An Identification Method for PLC-based Automated Discrete Event Systems

Ana Paula Estrada-Vargas, E. López-Mellado, Jean-Jacques Lesage

▶ **To cite this version:**

Ana Paula Estrada-Vargas, E. López-Mellado, Jean-Jacques Lesage. An Identification Method for PLC-based Automated Discrete Event Systems. 49th IEEE Conference on Decision and Control (CDC'10), Dec 2010, Atlanta, Georgia, United States. pp. 6740-6746. hal-00596551

HAL Id: hal-00596551
https://hal.science/hal-00596551

Submitted on 27 May 2011

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.

An Identification Method for PLC-based Automated Discrete Event Systems

Ana Paula Estrada-Vargas, Ernesto López-Mellado, Jean-Jacques Lesage, *Members IEEE*

Abstract— **This paper deals with identification of automated Discrete Event Systems (DES). A method for processing sequences of input/output signals from PLC-based controlled DES is proposed; it yields an interpreted Petri net model describing the closed-loop behavior of the automated DES. This new method extends a previously presented approach by taking into account the technological characteristics of industrial controllers and the data collection requirements. Based on polynomial-time algorithms, the method is implemented as an efficient software tool; its use is illustrated through a case study dealing with an automated manufacturing system.**

Keywords— Automated DES, Identification, Interpreted Petri Nets, Programmable Logic Controllers

I. INTRODUCTION

In recent years, the scientific community has proposed identification approaches (based on Petri net or automata) for obtaining approximated models of DES whose behavior is unknown or ill-known. In the context of automated DES, identification methods can be complementary to established modeling techniques; identification builds a closed-loop controller-plant model, which is generally obtained by a synchronous composition operation. Three main approaches for identifying DES have been proposed in literature [1].

The incremental synthesis approach, proposed in [2][3] deals with unknown partially measurable concurrent DES exhibiting cyclic behavior. Several algorithms have been proposed allowing the on-line building of interpreted Petri net (PN) models from the DES outputs. Although the techniques are efficient, the obtained models may represent more sequences than those observed.

In [4] a method to build a free labeled PN from a finite set of transitions strings is presented. This method is based on the resolution of an Integer Linear Programming (ILP) problem; the obtained PN generates exactly the observed language. This approach has been extended to other IPN classes [5], [6], [7]; however, issues regarding applications to actual DES have not yet been addressed in these works.

Another recent off-line method [8] allows building a non deterministic finite automaton (FA) from a set of input/output sequences, experimentally measured from the DES to be identified. Under several hypotheses, the constructed FA generates exactly the same input/output (I/O) sequences of given length than observed ones. The method was conceived for fault detection in a model-based approach [9]. Extensions to this work propose an identification method performing optimal partitioning of concurrent subsystems for distributed fault detection purposes [10].

In a previous paper we proposed a method for synthesizing interpreted PN (IPN) for coping with concurrent partially observable DES [11]; it processes a set of cyclic sequences of binary output signals yielding models including silent transitions and non labeled places.

In this paper that method has been extended for dealing with sequences of I/O signals captured during the closed-loop operation of PLC-based controlled DES. Several technological characteristics of industrial controllers are taken into account in data collection and processing. The obtained model is a safe (1-bounded) IPN describing the controller-plant concurrent behavior, including those that are not measurable directly from the PLC.

The paper is organized as follows. In section II the background on Petri nets and languages is outlined. Based on these definitions the problem of DES identification is stated in section III in terms of language associated to an IPN. Several constraints inherent of real controlled DES are analyzed in section IV. The identification algorithm is given in section V. Finally, a case study is included in section VI.

II. BASICS ON PETRI NETS AND LANGUAGES

This section presents the basic concepts and notation of PN and IPN used in this paper.

Definition 1: An ordinary Petri Net structure G is a bipartite digraph represented by the 4-tuple G = (P,T,I,O) where: $P = \{p_1,p_2,...,p_{|P|}\}$ and $T = \{t_1,t_2,...,t_{|T|}\}$ are finite sets of vertices named places and transitions respectively; $I(O) : P \times T \rightarrow \{0, 1\}$ is a function representing the arcs going from places to transitions (from transitions to places).

The symbol ${}^\bullet t_j$ ($t_j{}^\bullet$) denotes the set of all places p_i such that $I(p_i,t_j) \neq 0$ ($O(p_i,t_j) \neq 0$). Such places are called input (output) places of t_j. Analogously, ${}^\bullet p_i$ ($p_i{}^\bullet$) denotes the set of input (output) transitions of p_i.

The incidence matrix of G is $C = C^+ - C^-$, where $C^- = [c_{ij}^-]$; $c_{ij}^- = I(p_i,t_j)$; and $C^+ = [c_{ij}^+]$; $c_{ij}^+ = O(p_i,t_j)$ are the pre-incidence and post-incidence matrices respectively.

A marking function $M : P \rightarrow \mathbb{Z}^+$ represents the number of tokens residing inside each place; it is usually expressed as an |P|-entry vector. $\mathbb{Z}^+$ is the set of nonnegative integers.

E. López-Mellado is with CINVESTAV Unidad Guadalajara. Av. Científica 1145, Col. El Bajío 45015 Zapopan, Mexico. Email: elopez@gdl.cinvestav.mx. J-J. Lesage is with LURPA Ecole Normale Supérieure de Cachan. 61, av du Président Wilson, 94235 Cachan Cedex, France. Email: Jean-Jacques.lesage@lurpa.ens-cachan.fr A.P. Estrada-Vargas is with both Institutes. Email: aestrada@gdl.cinvestav.mx

Definition 2: A Petri Net system or Petri Net (PN) is the pair $N = (G, M_0)$, where G is a *PN* structure and M_0 is an initial marking.

In a *PN* system, a transition t_j is *enabled* at marking M_k if $\forall p_i \in P$, $M_k(p_i) \geq I(p_i, t_j)$; an enabled transition t_j can be fired reaching a new marking M_{k+1} which can be computed as $M_{k+1} = M_k + Cv_k$, where $v_k(i) = 0$, i≠j, v_k(j) = 1, this equation is called the *PN* state equation. The reachability set of a *PN* is the set of all possible reachable marking from M_0 firing only enabled transitions; this set is denoted by $R(G,M_0)$. A PN is called *safe* if $\forall M_k \in R(G, M_0), \forall p_i \in P, M_k(p_i) \leq 1$.

Now it is defined IPN, an extension to PN that allows associating input and output signals to PN models.

Definition 3: An IPN (Q, M_0) is a net structure $Q = (G, \Sigma, \Phi, \lambda, \varphi)$ with an initial marking M_0.

- G is a *PN* structure, $\Sigma = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the input alphabet, and $\Phi = \{\phi_1, \phi_2, ..., \phi_q\}$ is the output alphabet.
- $\lambda : T \rightarrow \Sigma \cup \{\varepsilon\}$ is a labeling function of transitions, where ε represents a system internal event externally uncontrollable; it is not allowed that the symbol ε is associated to more than one $t_j \in p_i^{\bullet}$.
- $\varphi : R(Q,M_0) \rightarrow (\mathbb{Z}^+)^q$ is an output function, that associates to each marking in $R(Q,M_0)$ a q-entry output vector; q=|Φ| is the number of outputs. φ is represented by a q×|P| matrix, such that if the output symbol ϕ_i is present (turned on) every time that $M(p_j) \geq 1$, then $\varphi(i,j) = 1$, otherwise $\varphi(i,j) = 0$.

When an enabled transition t_j is fired in a marking M_k, then a new marking M_{k+1} is reached. This behavior is represented as $M_k \xrightarrow{t_j} M_{k+1}$; the state equation is completed with the marking projection $y_k = \varphi M_k$, where $y_k \in (\mathbb{Z}^+)^q$ is the k-th output vector of the *IPN*.

According to functions λ and φ, transitions and places of an *IPN* (Q,M_0) can be classified as follows.

Definition 4: If $\lambda(t_i) \neq \varepsilon$ the transition t_i is said to be controllable (t_i can be fired when the associated input symbol is presented). Otherwise it is uncontrollable (t_i is autonomously fired). A place $p_i \in P$ is said to be measurable if the i-th column vector of φ is not null, i.e. $\varphi(\bullet, i) \neq 0$. Otherwise it is non-measurable. $P = P^m \cup P^{\mu}$ where P^m is the set of measurable places and P^{μ} is the set of non-measurable places.

Definition 5: The *l-length I/O language* of an IPN (Q,M_0) is defined as:

$$\pounds^l(Q, M_0) = \{\left[\frac{\lambda(t_{i+1})}{\varphi(M_{i+1})}\right], \left[\frac{\lambda(t_{i+2})}{\varphi(M_{i+2})}\right], \dots, \left[\frac{\lambda(t_{i+l})}{\varphi(M_{i+l})}\right] | M_i \xrightarrow{t_{i+1}} M_{i+1} \xrightarrow{t_{i+2}} M_{i+2} \dots M_{i+l-1} \xrightarrow{t_{i+l}} M_{i+l};\ M_i \in R(Q, M_0); i = 0,1,..\}.$$

III. IDENTIFICATON PROBLEM STATEMENT

Consider the following language definitions used in the statement and solution of the identification problem.

Definition 6: The set of *observed input/output* (I/O) *words* of a DES S with m inputs and n outputs is $\Gamma(S) = \{w_1, w_2, \dots\}$, such that $w_i = (\left[\frac{I_i(1)}{O_i(1)}\right], \left[\frac{I_i(2)}{O_i(2)}\right], \dots \left[\frac{I_i(|w_i|)}{O_i(|w_i|)}\right])$, where $\left[\frac{I_i(j)}{O_i(j)}\right]$ is the j-th observed input/output vector of size $m+n$ in sequence w_i and $|w_i|$ is the length of the I/O word w_i.

Definition 7: The *observed k-length I/O language* of a DES S is defined as $\mathcal{L}^k(S) = \{w_i(j+1)w_i(j+2) \dots w_i(j+l) | w_i \in \Gamma(S), j + l \leq |w_i| and\ l \leq k\}$.

Now the identification problem can be defined as follows: given a set of observed I/O words $\Gamma(S)$ generated by a real DES during its operation, the aim of the proposed method is to construct a safe IPN model (Q, M_0) such that $\pounds^{\kappa}(Q, M_0) = \mathcal{L}^{\kappa}(S)$; such a language approximates the actual behavior of the DES.

In our approach, $\mathcal{L}^{\kappa}(S)$ is computed for inferring the states of the system according to the parameter κ, which is used to adjust the accuracy of the identified IPN, similarly as proposed in [8].

IV. IDENTIFYING REAL AUTOMATED DES

The systems considered in this work are closed loop controlled DES (Figure 1); they consist of a plant and its industrial controller (in many cases a Programmable Logic Controller: PLC). The behavior of such systems (i.e. the PLC-plant compound system behavior) can be observed by collecting the signals exchanged between controller and plant.

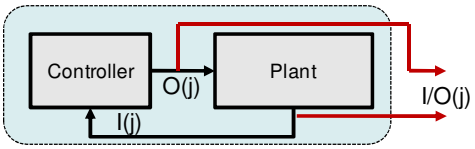

Figure 1. Closed loop controller-plant DES

Several phenomena, due to the interaction between plant and controller, increase the complexity of the identification process; however they must be taken into account when real controlled DES have to be identified:

- An input evolution (signal emitted by the plant through a sensor) does not always provoke an output evolution (signal emitted by the PLC to an actuator). In practice, few of input changes provoke output evolutions;
- Non simultaneous I/O events are often simultaneously observed;
- When output changes are provoked by input changes, this causal relationship is not necessarily captured simultaneously;

Now, we are going to explain these phenomena.

A. *PLC treatment cycle*

A PLC cyclically performs three main steps: "input reading" (I) where it reads the signals from the sensors, "program execution" (PEX) to determine the new outputs values for the actuators, and "output writing" (O) where the newly determined commands are sent to the plant actuators.

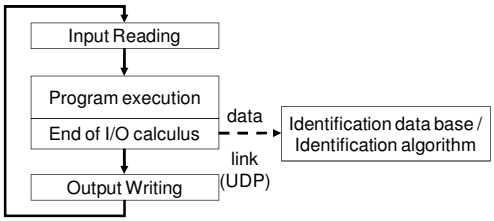

Figure 2. PLC cycle and data collection

At the end of the PEX phase the current values of inputs and outputs (I/O) are sent from the PLC to a computer and stored for a later treatment by the identification algorithm.

B. *Experimental constraints*

In this section, we use SFC only for precisely describe diverse behaviors. Recall that in the identification problem, the PLC program is assumed to be unknown.

Due to the PLC cycle, some situations between inputs and outputs could arise. Consider a situation described in Figure 3 (current active step is #10; *a* and *b* are two input signals to the PLC; A and B are two output signals).

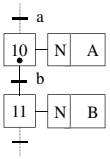

Figure 3. A single input is the condition for state evolution

Changes in the state and outputs will occur when signal *b* is active; however other input signals may evolve without consequence in the outputs. This must be considered in the identification algorithm.

Consider now the time diagram in Figure 4. Two signals are asynchronously emitted by sensors of the plant between two successive "input reading" phases (I) of the PLC cycle. These two signals will be simultaneously read during the next I phase and observed as simultaneous events in the identification data base. In DES theory events cannot occur simultaneously; so an observed event vector will therefore be defined as a change of value in an entry of an I/O vector.

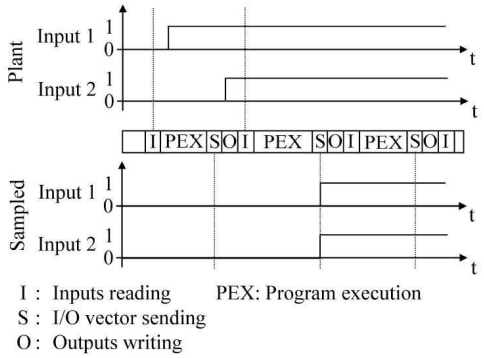

Figure 4. Apparent simultaneous evolution of several inputs

Now, let us consider the situation described in Figure 5(a). As shown in the time diagram in Figure 5(b), if input "b" changes its value from 0 to 1, the correspondent change in "B" is not provoked immediately, since it is necessary first a change in output "A". In this case cause and effect cannot be captured simultaneously but will be detected only if we observe a sequence of 4 consecutive events.

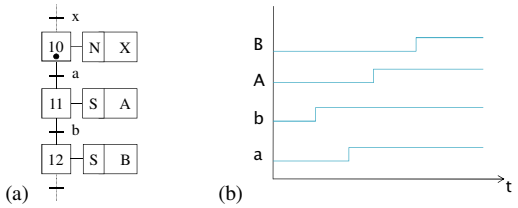

Figure 5. I/O causality and sequences of events

These three scenarios show that the implementation of a controller and its interaction with the plant introduces phenomena that must be taken into account by the identification algorithm.

Some approximations of the causality between inputs and outputs, allowing a more compact representation of the system's behavior can now be introduced. Consider the following I/O vector sequence σ involving one input *x* and two outputs *A*, *B*:

$$\begin{bmatrix} x \\ A \\ B \end{bmatrix} \qquad \sigma = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

This sequence can be represented as: $A \xrightarrow{x\_1} A \xrightarrow{\varepsilon} B$. Since such a behavior can be a consequence of the PLC cycle, it can be compacted as: $A \xrightarrow{x\_1} B$. Analogously, we can generalize: $A \xrightarrow{x\_1} A \xrightarrow{y\_1} A \xrightarrow{z\_0} A \xrightarrow{\varepsilon} B \cong A \xrightarrow{x\_1,y\_1,z\_0} B$.

V. IDENTIFICATION METHOD

A. *General strategy*

The identification procedure consists of several stages (Algorithm 1) that build systematically a safe IPN representing exactly the sampled output language of length κ+1 of the DES.

Algorithm 1. Global identification procedure

Inputs: A DES and an accuracy parameter κ
Output: (Q,M_0): an IPN model

1. Obtain the cyclic sequences w_i of observed I/O vectors.
2. Compute event vector sequences τ_i and symbolic input events λ' from the observed vectors.
3. For every sequence of event vectors τ_i, create event vector traces τ_i^{κ} of length κ.
4. Create the non-observable behavior of the IPN and simplify it.
5. Complete the IPN adding the observed behavior and deleting implicit places.

The stages of this procedure are described below.

B. *Sample processing*

B.1 Event sequences

As stated before, the data obtained from the system to be identified is a set of sequences of I/O vectors w_1, w_2, ..., such that $w_i = (w_i(1), w_i(2), \dots)$, where $w_i(j)$ refers to the j-th

observed vector in sequence w_i. Such sequences may have different length. From these sequences, strings of observed event vectors are first computed.

Definition 8: An observed *event vector* $\tau_i(j)$ is the variation between two consecutive I/O vectors $w_i(j)$, $w_i(j+1)$; it is computed as $\tau_i(j) = w_i(j+1) - w_i(j)$. An *input event vector* $\beta(\tau_i(j))$ is the variation between two consecutive input vectors $I_i(j)$, $I_i(j+1)$; it is computed as $\beta(\tau_i(j)) = I_i(j+1) - I_i(j)$. A *symbolic input event* $\lambda'(\tau_i(j))$ is a string representation of the input event vector $\beta(\tau_i(j))$; it is computed as:

$$\lambda'(\tau_i(j)) = \prod_{l=1}^{m} \begin{cases} I_l\_1 & if\ I_i(j+1)_l - I_i(j)_l = 1 \\ I_l\_0 & if\ I_i(j+1)_l - I_i(j)_l = -1 \\ \varepsilon & if\ I_i(j+1)_l - I_i(j)_l = 0 \end{cases}$$

Then for every sequence w_i, a sequence of observed event vectors $\tau_i = \tau_i(1)\tau_i(2)\dots\tau_i(|\tau_i|)$ is obtained. The maximum number of possible event vectors is $3^{(m+n)} - 1$. However, in practice, only a small subset of them is observed.

Example 1. Consider a DES with n = 4 output signals, Φ = {A, B, C, D}, and m = 3 input signals Σ = {a, b, c}. Three I/O sequences have been observed; vector entries correspond to distribution [a, b, c | A, B, C, D]^T

$$w_1 = \left(\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\1\\0\end{bmatrix}\right), w_2 = \left(\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\\0\\0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}\right), \quad w_3 = \left(\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}\right)$$

According to definition 8, for every sequence w_i, a sequence of observed event vectors $\tau_i = \tau_i(1)\tau_i(2)\dots\tau_i(|\tau_i|)$ is computed. During the process, if the difference has not been observed before, a new event vector e_j is created and stored ($\tau_i(j) = e_j$).

Now, for the example 1, the sequences τ_i of the detected event vectors e_j associated to I/O changes are obtained:

$$w_1 = \left(\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}\overset{e_1}{\rightarrow}\\1\\0\\0\\-1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}\overset{e_2}{\rightarrow}\\0\\0\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix}\right), \tau_1 = e_1e_2$$

$$\lambda'(e_1) = a\_1, \lambda'(e_2) = \varepsilon$$

$$w_2 = \left(\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}\overset{e_1}{\rightarrow}\\1\\0\\0\\-1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}\overset{e_3}{\rightarrow}\\0\\1\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}\overset{e_4}{\rightarrow}\\0\\-1\\0\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\\0\\0\\1\\1\end{bmatrix}, \begin{bmatrix}\overset{e_5}{\rightarrow}\\0\\0\\-1\\0\\0\\-1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}\overset{e_6}{\rightarrow}\\-1\\0\\0\\1\\0\\-1\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}\right), \tau_2 = e_1e_3e_4e_5e_6$$

$$\lambda'(e_3) = b\_1\ c\_1, \lambda'(e_4) = b\_0, \lambda'(e_5) = c\_0, \lambda'(e_6) = a\_0$$

$$w_3 = \left(\begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}\overset{e_1}{\rightarrow}\\1\\0\\0\\-1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}\overset{e_3}{\rightarrow}\\0\\1\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}\overset{e_5}{\rightarrow}\\0\\0\\-1\\0\\0\\-1\end{bmatrix}, \begin{bmatrix}1\\1\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}\overset{e_4}{\rightarrow}\\0\\-1\\0\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}\overset{e_6}{\rightarrow}\\-1\\0\\0\\1\\0\\-1\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\\0\\0\\0\end{bmatrix}\right), \tau_3 = e_1e_3e_5e_4e_6$$

B.2 Sequences of κ-length event vector traces

From every sequence $\tau_i = \tau_i(1)\tau_i(2)\dots\tau_i(|\tau_i|)$ we compute sequences of event vector traces $\tau_i^\kappa = \tau_i^\kappa(1), \tau_i^\kappa(2), \dots \tau_i^\kappa(|\tau_i|)$ such that every $\tau_i^\kappa(j)$ is the κ -length substring of τ_i that finishes with $\tau_i(j)$. For the first $\kappa - 1$ elements of the trace sequence the event vector ε (zero vector) is used. Such traces are used to determine equivalent states, according to the following equivalence notion.

Definition 9: Two states of the identified system are κ-equivalent if their I/O vectors are the same and if the κ last observed event vectors that lead to these states are the same.

Following with the previous example, the sequences of traces using $\kappa = 2$ are:

$\tau_1^2 = \varepsilon e_1, e_1e_2$ for τ_1
$\tau_2^2 = \varepsilon e_1, e_1e_3, e_3e_4, e_4e_5, e_5e_6$ for τ_2
$\tau_3^2 = \varepsilon e_1, e_1e_3, e_3e_5, e_5e_4, e_4e_6$ for τ_3

C. Building the basic structure

C.1 Representing event traces

Once the sequences of event vector traces have been obtained, every trace $\tau_i^\kappa(j)$ is related to a transition in the IPN through a function γ:T→$\{\tau_i^\kappa(j)\}$; the firing of a transition implies that κ consecutive event vectors related to such a transition have been observed.

In order to preserve firing order between transitions, dependencies are created between them and associated with an observed marking through the function $\mu: P^u \rightarrow \{\varphi M_i | M_i \in R(N, M_0)\}$, which relates every non-measurable place with an observed marking, such that every transition has only one input place and one output place ($\forall t_r \in T, |\ {}^\bullet t_r| = |t_r^\bullet| = 1$). Notice that the number of non-measurable places is not predefined. When an event vector trace $\tau_i^\kappa(j)$ is found again in a τ_i^κ the associated dependency must be used if it leads to the same observed marking.

Let e_j be the last event vector in the trace $\tau_i^\kappa(j)$; the associated transition will be denoted as $t_r^{e_j}$ (more than one transition may have associated the same e_j). This strategy can be systematically performed following the next procedure which is an adaptation from that included in [11].

Algorithm 2. Building the basic IPN structure

Input: The set $\Gamma^\kappa = \{\tau_i^\kappa\}$
Output: A PN structure G composed by p∈P^u

1. $T \leftarrow \emptyset; ET \leftarrow \emptyset;$
   $P \leftarrow \{p_{ini}\}; M_0(p_{ini}) \leftarrow 1; \mu(p_{ini}) \leftarrow \tau_i(1);$
2 $\forall \tau_i^\kappa \in \Gamma^\kappa$
  2.1 $current \leftarrow p_{ini};$
  2.2 $\forall \tau_i^\kappa(j) \in \tau_i^\kappa, 1 \le j \le |\tau_i^\kappa|$
    2.2.1 If $\tau_i^\kappa(j) \notin ET$
      then
        $ET \leftarrow ET \cup \{\tau_i^\kappa(j)\};$
        $T \leftarrow T \cup \{t_r^{e_j}\}; \gamma(t_r^{e_j}) \leftarrow \tau_i^\kappa(j);$
        $\forall p_a \in P$
          $I(p_a, t_r^{e_j}) \leftarrow 0; O(p_a, t_r^{e_j}) \leftarrow 0;$
        $I(current, t_r^{e_j}) \leftarrow 1;$
        If $j = |\tau_i^\kappa|$ and $\mu(current) = \mu(p_{ini})$

then $O(p_{ini}, t_r^{e_j}) \leftarrow 1$;
else $P \leftarrow P \cup \{p_{out}\}$;
$\forall t_b \in T$
$I(p_{out}, t_b) \leftarrow 0; (p_{out}, t_b) \leftarrow 0;$
$\mu(p_{out}) \leftarrow \mu(current) + e_j;\ O(p_{out}, t_r^{e_j}) \leftarrow 1;$
$current \leftarrow p_{out};$
2.2.2 If $\tau_i^{\kappa}(j) \in ET$
then
If $\exists p_{in} |\ p_{in} = {}^{\bullet}t_r^{e_j}, t_r^{e_j} \in T, \gamma(t_r^{e_j}) = \tau_j^{\kappa}(j)$ *and*
$\mu(p_{in}) = \mu(current)$
then $merge(current, p_{in});\ current \leftarrow (t_r^{e_j})^{\bullet}$;
else go to step 2.2.1, and take $\tau_i^{\kappa}(j) \notin ET$.
If $j = |\tau_i^{\kappa}|$ and $\mu(current) = \mu(p_{ini})$
then $merge(current, p_{ini})$;

Since search operation has linear complexity and the algorithm implies the addition of a transition for every computed trace that has not been yet observed, then Algorithm 2 is executed in polynomial time on the number of observed output sequences and their maximum length .

Proposition 1. The IPN G built through algorithm 2 represents all and only all the trace sequences (sub-sequences) in Γ^{κ}.

The proof is similar to that included in [11].

Using the previous algorithm, the obtained PN corresponding to the three sequences of event vector traces of the example 1 is shown in Figure 6. Notice that one of the sequences is not cyclic.

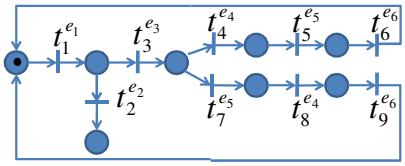

Figure 6. Basic model with sequences of event vector traces

C.2 Simplifying the basic structure

Additional node merging operations can be performed on the basic structure in order to obtain an equivalent trace model. Now we can take into account the event vector e_j associated to transitions. Consider the following transformation rules.

Algorithm 3. Simplifying the basic structure.

Input: G
Output: G': an equivalent IPN

Apply the following rule on the initial place and iteratively on the merged places
Rule 1: $\forall t_a^{e_j}, t_b^{e_j} \in p_{ini}{}^{\bullet} | a \neq b$
$merge(t_a^{e_j} t_b^{e_j}) merge(t_a^{e_j\bullet}, t_b^{e_j\bullet})$
Rule 2: $\forall t_a^{e_j}, t_b^{e_j} \in {}^{\bullet}p_{ini} | a \neq b$
$merge(t_a^{e_j} t_b^{e_j}) merge({}^{\bullet}t_a^{e_j}, {}^{\bullet}t_b^{e_j})$

Proposition 2. G' preserves the trace sequences in G.

The proof for is similar to that included in [11].

In the example, the application of the rules leads to the model of Figure 7(a). Since the initial place has two input transitions associated to e_6, they can merge and also their input places.

C.3 Concurrent transitions

Other transformations may be performed when there are transitions that appear in the sequences in different order describing their interleaved firing; this behavior is exhibited by concurrent transitions. The analysis can be performed on a model component comprised between two transitions relied by several paths containing the concurrent transitions. If there are *m!* paths, we can explore if there exists *m* different transitions in the paths and every sequence is a permutation from each other. When it is verified, the subnet can be transformed into a concurrent component of G' preserving the same behavior.

In Figure 7(a), notice that between the transition associated to e_3 and the new transition associated to e_6 there are paths with all possible permutations of e_4 and e_5; then, we can transform this into a concurrent component and we obtain the net showed in Figure 7(b).

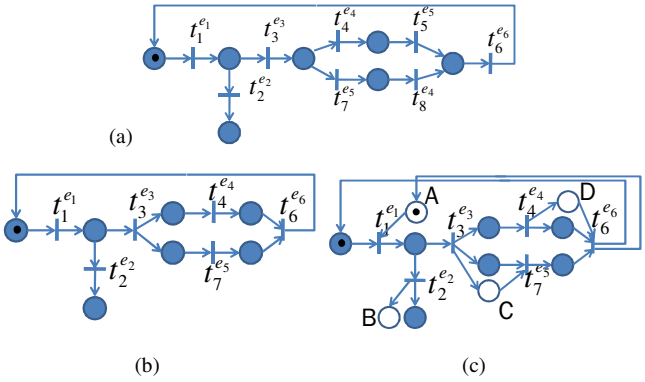

Figure 7. (a) Model after merging (b) Simplified basic model (c) IPN model including measurable places

Notice that this model preserves the same event vector sequences of the previous one.

The simplification by analysis of concurrency is not strictly necessary for representing the event vector sequences; however the equivalent model with concurrent transitions may be simpler. Although the analysis could be inefficient when the number of paths in the subnet is large, usually this number is reduced.

The aim of the simplification strategies given above is obtaining fairly reduced models useful for analyzing the DES behavior, rather than minimizing the number of nodes in the obtained models.

D. Adding interpretation to the PN model

D.1Representing outputs changes

Once the event vector sequences are represented in the basic model, it must be completed by adding the output and input changes. Recall that event vectors are computed from the difference of consecutive I/O vectors; thus an e_j relates measurable places representing the outputs changes. This

procedure is detailed in [11]. The number of observable places is then equal to the number of outputs. The net with measurable places for the example is showed in Figure 7(c).

D.2 Model simplifying

Implicit non-measurable places can be removed; if there is a non-measurable place p_k whose input and output transitions are exactly the same than any measurable place, then delete p_k and its input and output arcs.

D.3 Representing input changes

Once the output adding and implicit places deleting has been performed, it only remains to add input information to complete the IPN model. Input information is associated with labels for transitions in a natural way given by the symbolic event input function. Algorithm 4 describes a systematic way to do it.

Algorithm 4. Representing input changes

Input: G', $\lambda'(e_j)$

Output; Q : the final model of the identification process

Step 1. $\forall t_r^{e_j} \in T$, $\lambda(t_r^{e_j}) \leftarrow \lambda'(e_j)$ //Associate to each $t_r^{e_j}$ the symbolic input event registered at the detection of e_j.

The final model for the illustrative example is showed in Figure 8. The associated inputs for transitions are given by: $\lambda'(e_1)$=a_1, $\lambda'(e_2)$= ε, $\lambda'(e_3)$=b_1 c_1, $\lambda'(e_4)$=b_0, $\lambda'(e_5)$=c_0, $\lambda'(e_6)$=a_0.

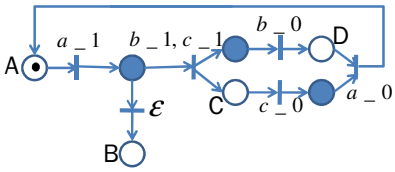

Figure 8. Simplified IPN model

Since every one of the transitions in the net actually represents a sequence of event vectors of length κ, the I/O language of length $\kappa + 1$ of the net is equal to the observed I/O language. Even, for the above example, the I/O language of the IPN is equal to the observed I/O language, i.e. only the observed cyclic I/O sequences are represented by the evolution of the net.

Proposition 3. For a DES S and an identification parameter κ, Algorithm 1 yields an IPN model (Q,M_0) which represents exactly $\mathcal{L}^{\kappa+1}(S)$.

Proof. Since the deletion of implicit places does not change $\mathcal{L}^{\kappa}(Q,M_0)$, the proof is made with the model obtained before this procedure. The firing of a transition t in the system is not affected by the addition of arcs to and from t, since these arcs have been computed from differences of vectors in $\Gamma(S)$. Then, also in this model, every event vector sequence σ of length less or equal than κ belongs to the language of the net iff it was observed.

The sequences of transitions of length less or equal than κ that can be fired, lead to markings in the measurable places that also belong to $\Gamma(S)$ (since the marking change provoked in the measurable places was obtained from the difference of observed vectors). Then, we have that sequences of observed output vectors of length less or equal than $\kappa + 1$ correspond to sequences of marking vectors in the net and $\mathcal{L}^{\kappa+1}(Q) = \mathcal{L}^{\kappa+1}(S)$.■

VI. CASE STUDY

Based on the algorithms presented in this paper, a software tool has been developed to automate the IPN model synthesis. Many examples on diverse size and complexity have been tested. Below we present a small size case study from [9] whose layout is showed in Figure 9. The function of this system is to sort parcels according to their size. It has 9 inputs (signal sensors from the system): a_0,a_1, a_2,b_0,b_1,c_0,c_1,k_1,k_2, and 4 outputs (signals to the actuators): A+,A-,B,C.

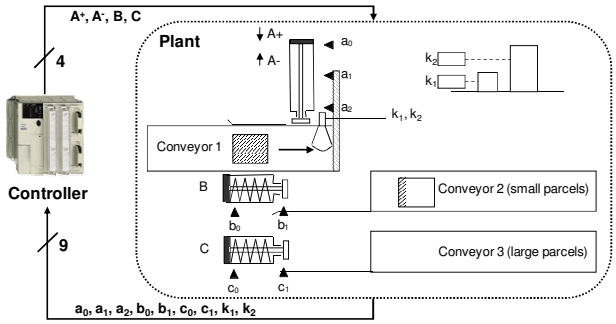

Figure 9. Manufacturing system layout

Due to the lack of room, we only consider nine observed cycles of production (a cycle consist in the arrival of a parcel and its sorting) given below by the nine I/O observed sequences of different length. The vector entries correspond to the following distribution: [A+ A- B C k_1 k_2 a_0 a_1 a_2 b_0 b_1 c_0 c_1].

After the identification process and before concurrence transformations, the model shown in Figure 10(b) is obtained. After some transformations considering phenomena described in section IV, the net in Figure 11 is obtained. These models have been automatically obtained using the developed software tool.

From the obtained figure, we can infer the following behavior:

- When a large parcel arrives (k1 and k2 rise), cylinder A is extended (A+ rise). After cylinder A pushes the parcel to the conveyor 3 (a0 falls, k1 falls, k2 falls, a1 rises, a1 falls and a2 rises), cylinder C extends (C rises) and cylinder A retracts to its initial position (A- falls). When parcel is completely pushed, cylinders retract to their initial position.

- When a short parcel arrives (only k1 rise), cylinder A is extended (A+ rise) until conveyor 2 is reached (a0 falls, k1 falls and a1 rises). Then, cylinder B extends (B rises) and cylinder A retracts (A- falls). When parcel is completely pushed, cylinder B retracts until its initial position.

This interpretation is valid only for the part of the behavior we have observed. If we consider that data collection has been made for a long time, we can state that the model is almost exact.

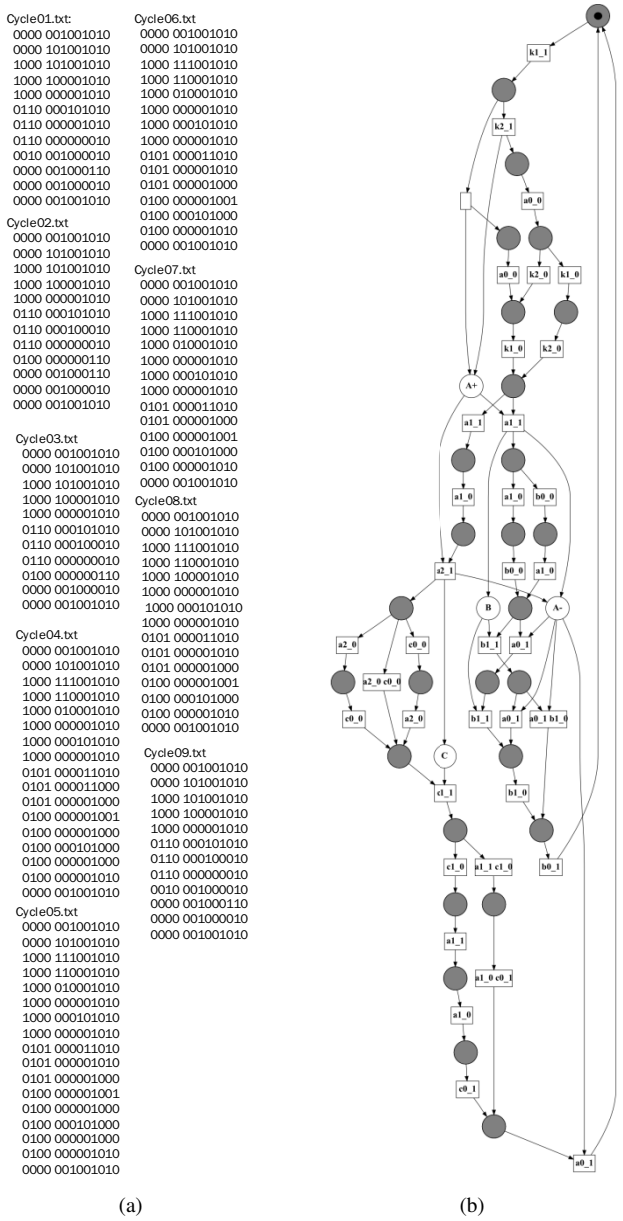

Figure 10. (a) Cyclic sequences (b) identified IPN for the case study

VII. CONCLUSIONS

Identification of automated concurrent Discrete Event Systems (DES) has been addressed. A previous synthesis method has been extended and adapted, allowing the identification of closed-loop PLC-based controlled systems from vector sequences including both inputs and outputs. In order to cope with technological issues regarding the automation based on PLC, simplifying transformations on the obtained model have been proposed, allowing a more compact representation of the system behavior.

Current research deals with the inference of cyclic sequences from a single sequence and consideration of a-priori known relations among DES components for identification.

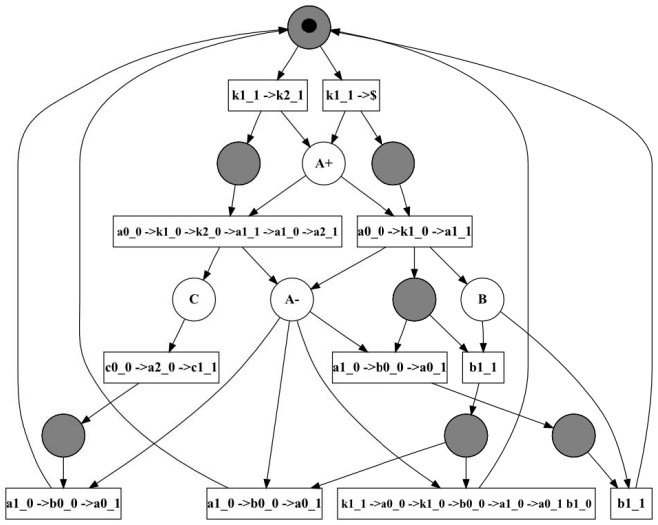

Figure 11. Reduced model for the case study

ACKNOWLEDGEMENT

A.P. Estrada-Vargas was sponsored by CONACYT (Mexico) Grant No. 50312, and by Région Île-de-France.

REFERENCES

[1] Estrada-Vargas, A.P., E. López-Mellado, J.J. Lesage. "A Comparative Analysis of Recent Identification Approaches for Discrete-Event Systems", Mathematical Problems in Engineering Volume 2010, Hindawi. doi:10.1155/2010/453254

[2] M. Meda, E. López, "A passive method for on-line identification of discrete event systems", Proc. of the IEEE Int. Conf. on Decision and Control, Orlando, Florida, USA. pp. 4990-4995, Dec 2001

[3] M. Meda, E. López, "Required Transition Sequences for DES identification", Proc. of the IEEE Conference on Decision & Control (CDC 2003), Maui, Hawaii USA. pp. 3778-3782, Dec 2003

[4] A. Giua and C. Seatzu, "Identification of free-labeled Petri nets via integer programming", Proc. of the 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, Seville, Spain, Dec 2005

[5] M. P. Cabasino, A. Giua, C. Seatzu, "Identification of unbounded Petri nets from their coverability graph", Proc. of the 45th IEEE Conference on Decision & Control, San Diego, CA, USA, Dec 2006

[6] M. Dotoli, M. P. Fanti, A. M. Mangini, "Real time identification of discrete event systems using Petri nets", Automatica, Vol. 44, No. 5, pp 1209-1219, May 2008

[7] M. P. Fanti and C. Seatzu, "Fault diagnosis and identification of discrete event systems using Petri nets", Proc. of the 9th International Workshop on Discrete Event Systems, Göteborg, Sweden, pp. 432-435, May, 2008

[8] S. Klein, L. Litz, J.-J. Lesage, "Fault detection of Discrete Event Systems using an identification approach", 16th IFAC World Congress, CDROM paper n°02643, Praha (Czech Republic), July 2005

[9] M. Roth, J.-J. Lesage, L. Litz, "An FDI Method for Manufacturing Systems Based on an Identified Model", Proc. of IFAC Symposium on Information Control Problems in Manufacturing (INCOM 2009), Moscow, Russia, pp. 1389-1394, June 2009

[10] M. Roth, J.-J. Lesage, L. Litz, "Black-box identification of discrete event systems with optimal partitioning of concurrent subsystems", Proc. of the American Control Conference (ACC 2010), Baltimore, Maryland, USA, pp. 2601-2606, June 2010

[11] A.P. Estrada-Vargas, E. López-Mellado, J.J. Lesage. "Off-line Identification of Concurrent Discrete Event Systems Exhibiting Cyclic Behavior". Proc. of IEEE Int. Conf. on Systems Man and Cybernetics, San Antonio Tx, USA, pp.181-186, Oct 2009