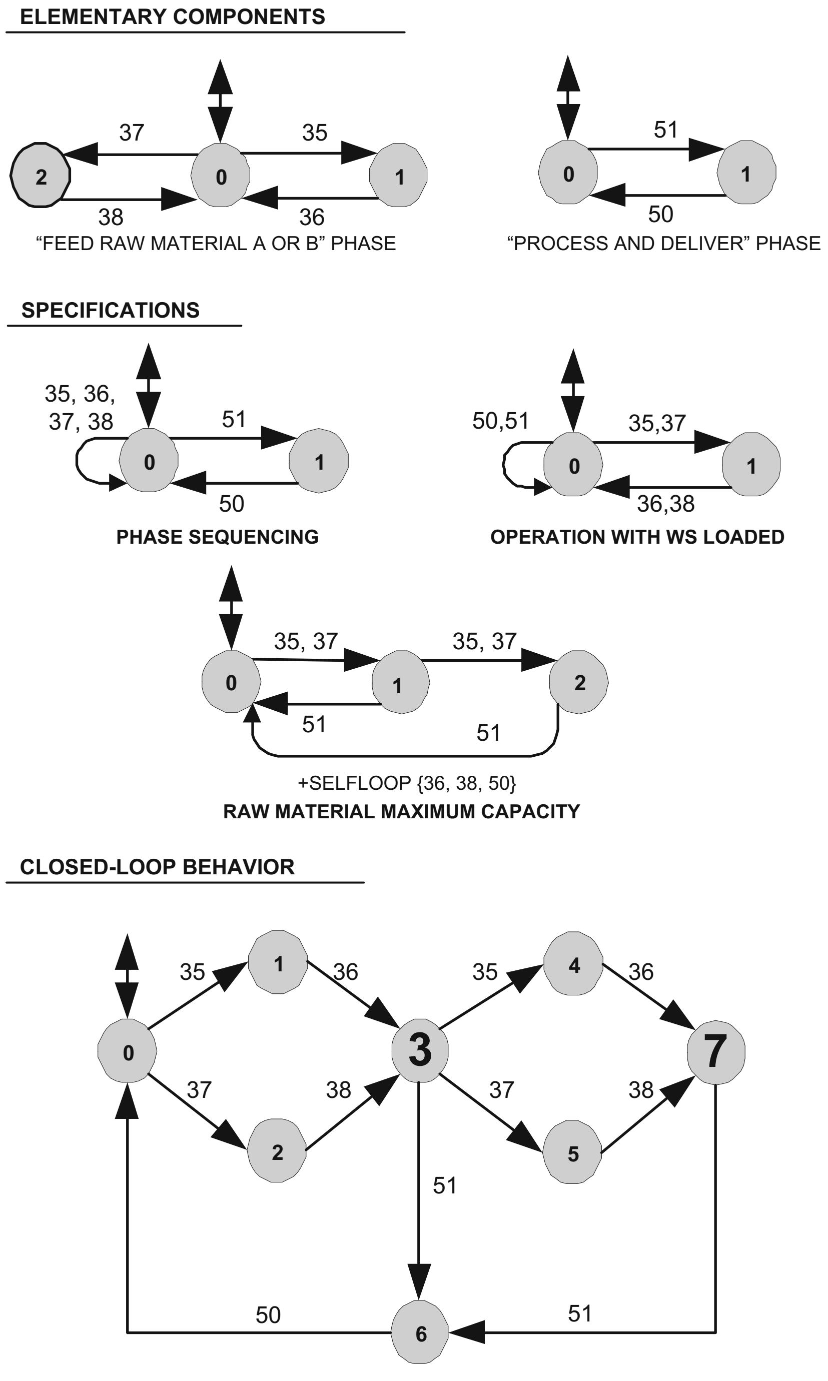

Fig. 8. Supervisor synthesis of “Process in WS1” procedure.

order is established as block A first (PhDRMA) followed by bock B (PhDRMB). Unit procedures are represented by IDEF3 *behavior units*. Phase sequencing is declared in the *resource* section of each behavior unit. Since the event sets of unit procedures are disjoint, each unit procedure can be translated into an automaton built with imperative events to drive the execution of the equipment unit for the sequential fabrication of a number of given products regardless of the operation of other units. Thus, DRM unit will dispatch as many blocks of raw material as indicated by the recipe procedure (in this case, one red block first (transition 11) followed by one black block (transition 21)) whilst workstations will take the raw material block A first (transition 35) and then feed raw material block B (transition 37). Other modes of operations have considered previously for similar AMS arrangements. For instance, campaign mode with inter-unit synchronization [9].

Before using each resulting automaton for driving the AMS equipment units, the *achievability* of the product must be verified as indicated in Proposition 1. For this particular AMS, phase sequence restrictions were established as part of the behavior specifications of the supervisor (e.g. in Fig. 8 raw material must be loaded before operation in workstation WS1 can start). Thus, *achievability* can be satisfied by taking into consideration these restrictions when writing the phase sequencing of behavior units. For example, in the recipe procedure for fabricating product ”A5” the raw material parts must be fed in an specific order into the workstation. Workstation “WS1” is chosen and the feeding sequence is translated into the automaton identified as “Process in WS1”. Achievability is visually verified. For larger production facilities or more complex recipe procedures, standard tools can be used to compute the required products and projections.

Recipe procedures are automatically translated into automata [6]. The hierarchical architecture including these procedures and equipment supervisors is implemented as a Simulink® model [11]. Data acquisition is carried out with a standard data acquisition PC-card and a signal condition interface built specifically for this purpose [4].

Table 4
Supervisor sizes of the AMS example.

	# States		
	Plant	**Specs**	**Sup**
Phase supervisor			
Dispatch raw material (A and B)	640/748	120	23
Feed raw material (A and B)	16/18	80/112	17
Process and deliver	40	36	12
Unit procedure supervisor			
Dispatch raw material	4	–	4
Process in WS i	6	6	8

7. Conclusions

The proposed approach makes a clear separation between product production control and equipment procedural actions. Conditions are established to guarantee that a given recipe procedure can be executed in a SCT-controllable and reliable AMS. However, a reliable automaton may impose conservative operation conditions to some AMS. Introducing new products for fabrication requires only the writing of its corresponding recipe procedure using imperative events and verifying achievability. Such recipe procedures may be the result of a planning stage. Compatible automata frameworks including optimality criteria [3] have been proposed in the past and could be used for this design task.

The coordination control architecture used in this work emphasizes the advantages of separating the production and control equipment activities. Future work will explore more complex relations among phases [10] in which hierarchical supervision satisfying both controllability and reliability at all levels can play an important role. Nonetheless, even a simple architecture reduces the size of supervisors and consequently the synthesis and maintenance efforts. As a comparison of the effort reduction, Table 5 presents the state space sizes for the plant, specification and supervisor automata resulting from synthesizing the closed-loop behavior considering each layer and the complete coordination controller as one supervisor. Calculation were executed using tools publicly available [14].

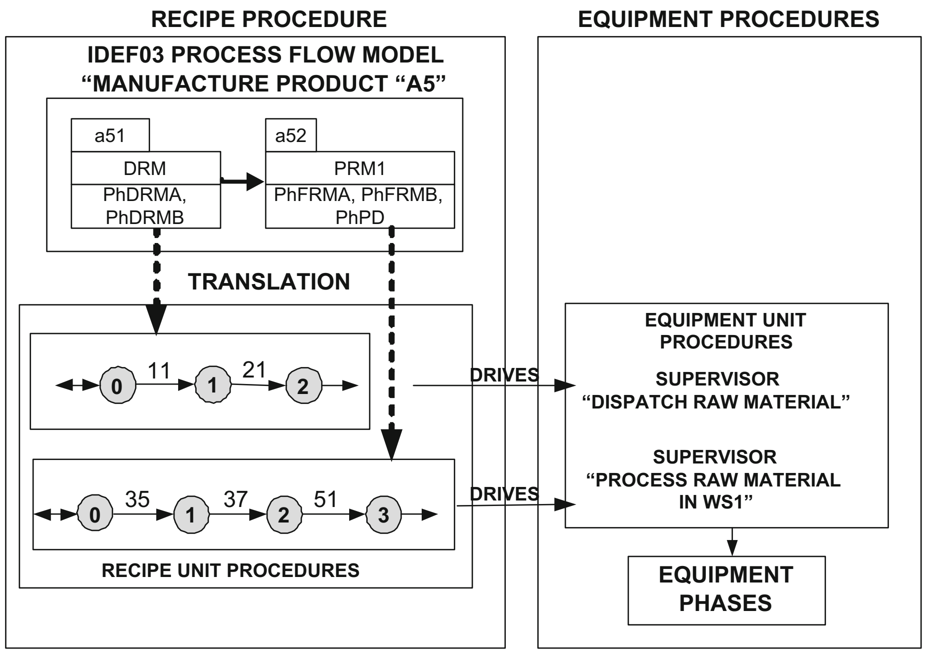

Fig. 9. Recipe implementation.

Table 5
Supervisor sizes of the AMS example for non-modular architectures.

Layer	States		
	Plant	Spec	Supervisor
Phase layer	$2.98(10^8)$	$1.4(10^9)$	957,168
Unit procedure layer	108	36	192
Monolithic	$32.2(10^9)$	$1.9(10^9)$	78,292

Acknowledgments

The authors kindly acknowledge partial financial support from CONACYT, Mexico in the form of three scholarships and Grants 36934U and 42093.

Appendix A

Proposition. *Given a reliable automaton G and automaton M, language $L(M)$ is achievable on G if $L(M) \subseteq Proj(L(G))$.*

Proof. Let $s \in L(M)$, $\sigma \in \Sigma_{imp}$ and $s\sigma \in L(M)$ with $s \notin L_m(M)$. Since $L(M) \subseteq Proj(L(G))$, there exists $w \in L(G)$ such that $w \in Proj^{-1}(s\sigma)$. Note that w is formed as $w = w_1 w_2 \sigma w_3$ with $w_1 \in Proj^{-1}(s)$, $w_2, w_3 \in \Sigma_{inf}^*$. Then, for $\delta_G(x_0, w_1)$ there exists an attainable state $x_1 = \delta_G(x_0, w_1 w_2)$ such that $Proj(w_1 w_2)\sigma \in L(M)$. Moreover, since G is a reliable automaton, there exists a decision state x_2 such that $x_1 \dashrightarrow x_2$, and in x_2, σ is enabled. □

A computationally easier alternative to verify the premise of this proposition is establishing if $Proj(L(M) \| L(G)) = L(M)$.

References

[1] Balemi S, Hoffmann GJ, Gyugyi P, Wong–Toi H, Franklin GF. Supervisory control of a rapid thermal multiprocessor. IEEE Transactions on Automatic Control (Joint Issue with Automatica) 1993;38(7):1040–59.

[2] Chandra V, Huang Z, Kumar R. Automated control synthesis for an assembly line using discrete event system control theory. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 2003;33(2):284–9.

[3] Darabi H, Jafari MA, Manapure SS. Finite automata decomposition for flexible manufacturing systems control and scheduling. IEEE Transactions on Systems, Man, and Cybernetics 2003;33(2):168–75.

[4] Garcia FA. Formal development of logic controllers. A case study in automated manufacturing systems. Master's thesis, Cinvestav, Unidad de Ingenieria Avanzada; 2007 [in Spanish].

[5] Gouyon D, Petin JF, Morel G. Control synthesis for product-driven automation. In: Proceedings of IFAC workshop on discrete event systems, September 22–24, Reims, Fr, 2004.

[6] Hernandez E. Hierarchical-modular modeling and control of resource coordination for flexible manufacturing systems. Master's thesis, Cinvestav, Seccion de Mecatronica; 2005 [in spanish].

[7] ISA-88.01. Batch control systems. Part 1. Models and terminology. Standards. Technical report, Instrument Society of America; 1995.

[8] ISA-95.1. Enterprise—control system integration. Part 1: models and terminology. Standards. Technical report, Instrument Society of America; 1999.

[9] Jaimes-Santin F, Sanchez A, Aranda-Bricaire E. Synthesis of supervisory controllers considering product production rules in automated manufacturing systems. In: Proceedings of fourth international symposium on robotics and automation, August 7–9, Queretaro, Mexico, 2005.

[10] Leduc RJ, Lawford M, Dai P. Hierarchical interface-based supervisory control of a flexible manufacturing system. IEEE Transactions on Control Systems Technology 2006;14(4):654–68.

[11] Llamas LE. Synthesis of coordination architectures in automated manufacturing systems. Master's thesis, Cinvestav, Unidad de Ingenieria Avanzada; 2008 [in spanish].

[12] Mayer RJ, Menzel CP, Painter MK, deWitte PS, Blinn T, Perakath B. Information integration for concurrent engineering. Process flow and object state description capture. Technical report AL-TR-1995, Knowledge Based Systems, College Station Texas; September 1995.

[13] Ramirez-Serrano A, Benhabib B. Supervisory control of multiworkcell manufacturing systems with shared resources. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 2000;30(5):668–83.

[14] Sanchez A, Douriet J, Ramirez E. Formal synthesis of a class of discrete-event controllers for large manufacturing systems. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 2007;37(4): 662–9.

[15] Sanchez A, Parra LF, Baird R, Macchietto S. Hybrid modelling and dynamic simulation of automated batch plants. ISA Transactions 2003;42:401–20.

[16] Sanchez A, Reza J, Douriet J, Gonzalez RE. A comparison of synthesis tools for supervisory controllers. In: Proceedings of European Control Conference, September 2003.

[17] Wonham WM. Supervisory control of discrete event systems. University of Toronto; 2008.

[18] Yalcin A. Supervisory control of automated manufacturing cells with resource failures. Robotics and Computer-integrated Manufacturing 2004;20:111–9.