Input-output identification of controlled discrete manufacturing ystems

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

INPUT-OUTPUT IDENTIFICATION OF CONTROLLED DISCRETE MANUFACTURING SYSTEMS

Ana Paula Estrada-Vargas[1,2], Ernesto López-Mellado[1], Jean-Jacques Lesage[2]

[1] *CINVESTAV Unidad Guadalajara. Av. Del Bosque 1145, Col. El Bajío, 45019 Zapopan, Mexico*

[2] *LURPA Ecole Normale Supérieure de Cachan. 61, av du Président Wilson, 94235 Cachan Cedex, France.*

ABSTRACT

The automated construction of discrete event models from observations of external system's behaviour is addressed. This problem, often referred to as system identification, allows obtaining models of ill-known (or even unknown) systems. In this article an identification method for Discrete Event Systems (DESs) controlled by a Programmable Logic Controller is presented. The method allows processing a large quantity of observed long sequences of input/output signals generated by the controller and yields an interpreted Petri net model describing the closed-loop behaviour of the automated DESs. The proposed technique allows the identification of actual complex systems because it is sufficiently efficient and well adapted to cope with both the technological characteristics of industrial controllers and data collection requirements. Based on polynomial-time algorithms, the method is implemented as an efficient software tool which constructs and draws the model automatically; an overview of this tool is given through a case study dealing with an automated manufacturing system.

Keywords: Automated Modelling, Controlled DES, I-O Identification, Interpreted Petri Nets.

1. Introduction

1.1 Industrial systems identification

In automation engineering, the first step of the development life cycle of new systems is generally devoted to the modelling of the system's behaviour. In most cases these models are used for functional or performance evaluation by analytical approaches or simulation techniques. In this paper, the problem we address is the model discovery of systems whose models are not updated or are not available at all. We will see later that this situation occurs frequently in industry. Nevertheless, despite the lack of a "reliable" model, performance evaluation or redesign is often required.

Modelling existing systems (natural or manmade) for performance assessment has been done in several fields:

- Natural systems: in [Chang and Hanna, 2004] a model for evaluating the quality of air is proposed,
- Organizations: Hospital activity management is analysed through workflow models [Du, 2009],
- Energy production [Connolly, 2010] and industrial systems [Mottershead, 1993] may require more detailed and complete models for simulation.

In industrial systems under intensive workload, initial models are not always updated. In [Lanubile, 2002], the authors explain that a typical approach to software maintenance is analysing just the source code, applying some patches, releasing the new version, and then updating the documentation. This quick-fix approach usually leads to documentation that is not aligned with the current system and degrades the original system structure, thus rendering the evolution of the system costly and error-prone.

This last case is representative of automated discrete production systems in which Programmable Logic Controller (PLC) code maintenance is often performed without updating the documentation nor the control algorithm or model. For such systems, identification from observations of external system's behaviour could be more appropriate, since the system is operating most of the time. In this case the main advantage of identification is that the obtained model approximates closely the actual system functioning. An identified model can be afterwards completed by applying well known modelling techniques using the knowledge about the process, for analysis or redesign purposes.

1.2 Identification methods

1.2.1 Languages inference techniques

Pioneer works on identification have been developed in computer science, where the problem of obtaining a language representation from sets of accepted words has been dealt since a long time. Such methods are generally referred as *languages inference techniques* or *learning techniques*.

Gold´s method [Gold, 1967] processes positive samples: an infinite sequence of examples such that the sequences contain all and only all the strings of the language to learn.

The Probably Approximately Correct (PAC) learning technique proposed in [Valiant, 1984] learns from random examples and studies the effect of noise on learning from queries.

The query learning model proposed in [Angluin, 1988] considers a learning protocol based on a "minimally adequate teacher"; this teacher can answer two types of queries: membership query and equivalence query.

Several works that have adopted state machines as representation model, allow describing the observed behaviour. In [Booth, 1967] a method to model a language as Moore or Mealy machines is presented. The system under investigation is placed within a test bed and connected to a so called experimenter, which generates the input signals and records the output signals of the system. The identification can be started considering a very few number of states. If, at some point of the experiment, it is impossible to find a correct machine with the assumed number of states, the identification is started again considering a machine with one more state.

The method proposed in [Kella, 1971] allows obtaining models representing Mealy machines from a single observed input-output sequence. The algorithm lists all reduced machines which may produce the given sequence. The construction principle is the merging of equivalent states.

In [Biermann and Feldman, 1972] a method for the identification of non deterministic Moore machines based on a set of input output sequences is presented. All the sequences start in the same initial state. The identification principle is the reduction of an initial machine represented as a tree.

The method presented in [Veelenturf, 1978] processes simultaneously a sample of sequences to produce stepwise convergent series of Mealy machines, such that the behaviour of every new machine includes the behaviour of the previous one. At each step, the last obtained machine is analysed and completed by adding transitions and possibly new states.

Later, in [Veelenturf, 1981] an algorithm to identify a unique Moore machine generating the behaviour observed during *m* sequences starting at the same initial state is proposed. The learning procedure operates in three steps: induction, contradiction, and discrimination. A state can never be deleted and only transitions between states can be modified. This method is improved in [Richetin, 1984], which proposes two algorithms to identify multiple systems as well as systems that may not be initialized between two records.

The identification problem for context free grammars (CFGs) needs, beside given examples, some additional structural information for the inference algorithm [Levy and Joshi, 1978].

[Ishizaka, 1990] has investigated a subclass of CFGs called simple deterministic grammars. A polynomial time algorithm that allows an exact identification of a simple deterministic language is given.

In [Takada, 1998] it has been shown that the grammatical inference problem for even linear languages can be reduced in polynomial time to the inference of regular languages.

Other works use as description formalism Petri net models. In [Hiraishi, 1992] an algorithm for synthesising Petri net models is presented. The proposed algorithm has two phases. In the first phase, the language of the target system is identified under the form of a DFA. In the second phase, a Petri net that accepts the same language as the DFA is built.

1.2.2 Recent DESs identification methods

In recent years, the scientific community has proposed identification approaches (based on Petri net or automata) for obtaining approximated models of DESs whose behaviour is unknown or ill-known. In the context of automated DESs, identification methods can be complementary to established modelling techniques; identification builds a closed-loop controller-plant model, which is more classically obtained by a composition of models of controller and plant. Three main approaches for identifying DESs have been proposed in literature [Estrada, 2010a].

The incremental synthesis approach, proposed in [Meda, 2000], [Meda, 2001], [Meda, 2003], deals with unknown partially measurable concurrent DESs exhibiting cyclic behaviour. 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 [Giua, 2005] a method to build a free labelled 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. Both the ILP statement and its solution are computational demanding. This approach has been extended to other PN classes [Cabasino, 2007], [Dotoli, 2008], [Fanti, 2008] [Dotoli, 2011]; however, issues regarding applications to actual industrial DESs have not yet been addressed in these works.

Another recent off-line method [Klein, 2005] allows building a non-deterministic finite automaton (FA) from a set of input/output (I/O) sequences, experimentally measured from the DES to be identified. Under several hypotheses, the constructed FA generates exactly the same I/O sequences of given length than observed ones. The method was conceived for fault detection in a model-based approach [Roth, 2012]. Extensions to this work propose an identification method performing optimal partitioning of concurrent subsystems for distributed fault detection purposes [Roth, 2010].

Other works on the matter, based on different approaches have been proposed. The techniques for workflow mining, published by van der Aalst and co-workers [Cook, 2004][van der Aalst, 2004], allows building Petri net models of workflow processes in which all the activities are observable. Other works pursue the construction of a stochastic PN from recorded event sequences [Ould El Medhi, 2006] [Ould El Medhi, 2012] for reliability analysis.

1.3 Approach

Our approach is oriented towards the identification of actual industrial automated DESs, such as discrete manufacturing systems in which concurrent repetitive tasks are performed. We focus on closed loop controlled systems whose only available knowledge is the observed behaviour in the form of many long cyclic sequences of input-output vectors measured from a PLC. The aim of this research is to define an efficient identification method, able to build a comprehensible IPN model that describes closely the actual behaviour of the system.

In a first paper we proposed a method for synthesizing interpreted PN (IPN) for coping with concurrent partially observable DESs [Estrada, 2009]; it processes a set of cyclic sequences of binary output signals yielding models including silent transitions and non-labelled places. Afterwards the method has been extended for dealing with sequences of I/O signals captured during the closed-loop operation of PLC-based controlled DESs [Estrada, 2010b]. Fundamental 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 behaviour, including that non observable directly from the PLC. The present paper gathers and details the results of these papers; it presents a global and coherent view of these results, includes a wider literature review, and extends them with important application issues. For that, we describe an operational software tool which implements the proposed identification method, which builds and draws by building and drawing the IPN model automatically.

The paper is organized as follows. In section 2 the background on Petri nets and languages is summarised. Based on these definitions the problem of DESs identification is stated in section 3 in terms of language associated to an IPN. Several constraints inherent to

real controlled DESs are analysed in section 4. The identification algorithm is given in section 5. Then, a software tool implementing the proposed method is outlined and a case study is included in section 6.

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

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 $\forall p_i \in P$, $M_k(p_i) \geqslant I(p_i, t_j)$; 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 $v_k(i) = 0$, $i \neq j$, $v_k(j) = 1$, this equation is called the PN state equation. The reachability set of a PN is the set of all possible reachable markings from M_0 firing only enabled transitions; this set is denoted by $R(G, M_0)$. A PN is called *safe* (or 1-bounded) 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 where:

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.

$\lambda : T \rightarrow g(\Sigma) \cup \{\varepsilon\}$ is a labeling function of transitions, where $g(\Sigma)$ is a conjunction of input changes (rising and falling edges), ε represents a system internal event externally uncontrollable.

$\varphi : R(Q, M_0) \rightarrow (Z^+)^q$ is an output function, that associates to each marking in $R(Q, M_0)$ a q-entry output vector; $q = |\Phi|$ is the number of outputs. φ is represented by a $q \times |P|$ matrix, such that if the output symbol ϕ_i is present (turned on) every time that $M(p_j) \geqslant 1$, 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 behaviour 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. $P = P^m \cup P^u$ where P^m is the set of measurable places and P^u is the set of non-measurable places.

Definition 5: The *l-length I/O language* $\mathcal{L}^l(Q, M_0)$ of an IPN (Q, M_0) contains words of length l formed by pairs $(\lambda(t_k), y_k)$, where $y_k = \varphi M_k$.

$$\mathcal{L}^l(Q, M_0) = \{(\lambda(t_{i+1}),\varphi(M_{i+1}))(\lambda(t_{i+2}),\varphi(M_{i+2}))\ldots(\lambda(t_{i+l}),\varphi(M_{i+l})) \mid M_i \xrightarrow{t_{i+1}} M_{i+1} \text{ and } M_i \in R(Q,M_0)\}$$

3. Controlled discrete event systems

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

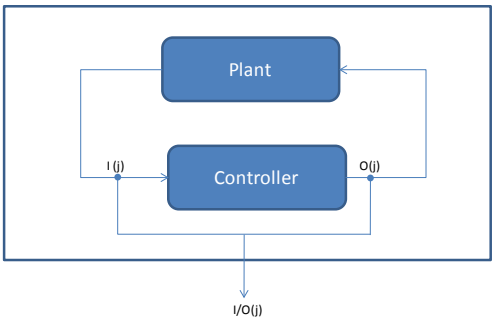

Figure 1. Closed loop controller-plant DES

Several phenomena, due to the interaction between plant and controller, increase the complexity of the identification process; they must be taken into account when real controlled DESs 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 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.

3.1 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 [Bel Mokadem, 2010].

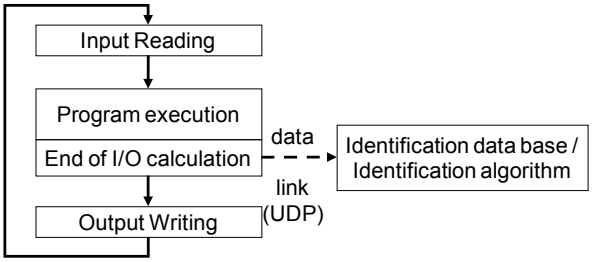

Figure 2. PLC cycle and data collection

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. In this paper we consider that the PLC operates *cycle driven*, i.e. the cycle is executed periodically [Lohmann, 2007].

3.2 Experimental constraints

In the identification problem we are addressing, the PLC program is assumed to be unknown. Sequential Function Charts (SFC) are used in this paper only for describing the diverse situations addressed by the proposed method.

SFC is a graphical programming language used for PLCs. It is one of the five languages defined by IEC 61131-3 standard. Main components of SFC are: *Steps* with associated actions, *Transitions* with associated logic conditions, *Directed links* between steps and transitions. Steps in an SFC diagram can be active or inactive. Steps are activated when all steps above it are active and the connecting transition is validated (i.e. its associated condition is true). When a transition is fired, all steps above are deactivated and simultaneously all steps below are activated. Actions associated with steps can be of several types, the most relevant ones being Continuous (N), Set (S) and Reset (R). Apart from the obvious meaning of Set and Reset, an N action ensures that its target variable is set to 1 as long as the step is active.

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).

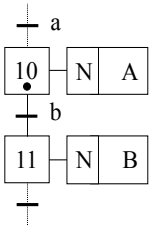

Figure 3. A single input is the condition for state evolution

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; such signals will be read during the next I phase and observed as a simultaneous change in the corresponding event vector in the identification data base; in general several input/output changes may be represented in an event vector. Therefore the events are defined when any change of value in at least one entry is detected between two consecutive I/O vectors.

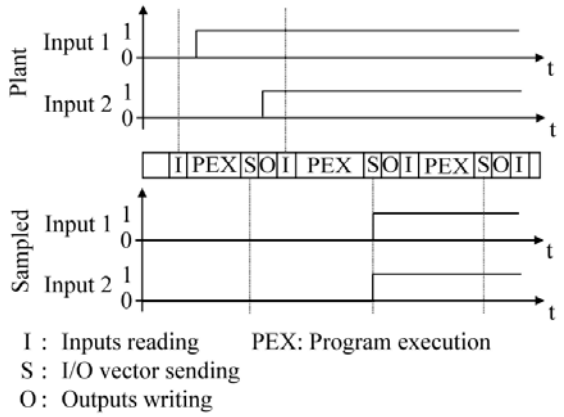

Figure 4. Apparent simultaneous evolution of several inputs

Let us now consider the specification described in Figure 5(a) in which the effect of input b is considered when the step 11 is active. Now consider the situation shown in the time diagram in Figure 5(b) in which input *b* changes its value from 0 to 1 before the change in *a* and then in *A*; after that *b* is taken into account to produce B. In this case the relationship cause-effect specified to activate step 11 cannot be captured simultaneously; however it will be detected only if we observe a sequence of 4 consecutive events.

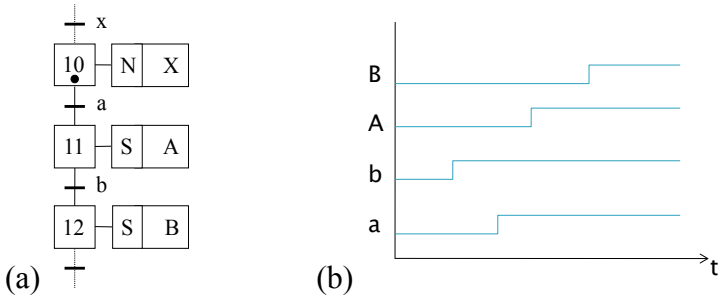

Figure 5. I/O causality and sequences of events

These three specific scenarios, among numerous possible others, 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.

4. 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 *r* inputs and *q* outputs is $\Gamma(S) = \{w_1, w_2, ...\}$, such that $w_i = \left[\frac{I_i(1)}{O_i(1)}\right]\left[\frac{I_i(2)}{O_i(2)}\right],...,\left[\frac{I_i(|w_i|)}{O_i(|w_i|)}\right]$, where $\left[\frac{I_i(j)}{O_i(j)}\right]$ is the *j*-th observed I/O vector of size $r + q$ 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 $\mathcal{L}^{\kappa}(S) = \{w_i(j+1)w_i(j+2)...w_i(j+l) \mid w_i \in \Gamma(S), j+l \le |w_i|, l \le k\}$.

Now the identification problem can be defined as follows: *given a set of observed I/O words Γ(S) generated by a real DES during its operation, the aim of our identification approach is to construct a safe (or 1-bounded) IPN model (Q, M_0) such that $\mathcal{L}^{\kappa}(Q, M_0) = \mathcal{L}^{\kappa}(S)$.*

Since Γ(*S*) does not provide any information related to the state evolution of the observed DES, these states will be inferred in the identified IPN from $\mathcal{L}^{\kappa}(S)$, where each word of length κ is part of the history (of length κ) of the system. The I/O language generated by the identified IPN is therefore inevitably an approximation of the actual behaviour of the DES. In our approach, the parameter κ is used to adjust the accuracy of the identified model, similarly as proposed in [Klein, 2005].

5. Identification algorithm

The identification method consists of several steps (Algorithm 1) that build systematically a safe IPN representing exactly the sampled output language of length κ+1 of the DES from the observed I/O vectors sequence Γ(S).

Algorithm 1. Global identification procedure

Inputs: Γ(S) and the parameter κ
Output: (Q,M_0): an IPN model

---

1. Compute event vector sequences τ_i and symbolic input events λ' from the observed vectors Γ(S).
2. For every sequence of event vectors τ_i, create event vector traces τ_i^{κ} of length κ.
3. Create the non-observable behaviour of the IPN and simplify it.
4. Complete the IPN adding the observed behaviour and deleting implicit places.

The steps of this procedure are described below.

4.1 Sample processing

4.1.1 Event sequences

As stated before, the data obtained by observing the system to be identified is a set of sequences of I/O vectors w_1, w_2,… such that $w_i = w_i(1)w_i(2)\ldots$ 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* $\lambda(\tau_i(j))$ is the variation between two consecutive input vectors $I_i(j)$, $I_i(j + 1)$; it is computed as $\lambda(\tau_i(j)) = I_i(j+1) - I_i(j)$.

An input event vector can be represented in a compact way by specifying only the input symbols that changed in the event. I_i_1 denotes the change from 0 to 1 of the input I_i; similarly I_i_0 denotes the change from 1 to 0 of the input I_i. Then the *symbolic input event* $\lambda'(\tau_i(j))$ is a string composed by the representation of the inputs changed in the event vector.

$$\lambda'(\tau_i(j)) = \prod_{i=1}^{m} I_i\_x \quad where \quad I_i\_x = \begin{cases} I_i\_1 & if \quad I_i(j+1) - I_i(j) = 1 \\ I_i\_0 & if \quad I_i(j+1) - I_i(j) = -1 \\ \varepsilon & if \quad I_i(j+1) - I_i(j) = 0 \end{cases}$$

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

Example 1. Consider a DES with $q = 4$ output signals, $\Phi = \{A, B, C, D\}$, and $r = 3$ input signals $\Sigma = \{a, b, c\}$. Three I/O sequences have been observed; vector entries correspond to distribution $[a\ b\ c \mid A\ B\ C\ D]^T$

$$w_1 = \left[\begin{array}{c|c|c} 0 & 1 & 1 \\ 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right] \qquad w_2 = \left[\begin{array}{c|c|c|c|c|c} 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right] \qquad w_3 = \left[\begin{array}{c|c|c|c|c|c} 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

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

For the Example 1, sequences τ_i of the detected event vectors e_j associated to I/O changes are obtained:

$$w_1 = \begin{bmatrix}0\\0\\\frac{0}{1}\\0\\0\\0\end{bmatrix} \xrightarrow{e_1} \begin{bmatrix}1\\0\\\frac{0}{-1}\\0\\0\\0\end{bmatrix} \begin{bmatrix}1\\0\\\frac{0}{0}\\0\\0\\0\end{bmatrix} \xrightarrow{e_2} \begin{bmatrix}0\\0\\\frac{0}{0}\\1\\0\\0\end{bmatrix} \begin{bmatrix}1\\0\\\frac{0}{0}\\1\\0\\0\end{bmatrix}, \ \tau_1 = e_1 e_2$$

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

$$w_2 = \begin{bmatrix}0\\0\\\frac{0}{1}\\0\\0\\0\end{bmatrix} \xrightarrow{e_1} \begin{bmatrix}1\\0\\\frac{0}{-1}\\0\\0\\0\end{bmatrix} \begin{bmatrix}1\\0\\\frac{0}{0}\\0\\0\\0\end{bmatrix} \xrightarrow{e_3} \begin{bmatrix}0\\1\\\frac{1}{0}\\0\\1\\0\end{bmatrix} \begin{bmatrix}1\\1\\\frac{1}{0}\\0\\1\\0\end{bmatrix} \xrightarrow{e_4} \begin{bmatrix}0\\-1\\\frac{0}{0}\\0\\0\\1\end{bmatrix} \begin{bmatrix}1\\0\\\frac{1}{0}\\0\\1\\1\end{bmatrix} \xrightarrow{e_5} \begin{bmatrix}0\\0\\\frac{-1}{0}\\0\\-1\\0\end{bmatrix} \begin{bmatrix}1\\0\\\frac{0}{0}\\0\\0\\1\end{bmatrix} \xrightarrow{e_6} \begin{bmatrix}-1\\0\\\frac{0}{1}\\0\\0\\-1\end{bmatrix} \begin{bmatrix}0\\0\\\frac{0}{1}\\0\\0\\0\end{bmatrix}, \tau_2 = e_1 e_3 e_4 e_5 e_6$$

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

$$w_3 = \begin{bmatrix}0\\0\\\frac{0}{1}\\0\\0\\0\end{bmatrix} \xrightarrow{e_1} \begin{bmatrix}1\\0\\\frac{0}{-1}\\0\\0\\0\end{bmatrix} \begin{bmatrix}1\\0\\\frac{0}{0}\\0\\0\\0\end{bmatrix} \xrightarrow{e_3} \begin{bmatrix}0\\1\\\frac{1}{0}\\0\\1\\0\end{bmatrix} \begin{bmatrix}1\\1\\\frac{1}{0}\\0\\1\\0\end{bmatrix} \xrightarrow{e_5} \begin{bmatrix}0\\0\\\frac{-1}{0}\\0\\-1\\0\end{bmatrix} \begin{bmatrix}1\\1\\\frac{0}{0}\\0\\0\\0\end{bmatrix} \xrightarrow{e_4} \begin{bmatrix}0\\-1\\\frac{0}{0}\\0\\0\\1\end{bmatrix} \begin{bmatrix}1\\0\\\frac{0}{0}\\0\\0\\1\end{bmatrix} \xrightarrow{e_6} \begin{bmatrix}-1\\0\\\frac{0}{1}\\0\\0\\-1\end{bmatrix} \begin{bmatrix}0\\0\\\frac{0}{1}\\0\\0\\0\end{bmatrix}, \tau_3 = e_1 e_3 e_5 e_4 e_6$$

4.1.2 Sequences of κ-length event vector traces

In order to distinguish sub-sequences including the same events, event traces of length κ are used. From every sequence $\tau_i = \tau_i(1)\ \tau_i(2)\ldots\ \tau_i(|w_i|-1)$ we compute sequences of event vector traces $\tau_i^{\kappa} = \tau_i^{\kappa}(1)\tau_i^{\kappa}(2)\ldots\ \tau_i^{\kappa}\ (|w_i|-1)$ such that every $\tau_i^{\kappa}(j)$ is the κ-length substring of τ_i that finishes with $\tau_i^{\kappa}(j)$. 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 Example 1, the sequences of traces using $\kappa = 2$ are:

$\tau_1^2 = \varepsilon e_1, e_1 e_2$ for τ_1

$\tau_2^2 = \varepsilon e_1, e_1 e_3, e_3 e_4, e_4 e_5, e_5 e_6$ for τ_2

$\tau_3^2 = \varepsilon e_1, e_1 e_3, e_3 e_5, e_5 e_4, e_4 e_6$ for τ_3

4.2 Building the basic structure

4.2.1 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$\rightarrow\{\tau_i^{\kappa}(j)\}$; the firing of a transition implies that κ consecutive event vectors related to such a transition have been observed.

In order to preserve the firing order between transitions, dependencies are created between them and associated with an observed marking through the function $\mu: P^u \rightarrow \{\varphi M_i \mid M_i \in R(G, M_0)\}$, which relates every non-measurable place with an observed marking, such that every transition has only one input place and one output place $\left(\forall t_r \in T, \left|{}^{\bullet}t_r\right| = \left|t_r^{\bullet}\right| = 1\right)$. 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.

Algorithm 2. Building the basic IPN structure

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

1. $T \leftarrow \varnothing$; $ET \leftarrow \varnothing$; $P \leftarrow \{p_{ini}\}$; $M_0(p_{ini}) \leftarrow 1$; $\mu(p_{ini}) \leftarrow \tau_i(1)$; *//Initialise the net*
2. $\forall \tau_i^{\kappa} \in \Gamma^{\kappa}$
   2.1. $current \leftarrow p_{ini}$; *//Keep track of the current place*
   2.2. $\forall \tau_i^{\kappa}(j) \in \tau_i^{\kappa}$, $1 \le j \le |\tau_i^{\kappa}|$ *//Analyse every event trace*
   2.2.1. If $\tau_i^{\kappa}(j) \notin ET$ *//The trace is new*
   Then
   $ET \leftarrow ET \cup \{\tau_i^{\kappa}(j)\}$; *//Create a new transition label*
   $T \leftarrow T \cup \{t_r^{ej}\}$; $\gamma(t_r^{ej}) \leftarrow \tau_i^{\kappa}(j)$; *//Create a new transition labelled with the trace*
   $\forall p_a \in P$
   $I(p_a, t_r^{ej}) \leftarrow 0$; $O(p_a, t_r^{ej}) \leftarrow 0$;
   $I(current, t_r^{ej}) \leftarrow 1$; *//Create an arc from current place to the new transition*
   If $j = |\tau_i^{\kappa}|$ and $\mu(current) + e_j = \mu(p_{ini})$ *//The trace is the last one*
   Then $O(p_{ini}, t_r^{ej}) \leftarrow 1$; *//Create an arc from the transition to the initial place*
   Else
   $P \leftarrow P \cup \{p_{out}\}$; *//Create a new place*
   $\forall t_b \in T$
   $I(p_{out}, t_b) \leftarrow 0$; $O(p_{out}, t_b) \leftarrow 0$;
   $\mu(p_{out}) = \mu(current) + e_j$; $O(p_{out}, t_r^{ej}) \leftarrow 1$; *//Relate such a place with observed marking and add an arc from the transition to such a place*
   $current \leftarrow p_{out}$ *//Take such a place as current*
   2.2.2. If $\tau_i^{\kappa}(j) \in ET$ *//The trace is not new*
   Then

If $\exists p_{in} \mid p_{in} = {}^{\bullet}(t_r^{ej})$, $t_r^{ej} \in T$, $\gamma(t_r^{ej}) = \tau_i^{\kappa}(j)$ and $\mu(p_{in}) = \mu(current)$ *//If there is a place with the same observed marking preceding a transition with the same label*

Then *merge*($current$, p_{in}); $current \leftarrow (t_r^{ej})^{\bullet}$; *//Merge such a place with current*

Else Go to step 2.2.1 // $\tau_i^{\kappa}(j) \in ET$ *is not considered*

If $j = |\tau_i^{\kappa}|$ and $\mu(current) += \mu(p_{ini})$ *//The trace is the last one*

Then *merge*($current$, p_{ini});

Properties of Algorithm 2

- ***Characteristics of G.*** The obtained model is a PN state machine (PNSM). After processing the first sequence, the PN graph yield is a circuit; afterwards, when previously created transitions whose associated traces have been found, such a transition and its input place are merged (step 2.2.2); thus the fusion preserves the structural nature of a PNSM. Since $M_0(p_{ini})=1$, (G, M_0) is 1-bounded.
- ***Complexity.*** Since search operation has linear-time complexity and the algorithm performs the addition of a transition for every computed trace that has not been yet observed, *Algorithm 2* is executed in polynomial-time on the number of observed input/output sequences and their maximum length.
- ***Completeness.*** The IPN G built through *Algorithm 2* represents all and only all the trace sequences (sub-sequences) in Γ^{κ}. In fact, by construction, every sequence τ_i^{κ} is represented in G by a circuit starting from p_{ini} including the sequence of t_r^{ej}; this circuit represents also the sequence τ_i of events. Furthermore, the reuse of computed transitions having associated the same event traces, during the processing of subsequent τ_i^{κ}, is done only when common paths of length κ are built, that does not introduce other sequences.

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

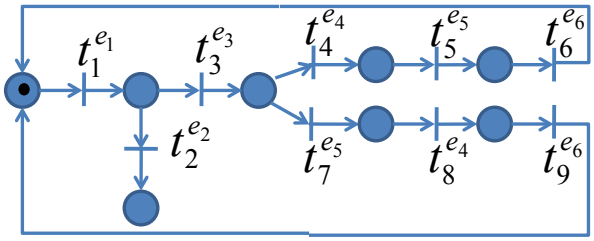

Figure 6. Basic model with sequences of event vector traces

4.2.2 Simplifying the basic structure

Additional node merging operations can be performed on the basic structure in order to obtain a simpler 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 rules on the initial place and iteratively on the merged places

Rule 1: $\forall\ t_a^{ej}, t_b^{ej} \in p_{ini}^{\bullet} |\ a \neq b$ // Common prefixes
$merge(t_a^{ej}, t_b^{ej})$; $merge((t_a^{ej})^{\bullet}, (t_b^{ej})^{\bullet})$ //Merging of transitions and their output places

Rule 2: $\forall t_a^{ej}, t_b^{ej} \in {}^{\bullet}p_{ini} |\ a \neq b$ // Common suffixes
$merge(t_a^{ej}, t_b^{ej})$; $merge({}^{\bullet}(t_a^{ej}), {}^{\bullet}(t_b^{ej}))$ //Merging of transitions and their input places

Properties of Algorithm 3

- ***G' is a safe PNSM.*** The node fusions are performed similarly as in Algorithm 2, thus the structure remains as a PNSM.
- ***G' preserves the trace sequences in G***. Let us analyse the effect of Rule 1; let $Lf_{pi} = \{\lambda(t_1)\lambda(t_2)... \lambda(t_r) | t_1 \in p_i^{\bullet}, t_{i+1} \in (t_i^{\bullet})^{\bullet}\}$ be the set of observable sequences from place p_i. Consider $t_a^{ej}, t_b^{ej} \in p_{ini}^{\bullet} | a \neq b$; before applying Rule 1, $Lfp_{ini} = e_j Lp_a \cup e_j Lp_b \cup (\cup\lambda(t_i)Lp_i)$, with $p_a = (t_a^{ej})^{\bullet}$, $p_b = (t_b^{ej})^{\bullet}$; $t_i \in p_{ini}^{\bullet} | t_i \neq t_a^{ej}, t_i \neq t_b^{ej}$, $p_i = (t_i)^{\bullet}$. After applying Rule 1, $Lfp_{ini} = e_j(Lp_a \cup Lp_b) \cup (\cup\lambda(t_i)Lp_i)$; thus the language of G is not changed by the application of Rule 1. A similar reasoning can be done for Rule 2.

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

4.2.3 Concurrent transitions

Other transformations may be performed when some events associated to transitions appear in different order in different sequences, describing their interleaved firing; this behaviour is exhibited by concurrent transitions in parallel paths in the model. The analysis can be performed on a model component comprised between two transitions t_f and t_j, which have respectively a single output place $t_f^{\bullet}$ and $t_j^{\bullet}$, which are relied by several paths containing the concurrent transitions. If there are $m!$ paths, we can explore if there exists m different transitions in each path and the events sequence in every path appear as a permutation from each other. When it is verified, the subnet can be transformed into a component formed by m concurrent paths from t_f and t_j containing each one a transition related to one of the concurrent events. This transformation of G' preserves the same behaviour.

In Figure 7a, 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 7b.

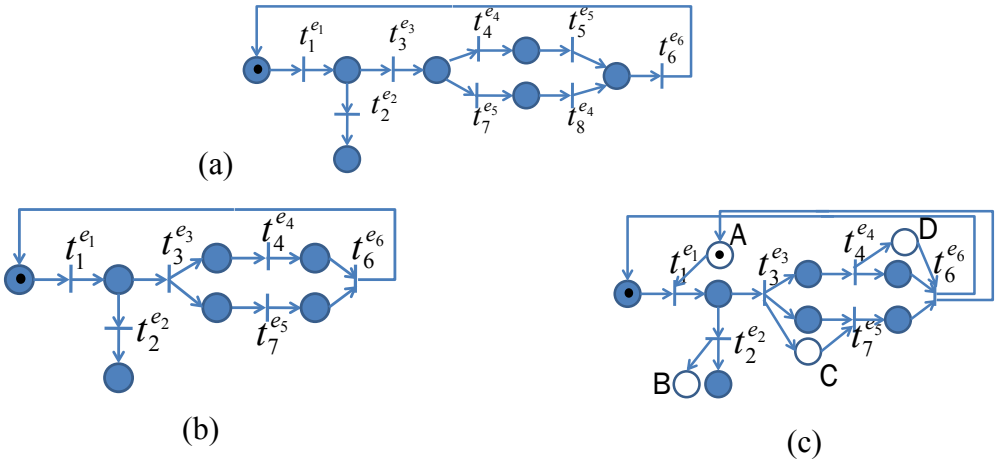

Figure 7. (a) Model after merging (b) Simplified basic model
(c) IPN model including measurable places

Notice that this model preserves the same event vector sequences of the previous one. The equivalent fork-join structure obtained by this transformation is composed by *m* elementary paths composed by two places and one transition between the fork an join transitions. Each path will contain one token in one of its places; this substructure is 1-bounded, the transformed PN is also 1-bounded.

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 behaviour, rather than minimizing the number of nodes in the obtained models.

4.3 Adding interpretation to the PN model

4.3.1 Representing 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 Algorithm 4.

Algorithm 4. Representing outputs changes

Input: G', $\{e_i(j)\}$
Output: H, the IPN including measurable places

Step 1. $P \leftarrow P \cup \{p_1, p_2, ..., p_q\}$.
Step 2. $\forall t_r^{ej} \in T$:
If $e_j(i) = -1$ Then $I(p_i, t_r^{ej}) = 1$ and $O(p_i, t_r^{ej}) = 0$;
If $e_j(i) = 1$ Then $I(p_i, t_r^{ej}) = 0$ and $O(p_i, t_r^{ej}) = 1$;
If $e_j(i) = 0$ Then $I(p_i, t_r^{ej}) = 0$ and $O(p_i, t_r^{ej}) = 0$;
Step 3. If component i of vector $w_j(1)$ is 1 Then $M_0(p_i) = 1$ Else $M_0(p_i) = 0$

The number of measurable places is then equal to the number of outputs. The net with measurable places for the example is showed in Figure 7c.

4.3.2 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 p_k is deleted and its input and output arcs.

4.3.3 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 ***5*** describes a systematic way to do it.

Algorithm 5. Representing input changes

Input: H, $\lambda'(e_j)$
Output: (Q,M_0) the final model of the identification process

Step 1. $\forall t_r^{ej} \in T$, $\lambda(t_r^{ej}) \leftarrow \lambda'(e_j)$

The final model for the illustrative example is showed in Figure 8. The associated inputs for transitions are given by: $\lambda(t_1)=a\_1$, $\lambda(t_2)=\varepsilon$, $\lambda(t_3)=b\_1\ c\_1$, $\lambda(t_4)=b\_0$, $\lambda(t_5)=c\_0$, $\lambda(t_6)=a\_0$.

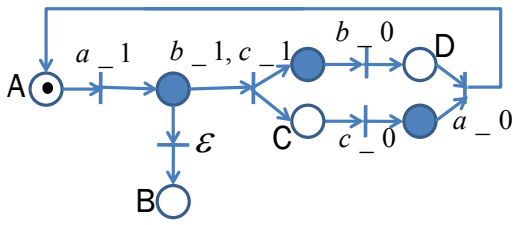

Figure 8. Simplified IPN model

5.4 Completeness of the identified model

Since every one of the transitions in the net actually represents a sub-sequence of event vectors of length κ, the input-output 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 1. 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 $\mathcal{L}^{\kappa+1}(Q,M_0)$, 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 output events (differences of output 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 if and only if it has been 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 $\mathcal{L}^{\kappa+1}(Q) = \mathcal{L}^{\kappa+1}(S)$. ■

5.5 Further simplifications

The identification procedure takes into account any change in the inputs or outputs and represents it in the identified model; in such a model we often find consecutive input events that do not provoke output changes. This situation is due to variations in some inputs, during the system operation, which are not directly related with the outputs changes.

In systems where the number of inputs/outputs and the length of the sequences are large, the identified model size may grow because of this reason. Thus further reductions to the model can be done about the above mentioned sub-sequences of input events, allowing a more compact representation of the system's behaviour. For instance, consider the following I/O vector sequence w involving one input x and two outputs A, B:

$$\begin{bmatrix} x \\ \hline A \\ B \end{bmatrix} \qquad w = \begin{bmatrix} 0 \\ \hline 1 \\ 0 \end{bmatrix}\begin{bmatrix} 1 \\ \hline 1 \\ 0 \end{bmatrix}\begin{bmatrix} 1 \\ \hline 0 \\ 1 \end{bmatrix}$$

In this sequence the event x_1 appears first, and later an output change is produced; it can be represented as: $A \xrightarrow{x\_1} A \xrightarrow{\varepsilon} B$. This behaviour can be represented in a more compact way: $A \xrightarrow{x\_1} B$. In general, this transformation including several input events can be represented as:

$$A \xrightarrow{e_i} A \xrightarrow{e_j} \ldots A \xrightarrow{e_k} B \cong A \xrightarrow{e_i e_j \ldots e_k} B$$

This transformation is applied to paths in the PN model that do not include decision places. This is going to be illustrated on the case study included in Section 6.

6 Method implementation and application

This section presents a software tool for DESs identification and its application to a case study.

6.1 An operational identification tool

Based on the algorithms presented above, a software tool has been developed using Java (JDK 6 Update 11 for compiling and JRE 6 Update 26 for executing) to automate the IPN model synthesis. The architecture of the tool is showed in Figure 9.

The user interface allows capturing the input-output sequences and shows the obtained model graphically. Several data is provided to the tool in text files: the sequences, the parameter κ, the names of the input and output signals, and the output file name. Additionally it is specified the order in which inputs and outputs appear in the txt files and the index numbers of the signals to take into account if a mask is going to be applied.

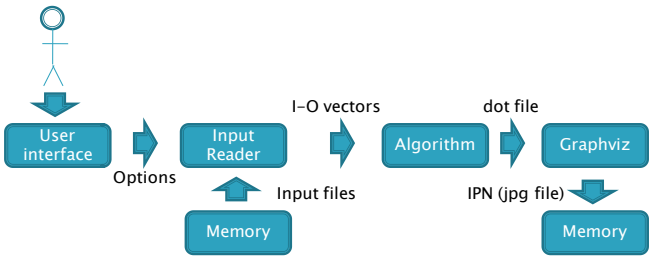

Figure 9. Software architecture

Later, an input reader component processes these input files and transforms every input-output sequence into a vector. These vectors will be delivered to a component called Algorithm in which the identification algorithm is implemented. The output of this component is a dot file that can be treat by Graphviz (an open source graph visualization software: http://www.graphviz.org/) to generate an image file jpg.

The presented identification tool has been tested on many examples of diverse size and complexity, which have been previously solved following the identification procedures. For better understanding, we develop below a small size case of study.

6.2 Case study

We illustrate the use of the software tool through the identification of a small size manufacturing system obtained from [Roth, 2009] described on Figure 10. The purpose of such a system is to sort parcels according to their size. It has 9 PLC inputs, that are signals generated by the sensors of the plant for detecting positions (a0, a1, a2, b0, b1, c0, c1) and presence of parcels detection (k1, k2), and 4 PLC outputs, that are signals controlling the actuators of the plant (A+, A-, B, C).

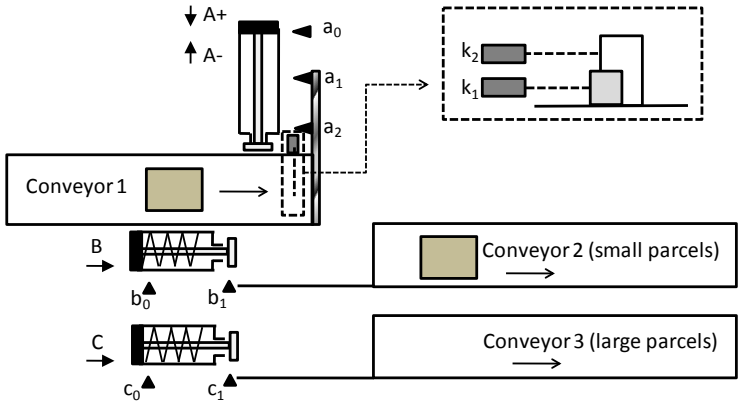

Figure 10. Case study layout

For space considerations we show the processing by the identification tool of nine cyclic I/O sequences; the text files including these sequences are showed in Figure 11. 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].

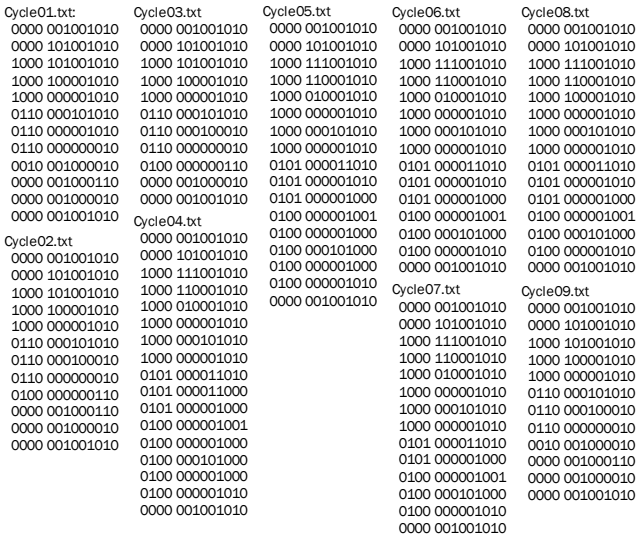

Figure 11. Input-output cyclic sequences of the case study

After the execution of the identification process using κ=1, and before concurrence transformations, the model showed in Figure 12 is obtained; the identified model using κ=2 is showed in Figure 13. After some transformations of the model in figure 12 considering the phenomena described in section 6, the net in Figure 14 is obtained.

From the obtained model, we can infer the following behaviour:

-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 observed behaviour. If the data collection has been made for a long time, we can state that the model is almost exact.

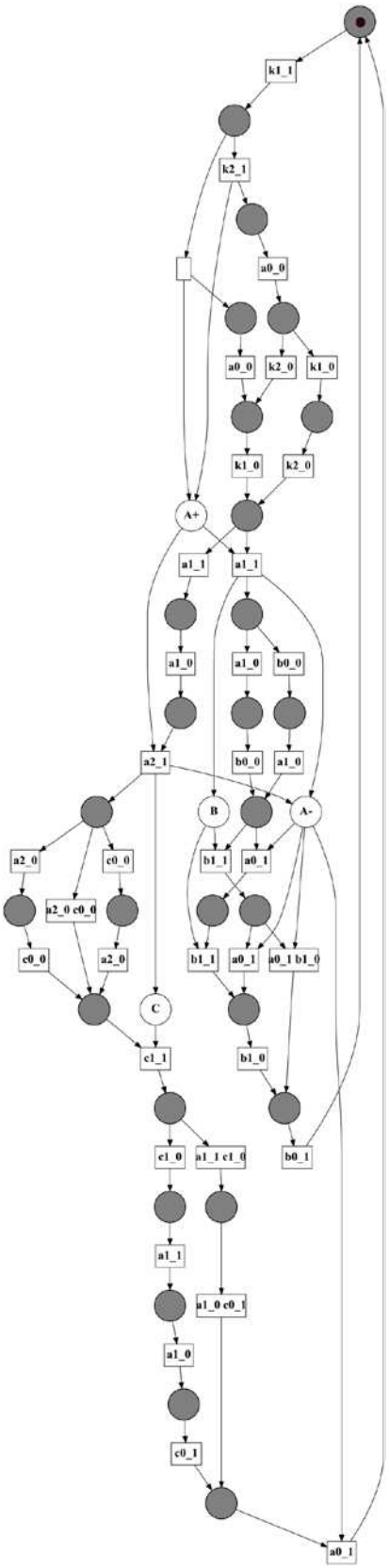

Figure 12. Identified IPN model (κ=1)

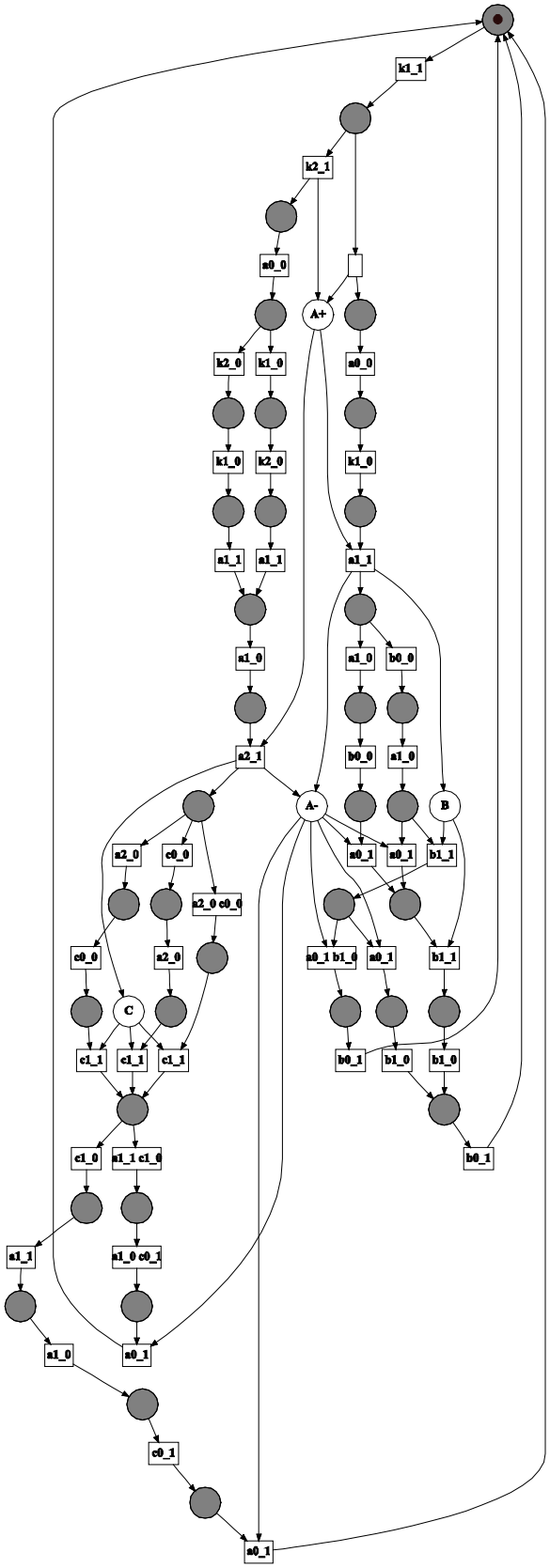

Figure 13. Identified IPN model (κ=2)

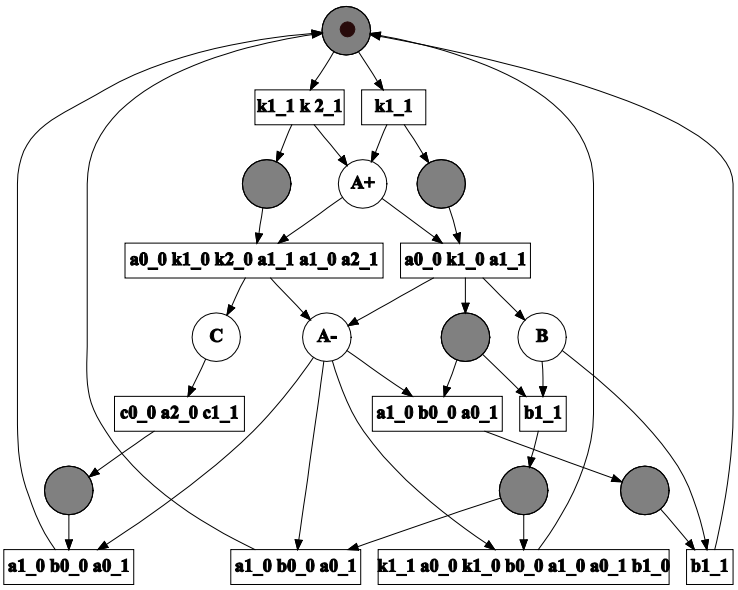

Figure 14. Reduced model for the case study (κ=1)

6.3 Tuning the parameter κ

The parameter κ helps to distinguish sequences of events that look similar during the construction of the basic internal model; its value indicates the history of past events that have to be considered for deciding the state equivalence. On the one hand, high values of κ imply distinguishing more sequences avoiding path fusion during the model construction; thus the obtained models are more accurate but less compact. On the other hand low values of κ allow more state fusions; the obtained model are more compact but more paths can be created yielding an overrepresentation of the observed behaviour.

In general it is not possible to establish a priori the value of κ, since it is assumed that the system is unknown. However, in practice the identification procedure can be applied using several values of κ (because it is not time consuming). Compact models allow a first approximation to the understanding of the system functioning, whilst larger models provide a more precise description. For small examples, such as that included in this section, κ=1 or κ=2 allows distinguishing event sequences whilst compact models are built.

In actual industrial systems the difference can be more drastic. Results in testing the algorithm on an industrial flexible manufacturing line called “Fischertechnik” [Roth, 2009] showed a more remarkable difference between the obtained models for several values of κ. The manufacturing line consists of three machines and three conveyors connecting them. The plant has 30 binary I/Os. During one production cycle, three work pieces are treated. The test has been performed using a Siemens PLC (CPU 315-2 DP); a database has been built using a reliable and efficient UDP (User Datagram Protocol) connection.

The database containing 100 input-output sequences whose length varies from 178 to 186, has been processed by the identification procedure using different values of κ from 1 to 6. Since the obtained models are huge, they are not showed herein; instead, the size of

the identified models are summarised in table 1. The execution time of the identification procedure is also included to provide an idea of the performance of the algorithm. The tests have been performed in a Laptop computer based on an Intel Core 2 Duo T7300 processor at 2.00 GHz with 2.00 GB of RAM under Windows XP Professional 2002 Service Pack 3. The time has been measured excluding the execution of the Graphviz visualisation software.

κ	Transitions	Places	Total of nodes	Δ	Processing time
1	541	375	916		516 ms
2	748	580	1328	412	547 ms
3	958	783	1741	413	547 ms
4	1175	996	2171	430	562 ms
5	1399	1212	2611	440	578 ms
6	1623	1443	3066	455	609 ms

Table 1. Size of identified models for different values of κ.

7 Concluding remarks

Identification of automated concurrent Discrete Event Systems (DESs) has been addressed. The method herein proposed allows dealing with complex actual automated DESs because it takes into account technological characteristics of actual industrial controlled systems, and because it is based on efficient algorithms. This feature is not still addressed in current literature on the matter in which several features considered in the current stated problem have not been dealt.

Although in [Meda, 2000] and subsequent works the proposed algorithms are efficient, the identified models represent also non observed behaviour, due to the fact that the state equivalence is based on the observation of the same observed outputs vector, which is not very often the case for real systems; besides system's inputs are not taken into account. The Techniques based on ILP derived from [Giua 2005] yield models representing exactly the event sequences regardless how the events are obtained from I/O data; however, they are limited to deal with few short event sequences. In [Klein 2005] and subsequent papers, both inputs and outputs are considered, however obtained FA models do not describe explicitly the input-output reactive behaviour nor output concurrent evolutions.

The black-box approach proposed herein allows obtaining IPN models from cyclic input/output vector sequences that represent the closed loop behaviour of PLC-based controlled plants. The proposed technique builds the IPN that is a close approximation of the compound controller-plant behaviour, which can be detailed later for controller redesign or model-based diagnosis purposes.

Current research deals with inference of cyclic sequences from a single long sequence and with the reduction of the obtained model by the analysis of the ulterior influence of inputs that apparently do not provoke changes in the outputs. Also, the inference of non-observed behaviour regarding concurrent sub-processes is an issue to deal with.

8 References

[Angluin, 1988] D. Angluin, *Queries and Concept Learning*, Machine Learning, Vol. 2, No.4 pp. 319-342, 1988.

[Bel Mokadem, 2010] H. Bel Mokadem, B. Bérard, V. Gourcuff, J.M. Roussel, O. De Smet, *Verification of a timed multitask system with UPPAAL,* IEEE Trans. on Automation Science and Engineering, Vol. 7, No. 4, pp. 921-932, October 2010

[Biermann and Feldman, 1972] A.W. Biermann and J.A. Feldman, *On the Synthesis of Finite-State Machines from Samples of Their Behavior*, IEEE Trans. on Computers, Vol. 21, No. 6, pp. 592-597, 1972

[Booth, 1967] T.L. Booth, *Sequential Machines and Automata Theory*, John Wiley and Sons, Inc. New York, London, Sidney, 1967

[Cabasino, 2007] M. P. Cabasino, A. Giua, C. Seatzu, *Identification of Petri Nets from Knowledge of Their Language*, Discrete Event Dynamic Systems Vol. 17, No. 4, pp. 447-474, 2007

[Chang and Hanna, 2004] J. C. Chang and S. R. Hanna, *Air quality model performance evaluation*, Meteorology and Atmospheric Physics, Vol. 87, pp. 167–196, 2004

[Conolly, 2010]D. Connolly, H. Lund, B.V. Mathiesen, M. Leahy, *Modelling the existing Irish energy system to identify future energy costs and the maximum wind penetration feasible*, Energy, Vol. 35, pp. 2164–2173, 2010

[Cook, 2004] J. E. Cook, Z. Du, C. Liu, A. Wolf, *Discovering models of behavior for concurrent workflows*, Computers in Industry, Vol. 53, No. 3, pp. 297 – 319, doi: 10.1016/j.compind.2003.10.005.

[Dotoli, 2008] 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, 2008

[Dotoli, 2011] M. Dotoli, M. P. Fanti, A. M. Mangini, W. Ukovich, *Identification of the unobservable behaviour of industrial automation systems by Petri nets,* Control Engineering Practice, Vol. 19, Issue 9, September 2011, pp. 958-966

[Du, 2009] G. Du and al., *A dynamic workflow modelling and performance analysis methodology for complicated clinical pathway with variations*, Proc. Of the 2nd IFAC workshop on Dependable Control of Discrete Systems (DCDS'09), Bari, Italy, pp. 205-210, 2009

[Estrada, 2009] 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

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

[Estrada, 2010b] A.P. Estrada-Vargas, E. Lopez-Mellado , J-J. Lesage. "An Identification Method for PLC-based Automated Discrete Event Systems". IEEE Int. Conference on Decision and Control, pp.6740-6746. Atlanta, USA, December 2010.

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

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

[Gold, 1967] E.M. Gold, *Language Identification in the Limit*, Information and Control, Vol. 10, No.5 pp. 447-474, 1967

[Hiraishi, 1992] K. Hiraishi, *Construction of Safe Petri Nets by Presenting Firing Sequences*, *Lectures* Notes in Computer Sciences, Vol. 616, pp. 244-262, 1992

[Ishizaka, 1990] H. Ishizaka, *Polynomial Time Learnability of Simple Deterministic Languages*, Machine Learning, Vol. 5, pp. 151-164, 1990

[Kella, 1971] J. Kella, *Sequential Machine Identification*, IEEE Trans. on Computers, Vol. 20, No. 3pp. 332-338, 1971

[Klein, 2005] 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), 2005

[Lanubile, 2002] F. Lanubile and G. Visaggio, *Iterative Re-engineering to compensate for Quick-Fix Maintenance*, Proc. Of IEEE Int. Conf. on Software maintenance, Opio, France, pp. 140-146, 2002

[Levy and Joshi, 1978] L.S. Levy and A.K. Joshi, *Skeletal structural descriptions*, Information and Control, Vol. 39, No. 2 pp. 192-211, 1978

[Lohman, 2007] S. Lohmann, O.Stursberg, and S. Engell, *Comparison of Event-Triggered and Cycle-Driven Models for Verifying SFC Programs*, Proc. of IEEE American Control Conference, New York City, USA, pp. 3606-3611, July 11-13, 2007

[Meda, 2000] M. Meda, A. Ramírez, E. López "Asymptotic Identification for DES", Proc. *IEEE Conf. on Decision and Control*, Sydney, Australia, pp. 2266-2271, Dec 2000.

[Meda, 2001] M. Meda-Campaña, E. López-Mellado, *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, 2001

[Meda, 2003] M. Meda-Campaña, E. López-Mellado, *Required Transition Sequences for DES identification*, Proc. of the IEEE Conf. on Decision and Control (CDC 2003), Maui, Hawaii USA. pp. 3778-3782, 2003

[Mottershead, 1993] J.E. Mottershead and M.I. Friswell, *Model updating in structural dynamics: a survey*, Journal of Sound and Vibration, 167(2), pp. 347-375, 1993

[Ould El Medhi, 2006] S. Ould El Medhi, E. Leclercq, D. Lefebvre, *Petri nets design and identification for the diagnosis of discrete event systems*, IAR Annual Meeting, Nancy, Nov 2006.

[Ould El Medhi, 2012] S. Ould El Medhi, R. Bekrar, N. Messai, E. Leclercq, D. Lefebvre, B. Riera, *Design and implementation of stochastic and deterministic stochastic Petri nets*, IEEE Trans. on Systems, Man and Cybernetics – Part A, On-line version: DOI: 10.1109/TSMCA.2011.2173798

[Richetin, 1984] M. Richetin, M. Naranjo and P. Luneau, *Identification of Automata by Sequential Learning*, Pattern Recognition Letters 2, Vol. 2, No. 6, pp. 379-385, 1984

[Roth, 2009] M. Roth, J.-J. Lesage, L. Litz, *Distributed identification of concurrent discrete event systems for fault detection purposes,* Proc. of the European Control Conference (ECC 2009), pp. 2590-2595

[Roth, 2010] 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, 2010

[Roth, 2012] M. Roth, S. Schneider, J.-J. Lesage, L. Litz, *Fault detection and isolation in manufacturing systems with an identified discrete event model* Int. Journal of Systems Science, DOI**:**10.1080/00207721.2011.649369, Available online: 21 Feb 2012

[Takada, 1998] Y. Takada, *Grammatical Inference for Even Linear Languages Based on Control Sets*, Information Processing Letters, Vol. 28, pp. 193-199, 1998

[Valiant, 1984] L.G. Valiant, *A theory of the Learnable*, Communications of the ACM*,* Vol. 27, pp. 1134-1142, 1984

[Van der Aalst, 2004] W. van der Aalst, T. Weijters, L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs", IEEE Trans. on Knowledge and Data Engineering, Vol. 16, No. 9, Sep 2004.

[Veelenturf, 1978] L.P.J. Veelenturf, *Inference of Sequential Machines from Sample Computations*, IEEE Trans. on Computers, Vol. 27, No. 2 pp. 167-170, 1978

[Veelenturf, 1981] L.P.J. Veelenturf, *An Automata theoretical approach to developing learning neural networks*, Cybernetics and Systems, Vol. 12, No. 1&2 pp. 179-202, 1981

Notes on contributors

Ana-Paula Estrada-Vargas received the B.Sc. degree in computer engineering from the Universidad de Guadalajara, Guadalajara, Mexico, in 2007, and the M.Sc. degree from CINVESTAV, Guadalajara, Mexico, in 2009. She is currently a Ph.D. student in both CINVESTAV in Guadalajara and the ENS de Cachan, in Cachan, France. Her research interests include identification of Discrete Event Systems and formal modelling and analysis using Petri nets.

Ernesto López-Mellado received the B.Sc. degree in electrical engineering from the Instituto Tecnolgico de Cd. Madero, México, in 1977, the M.Sc. degree from the CINVESTAV, México City, México, in 1979, and the Docteur-Ingnieur degree in automation from the University of Toulouse, France, in 1986. Currently, he is Professor of Computer Sciences at CINVESTAV Unidad Guadalajara, Guadalajara, México. His research interests include discrete event systems, and distributed intelligent systems.

Jean-Jacques Lesage received the Ph.D. degree from the Ecole Centrale de Paris and the "Habilitation à diriger des recherches" from the University Nancy 1 in 1989 and 1994 respectively. He is currently Professor of Automatic Control at the Ecole Normale Supérieure de Cachan, France, where he was head of the Automated Production Research Laboratory during eight years. His research interests are in the field of formal methods and models for synthesis, analysis and diagnosis of Discrete Event Systems (DES), and applications to manufacturing systems, network automated systems, energy production, and ambient assisted living.