Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 303107, 13 pages
http://dx.doi.org/10.1155/2015/303107

Research Article

Stochastic DES Fault Diagnosis with Coloured Interpreted Petri Nets

Doyra Mariela Muñoz,[1] Antonio Correcher,[2] Emilio García,[2] and Francisco Morant[2]

[1]*Grupo de Automática Industrial, Universidad del Cauca, Popayán, Colombia*
[2]*Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera, s/n, 46022 Valencia, Spain*

Correspondence should be addressed to Doyra Mariela Muñoz; mamunoz@unicauca.edu.co

Received 22 October 2014; Revised 3 February 2015; Accepted 4 February 2015

Academic Editor: Hiroyuki Mino

This proposal presents an online method to detect and isolate faults in stochastic discrete event systems without previous model. A coloured timed interpreted Petri Net generates the normal behavior language after an identification stage. The next step is fault detection that is carried out by comparing the observed event sequences with the expected event sequences. Once a new fault is detected, a learning algorithm changes the structure of the diagnoser, so it is able to learn new fault languages. Moreover, the diagnoser includes timed events to represent and diagnose stochastic languages. Finally, this paper proposes a detectability condition for stochastic DES and the sufficient and necessary conditions are proved.

1. Introduction

Fault diagnosis has a major role in industrial systems since it allows the fault detection as soon as possible to avoid serious damages of the system or the injury of an operator. Fault diagnosis of Discrete Event Systems (DES) is an issue that has been addressed from different approaches. A fault is a deviation of the normal or required behavior. Fault diagnosis is the process of detecting and identifying such deviations of the system by using the information available on system variables [1].

According to [2], fault diagnosis aims to achieve three complementary tasks: fault detection, fault isolation, and fault identification. Fault detection is a functionality that decides whether the system works in normal conditions or whether a fault has occurred. If a fault has occurred, fault isolation aims to locate the component(s) causing the fault. Fault identification is concerned with identifying the specific nature of the fault (its size, criticality, importance, etc.). This problem has been addressed by many researchers related with developing new models, new properties, new algorithms, and efficient solutions to fault diagnosis of DES. Model based diagnosis techniques can be divided into two groups. The first group uses models which include fault-free and faulty behaviors. The second group only uses fault-free models.

The work of [3, 4] has provided a formal foundation of fault diagnosis and diagnosability analysis of DES that has been the base for many approaches of diagnosis. They use an automaton which generates all the possible event sequences in nominal and faulty operation.

Petri Nets (PNs) have been recognized as a suitable model to describe DES, particularly when a system is asynchronous [5, 6]. PN has been used for fault diagnosis starting from [7–9] who presented diagnosis proposals of estimating faulty states. In [10] a net unfolding approach to online asynchronous diagnosis is presented. This proposal avoids the state explosion problem that typically results from having concurrent components interacting asynchronously in a distributed system, but the computing cost of performing the online diagnosis increases for offline diagnosis. In [11], the authors extend the proposal of [3] to online fault diagnosis of modeled systems by PN. Some years later, these authors in [12] present two new algorithms to deal with the case of multiple modules and real-time communication requirements. In [13] the authors not only model faults by unobservable transitions but also include other transitions representing legal unobservable behaviors as well. They prove that all possible firing sequences corresponding to a given observation can be characterized and based on the notion of basis markings and justifications. The authors use a basis reachability tree to

compute the set of basis markings; [6] changes the concept of basis marking and enumerates only a subset of the reachability space. This approach includes a different characterization in terms of new original notions such as justifications and minimal explanations. The work of [14] considers the system modeled as an interpreted PN (IPN) with partially observable states and events; the model includes the possible faults that may happen. Reference [15] proposes an online fault detection technique to avoid the redesign and the redefinition of the diagnoser when the structure of the system changes. The diagnoser waits for an observable event and an algorithm decides whether the system behavior is normal or may exhibit some possible faults. The solution of an integer linear programming (ILP) problem provides a sequence of unobservable transitions containing the faults that may have occurred. The system is modeled by IPN where fault events are modeled as unobservable transitions. It associates a different label to each transition, so it models the regular behavior. In [16] the authors started from the results of [15]. They extend the work by considering a new source of nondeterminism (different observable transitions sharing the same label) and by considering distributed systems. To conclude [17] builds an online diagnoser based on PN approach, using the ILP definition and resolution.

The advantage of this class of methods lies in the possibility to give guarantees about the diagnosability of faults; moreover, if certain conditions hold, modeled faults can be precisely localized. An inherent disadvantage is that only faults explicitly considered in the system model can be detected and localized.

Diagnosis methods without fault model avoid this disadvantage; moreover, they build straightforward models since no special knowledge of system fault behavior is necessary. Nevertheless, the main drawback of these approaches is how to locate the fault since the models have less knowledge. Moreover, diagnosability of a given set of faults usually cannot be guaranteed. These methods are based on comparing the system outputs with model nominal outputs. In [18, 19] the proposed method compares the observed and the expected behavior, a fault can be detected, and a set of fault candidates is determined. Inspired by residuals known from diagnosis in continuous systems, different set operations are introduced to generate the fault candidate set. After fault detection and a first fault localization, a procedure is given to locate the fault more precisely by an analysis of the further observed system behavior.

Coloured Petri Nets (CPNs) have also been oriented to fault diagnosis. Some approaches deal with combinatorial explosion, so they can be used to diagnose large systems. In [20], the authors present a method for modeling flexible manufacturing systems including fault models (based on fault trees). In [21], the authors present a method for modeling and diagnosing an orchestrated complex Web service. This approach is very restrictive to Web services models. In [22] the author presents a method for including the fault diagnosis within an embedded controller. The integration between diagnosis and controller in a reduced CPN model is suitable since it allows merging information about device states in a single token. Nevertheless, this approach has the same weak point as other model based approaches; it needs a fault model.

Reference [23] presented a new diagnosis method based on CPN called Latent Nestling Method. The initial model of the normal behavior of the system is performed from modeling techniques, based on generalized PNs. However, for complex systems the synthesis capabilities of CPNs can be used in these modeling steps. The set of faults to be diagnosed are defined and assigned to the subset of coloured tokens. A faulty event will be defined by establishing dynamic conditions in every marking and subsequently in every state reached by the system and the set of unexpected signals of the sensor readings. Next the coloured tokens of faulty events are allocated in appropriate places called places of latent nestling faults. These tokens are susceptible to fire from that place by the activation of an event sequence, s, associated with an abnormal sensor reading.

Regarding diagnosability [3] defines diagnosability in the framework of formal languages and presents necessary and sufficient conditions for diagnosability of DES. The authors in [24] focused on diagnosability of IPN. They defined and characterized the property of input-output diagnosability in IPN models, so they avoided the reachability analysis. In their next work [25], they presented a polynomial algorithm to decide if an IPN is diagnosable. Reference [26] provides a necessary and sufficient condition for diagnosability of bounded PNs, namely, PNs whose set of reachable markings is finite. The effectiveness of the proposed procedure was illustrated in [27]. They showed that, under certain conditions, the number of basis markings (a basis marking is a marking reached from M_0 with the firing of σ_0, where $\sigma_0 \in Tr_0 \mid \mathscr{L}(\sigma_0) = \omega$, and of all unobservable transitions whose firing is strictly necessary to enable ω) is always smaller than the number of reachable markings (that increase exponentially with the size of the net).

Approach of the Work. In [2], the authors made a classification of diagnosis methods with respect to a number of criteria such as fault compilation (offline or online), modeling tools (automaton, PN, and state machines), fault representation (fault model: event-based or state-based, fault-free model), and decision structure or architecture (centralized, decentralized, and distributed). According to this classification, the diagnoser presented in this proposal can be classified as online fault diagnosis, based on PN without previous model and under a centralized structure.

In general terms the proposal is based on language theory and on stochastic timed interpreted Petri Nets (st-ICPN), as structures to generate DES languages. The diagnosis process starts by identifying the fault-free model, from the observed language. As a result, a st-ICPN language generator is built. The generator of the fault-free language is a base to building the diagnoser and, in addition, to the concepts of Coloured Petri Nets. A learning algorithm modifies the net structure each time a new fault is detected, so the diagnoser is able to learn fault languages. The net structure changes with each new detected fault. The modifications are as follows: addition of a token in the fault transition, modification of the arcs linking that transition, and the addition of a specific fault token to the initial marking.

The main advantages of this proposal about other diagnosis methodologies are the generation of deterministic models and the absence of previous fault models. The learning algorithm guarantees the diagnosis of faults not included in the fault set.

This paper is organized as follows: Section 2 describes the background on PN; Section 3 describes the fault behavior; Section 4 presents the diagnosis method; Section 5 shows an application case; and finally, the concluding remarks and discussion are shown in Section 6.

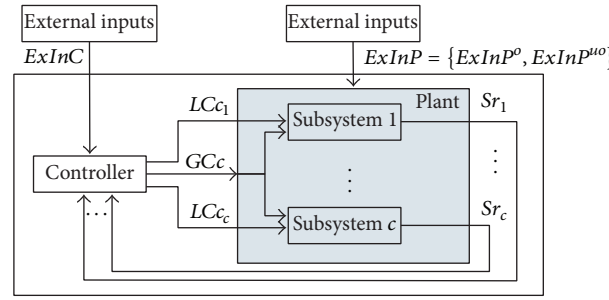

Figure 1: Complex system.

2. Background

This section introduces the formalism and definitions used in the paper.

2.1. System under Study. In real systems, there are various stochastic disturbances such as sensor noises, stochastic disturbance, fault, or random variation of parameters. Thus, the system representation should be based on stochastic models [28]; therefore, the system to be diagnosed is a stochastic DES, whose dynamics can be described by the interrelation of I/O signals and its behavior could be described with formal languages based on event sequences; the system is split into c subsystems; a subsystem is a part of a system with a particular behavior. This structure can be seen in Figure 1.

In a closed loop system, there exist two kinds of inputs. Some inputs are external observable SCADA commands or operator requirements that modify the operation mode of the controller ($ExInC$). The other kind of inputs is external events affecting the plant ($ExInP$); these inputs can be either observable or unobservable. $ExInP$ includes disturbances and interaction with other systems or faults. Moreover, control commands are plant inputs and can be considered as internal signals ($IntInP$) of the closed loop model. When the system can be split into subsystems, control commands can be considered as local or global. A control command is considered global if it is applied to more than one subsystem, and it is considered local otherwise.

System outputs are sensor reading; each sensor reading belongs to a subsystem l; then, the set of sensor readings (S) will be $S = \cup Sr_l$. An input symbol for a subsystem l is composed of global and local control commands, $[gcc_1 \cdots gcc_{n_{gc}} cc_{l,1} \cdots cc_{l,n_{l,cc}}]$, where gcc_i, $i = 1 \cdots n_{gc}$, are global control commands and $cc_{l,j}$, $j = 1 \cdots n_{l,cc}$, are local control commands; it is represented as $u_{l,s}$, $u_{l,s}$ being a binary representation of s; s stands for the input symbol at time τ_i. An output symbol for a subsystem l is $[sr_{l,1} \cdots sr_{l,n_l}]$; it is represented as $y_{l,j}$, $y_{l,j}$ being a binary representation of j; j stands for the output symbol at time τ_i. An operation mode is composed of a combination of $ExInC$ signals, $[ec_1 \cdots ec_{n_{ec}}]$; it is represented as o_{om}, o_{om} being a binary representation of om; om stands for the external event at time τ_i. For example, given a set of sensor readings for subsystem $l = 1$: $Sr_1 = \{sr_{1,1}, sr_{1,2}\}$; if $sr_{1,1} = 1$ and $sr_{1,2} = 0$, then the output symbol is represented by $y_{1,2}$.

2.2. Events and Languages. Let Ω_1, Ω be two event sets, such that $\Omega_1 \subset \Omega$. A language, $\mathscr{L}$, defined over Ω is a set of finite-length strings formed from events in Ω; that is, $\mathscr{L} \subset \Omega^*$. The projection operation, $P : \Omega^* \to \Omega_1$, is defined as $P_\Omega(\varepsilon) = \varepsilon$; $P_\Omega(as) = aP(s)$ if $a \in \Omega_1, s \in \Omega_1^*$; $P_\Omega(as) = P(s)$ if $a \notin \Omega_1$. Given $s, s' \subset \Omega^*$, ss' is the concatenation of s and s'. $|s|$ is the length of s.

Definition 1 (compound event). Given two event sets Ω_p, Ω_q and given two events e_p and e_q, such that $e_p \in \Omega_p$ and $e_q \in \Omega_q$, a compound event ω is the concatenation of e_p and e_q; $\omega = e_p e_q$. $\Omega = \Omega_p \Omega_q$ is a compound event set (Ω over $\Omega_p \Omega_q$) and a language defined over Ω will be $\mathscr{L} \subset (\Omega_p \Omega_q)^*$.

For example, given two event sets $\Omega_p = \{p_1, p_2\}$, $\Omega_q = \{q_1, q_2\}$, a compound event set will be $\Omega = \{(p_1 q_1), (p_1 q_2), (p_2 q_1), (p_2 q_2)\}$.

Definition 2 (projection operation over compound event sets, Pc). Given a compound event set Ω over $\Omega_p \Omega_q$, Pc operation over compound events, $Pc : \Omega \to \Omega_q$, is $Pc_{\Omega_q}(\varepsilon) = \varepsilon$; $Pc_{\Omega_q}(e_p e_q) = P_{\Omega_q}(e_p) P_{\Omega_q}(e_q)$.

Pc used at the previous example will give $Pc : \Omega \to \Omega_p$; $Pc_{\Omega_p}(p_2 q_2) = p_2$. Given an event sequence $s \mid s \in \Omega^*$, where $s = (p_1 q_1)(p_2 q_2)(p_1 q_1)$, Pc of s over Ω_p is $Pc_{\Omega_p}(s) = p_1 p_2 p_1$.

Definition 3 (timed event and stochastic timed event). A timed event is a composition of an event and the elapsed time between two consecutive events. Then $e^i = e \cdot t^{\text{ev}}$ at time τ_i, where $t^{\text{ev}} = |\tau_{\text{current\_ev}}| - |\tau_{\text{previous\_ev}}|$.

A language, $\mathscr{L}$, defined over timed event set Ω is a set of finite-length strings formed from timed events in Ω; that is, $\mathscr{L} \subset \Omega^*$; $\mathscr{L} = \{s \mid s \subset \Omega^*\}$, where s is a timed event sequence, at times $\tau_0 \leq \cdots \leq \tau_k$. In real systems, it is nearly impossible to repeat the same event sequence with the same times between events. In that case, if times fit a probability density function, $t^{\text{ev}} \sim f(t^{\text{ev}})$, timed events are called stochastic timed events.

Definition 4 (projection operation of timed event sequences: Pt). Given two timed event sets Ω_1, Ω, such that $\Omega_1 \subseteq \Omega$, let $s = e^0 \cdots e^k$ be a timed event sequence with $e^i = e \cdot t^{\text{ev}}$; the projection operation of timed event sequences, $Pt : \Omega^* \to \Omega_1$, is defined as $Pt_{\Omega_1}(s) = Pt_{\Omega_1}(e^0) \cdots Pt_{\Omega_1}(e^k)$, for all $e^i \in s$, which is calculated as follows: a variable is started to zero $t_{\text{proy}} = 0$; for $i = 0 \cdots k$, $Pt_{\Omega_1}(e^i) = e \cdot t^{\text{ev}}$ if $e^i \in \Omega_1$; then $t_{\text{proy}} = t^{\text{ev}}$

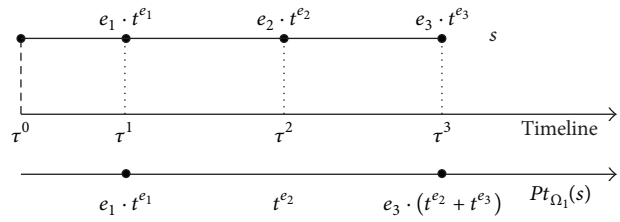

Figure 2: Example of Pt.

and the variable back to zero $t_{\text{proy}} = 0$; else $Pt_{\Omega_1}(e^i) = \varepsilon$ and $t_{\text{proy}} = t_{\text{proy}} + t^{\text{ev}}$.

For example (see Figure 2), given a timed set $\Omega_1 = \{e_1 \cdot t^{e_1}, e_3 \cdot t^{e_3}\}$, let s be a timed event sequence, $s = e_1 \cdot t^{e_1} e_2 \cdot t^{e_2} e_3 \cdot t^{e_3}$; $Pt : \Omega^* \to \Omega_1$; $Pt_{\Omega_1} s = Pt_{\Omega_1}(e_1 \cdot t^{e_1}) Pt_{\Omega_1}(e_2 \cdot t^{e_2}) Pt_{\Omega_1}(e_3 \cdot t^{e_3})$; then

for $i = 0$,

$e_1 \cdot t^{e_1} \in \Omega_1$; then $t_{\text{proy}} = t^{e_1}$; $Pt_{\Omega_1}(e_1 \cdot t^{e_1}) = e_1 \cdot t_{\text{proy}}$; $t_{\text{proy}} = 0$;

for $i = 1$,

$e_2 \cdot t^{e_2} \notin \Omega_1$; then $t_{\text{proy}} = t_{\text{proy}} + t^{e_2} = t^{e_2}$; $Pt_{\Omega_1}(e_2 \cdot t^{e_2}) = \varepsilon$;

for $i = 2$,

$e_3 \cdot t^{e_3} \in \Omega_1$; then $t_{\text{proy}} = t_{\text{proy}} + t^{e_3} = t^{e_2} + t^{e_3}$; $Pt_{\Omega_1}(e_3 \cdot t^{e_3}) = (e_3 \cdot t_{\text{proy}})$.

Therefore $Pt_{\Omega_1} s = e_1 \cdot t^{e_1} e_3 \cdot (t^{e_2} + t^{e_3})$.

Now, if $\Omega_1 = \{e_4 \cdot f(t^{e_4}) e_5 \cdot f(t^{e_5})\}$ and given a stochastic timed event sequence $s = e_4 \cdot t^{e_4} e_5 \cdot t^{e_5}$, then Pt of s over Ω_1 is $Pt_{\Omega_1}(s) = Pt_{\Omega_1}(e_4 \cdot t^{e_4}) Pt_{\Omega_1}(e_5 \cdot t^{e_5})$, where $Pt_{\Omega_1}(e_4 \cdot t^{e_4}) = (e_4 \cdot t^{e_4})$ if $e_4 \in \Omega_1 \wedge a \le t^{e_4} \le b$, where $\int_a^b f(t^{e_4}) \ge (1 - \alpha)$, $(1 - \alpha)$ is the confidence level.

2.3. Petri Nets. Petri Nets (PNs) are widely used for modeling DES ([29]). A PN, N, is a bipartite digraph represented by the five-tuple $N = (P, TR, Pre, Post, M_0)$, where P is a set of places with cardinality np and TR is a set of transitions with cardinality ntr; $Pre : P \times TR \to \mathbb{N}$ and $Post : TR \times P \to \mathbb{N}$ are the *Pre* and *Post incidence matrices* ($I = Post - Pre$). The marking function $M : P \to \mathbb{N}$ represents the number of tokens residing inside each place; M_0 is the initial marking [30, 31]. For the *Pre* and *Post sets*, the dot notation is used: $\dot{}tr = \{p \in P : Pre(p, tr) > 0\}$ [31].

IPN is an extension of PN allowing the association of input and output signals to models [32].

Definition 5 (interpreted Petri Net). An IPN is a tuple $Q = (N, U_o, Y, \lambda, \varphi)$, where N is a PN. $U_o = \{u_0, u_1, \ldots, u_{|2^{m_o}|-1}\}$ is the observable inputs set, u_s is an input symbol, and m_o is the number of observable inputs; $Y = \{y_0, y_1, \ldots, y_{|2^n|-1}\}$ is the output set, y_j is an output symbol, and n is the number of outputs; $\lambda : TR \to U_o$ is a transition labeling function that assigns an input symbol to each transition. $\varphi : R(N) \to Y$ is an output function that assigns an output symbol to each reachable marking.

Differential of output symbols is introduced in [33] to avoid IPN nondeterminism.

Definition 6 (differential of output symbol dy). Given two output symbols $y_j, y_{j-1} \in Y$, at times τ_i and τ_{i-1}, respectively, dy_j is defined as $dy_j(y_j \times y_{j-1}) \to \{-1, 0, 1\}$, where $dy_j = y_j - y_{j-1}$ with $dy_j \in \{-1, 0, 1\}$; so dy_j possible values are $0 \times 0 \to 0$; $0 \times 1 \to 1$; $1 \times 0 \to -1$; $1 \times 1 \to 0$.

A st-IPN is defined as follows.

Definition 7 (stochastic timed interpreted Petri Net (st-IPN) (see [33])). A st-IPN is a structure represented by $stQ = (Q, \Omega, \delta, OM)$, where $Q = (N, U_o, Y, \lambda, \varphi)$ is an IPN; N, U_o, Y have the same meaning as in Definition 5; $\lambda : TR \to U_o \times \delta$ is a labeling function that assigns an input symbol and a time density function to each transition; φ is defined as $\varphi : (RN) \to Y/dY$; φ is isomorphic over Y/dY. Consider $\Omega := (U_o \times Y)$. δ is the system alphabet. $\delta := TR \times OM \to f(t_{TR \times OM})$ is a transition firing time density function for each o_{om}. $OM = \{o_0, \ldots, o_{|2^{n_{ec}}|-1}\}$ is the set of operation modes.

The system alphabet $\Omega = (U_o \times Y)$. δ relates signals. A letter $\omega^i \in \Omega$ is a symbol that concatenates input signals and output signals at every instant τ_i, (I/O symbol); this symbol is a compound event as in Definition 1 but is a timed event (see Definition 3); therefore Ω is a timed compound event set over U_o, Y. Consider

$$\omega^i = \left(u_s y_j\right) \cdot t^{\text{ev}}. \tag{1}$$

Definition 8 (st-ICPN language). The $\mathscr{L}(stCQ)$ is $\mathscr{L}(stCQ) = \{s \mid s \subset \Omega^*\}$, where $s = \omega^0, \ldots, \omega^k$ is a timed compound event sequence, at times $\tau_0 \le \cdots \le \tau_k$; that is, s is an ordered sequence at a time line. If the elapsed time between events is not constant, the time fits a probability density function $f(t^{\text{ev}})$, that is, $t^{\text{ev}} \sim f(t^{\text{ev}})$, and $\mathscr{L}(stCQ)$ is a stochastic language.

Coloured Petri Nets (CPNs) have formal semantics, and they allow different types of analysis [34]. CPNs are defined as follows.

Definition 9 (Coloured Petri Net (CPN) (see [35])). A CPN is defined by a tuple $CN = (P, TR, C, cd, Pre, Post, M_0)$, where P, TR have the same meaning of a PN; $C = \{\text{class}_1, \ldots, \text{class}_{ncc}\}$ is the set of colour classes with cardinality ncc; $cd : P \cup TR \to C$ is the colour domain mapping; $Pre, Post \in \beta^{|P| \times |TR|}$ are incidence matrices, such that $Pre[p, tr] : cd(tr) \to Bag(cd(p))$ and $Post[p, tr] : cd(tr) \to Bag(cd(p))$ are mappings for each pair $(p, t) \in P \times TR$. β can be taken as the set of mappings of the form $f : cd(tr) \to Bag(cd(p))$. Representation of the incidence matrices entries *Pre* and *Post* will be by vectors. A place marking is a vector, such that $m[p] \to Bag(cd(p))$ for each $p \in P$; m_0 is the initial marking.

If a CPN has output and transition labeling functions, it can be considered as an ICPN. Therefore, a PN including the characteristics of CPN and st-IPN can be defined as follows.

Definition 10 (stochastic timed interpreted Coloured Petri Net (st-ICPN)). A st-ICPN is a structure represented by $stCQ = (CN, U_o, Y, \lambda, \varphi, \Omega, \delta, OM)$, where CN is a CPN defined as in Definition 9, U_o, Y are input and output alphabets, respectively, $U_o = \cup U_o^{cl}$ and $Y = \cup Y^{cl}$, with $cl = 1 : ncc$, $\lambda : TR \times Bag(cd(tr)) \rightarrow U_o$. δ is the transition labeling function. $\varphi : R(CN) \times Bag(cd(p)) \rightarrow Y/dY$ is the output function. Consider $\Omega := (U_o \times Y)$. δ is the system alphabet. $\delta := TR \times OM \times C \rightarrow f(t_{TR\times OM\times C})$ is the transition firing time density function for each o_{om} in each cl. $OM = \{o_0, \ldots, o_{|2^{n_{ec}}|-1}\}$ is the set of operation modes.

A transition $tr_r \in TR$ with $\lambda(tr_j) = u_s \cdot f(t_{tr_r,o_{om},cl})$ is enabled with respect to $cd(tr)$ at operation mode o_{om}, if and only if, for all $p_q \in {}^{\cdot}tr_r$, $m[p_q] \geq Pre[{}^{\cdot}tr](cd(tr))$ and if $a \leq t^{\text{ev}} \leq b$, where $\int_a^b f(t_{tr_r,o_{om},cl}) \geq (1-\alpha)$, $(1-\alpha)$ is the confidence level and $[a, b]$ is the confidence interval. If tr_r is enabled and tr_r is fired a new marking M_{k+1} is reached; it is computed with the classical state space equation: $M_{k+1} = M_k + I \cdot \overrightarrow{tr_r}$, $y_j/dy_j = \varphi(M_k)$.

Definition 11 (firing language of a st-ICPN). Let σ^{cl} be a firing sequence $\sigma^{cl} = tr_1 \cdots tr_k$ for colour class cl, of a $stCQ$, such that $M_0[\sigma_i > M_k$. The set of all firing sequences for the colour class cl is called the firing language $\mathscr{L}_F^{cl}(stCQ)$ for cl. Consider $\mathscr{L}_F^{cl}(stCQ) = \{\sigma^{cl} \mid Prob(\sigma^{cl} \mid m_0) \geq (1-\alpha)\}$.

The transition and output labeling sequences generated by σ^{cl} allow the definition of the generated languages as follows.

Definition 12 (input and output languages of a st-ICPN). Let σ^{cl} be a firing sequence such that $\sigma^{cl} \in \mathscr{L}_F^{cl}(stCQ)$; the input language for cl is defined as the labeling function sequences of the $tr_r \in \mathscr{L}_F^{cl}(stCQ)$; that is, $\mathscr{L}_{\text{in}}^{cl}(stCQ) = \{\sigma^{cl} \mid \sigma^{cl} = \lambda(tr_1) \cdots \lambda(tr_k)\}$, and the output language for cl is defined as the reached marking sequences by the firing of σ^{cl}; that is, $\mathscr{L}_{\text{out}}^{cl}(stCQ) = \{\varphi(M_0) \cdots \varphi(M_k)\}$. st-ICPN language is $\mathscr{L}(stCQ) = \{\mathscr{L}^{cl}(stCQ)\}$.

3. Fault Behavior

The system generates a language that can be split into sublanguages, taking into account if a fault has occurred or not. $\mathscr{L}_l$ is the language generated by a subsystem.

The set of timed compound events is partitioned into observable and unobservable events, $\Omega_l = \Omega_{o_l} \cup \Omega_{uo_l}$, where Ω_{uo_l} includes two subsets: fault and regular unobservable event subsets $\Omega_{uo_l} = \Omega_{f_l} \cup \Omega_{\text{reg}_l}$ (adapted from [36]). An event $\omega_l^i \in \Omega_l$ is of the form $\omega_l^i = (u_{l,s} y_{l,j}) \cdot t^{\text{ev}}$ (see (1)). The set of normal timed compound events Ω_{N_l} is a subset of Ω_{o_l} such that $\mathscr{L}^{N_l} \subset \Omega_{N_l}^*$.

A timed fault event can be defined as follows.

Definition 13 (timed fault event). Given a timed event sequence s and an event of the form $\omega_l^{i*} = (u_{l,s} y_{l,j}) \cdot t^{\text{ev}}$ and if $s \in \mathscr{L}^{N_l} \wedge s\omega_l^{i*} \notin \mathscr{L}^{N_l}$, then ω_l^{i*} is a timed fault event.

A fault event, ω_l^i, does not belong to the normal language (based on Definition 2) $\omega_l^i = (u_{l,s}^i y_{l,j}^i) \cdot t^{\text{ev}} \notin \mathscr{L}^{N_l}$ (the superscript "i" has been added to identify the event time), if

(i) $Pc_{\Omega_{N_l}}(u_{l,s}^i) \notin \mathscr{L}_{\text{in}}^{N_l} \wedge Pc_{\Omega_{N_l}}(y_{l,j}^i) \in \mathscr{L}_{\text{out}}^{N_l}$ or

(ii) $Pc_{\Omega_{N_l}}(u_{l,s}^i) \in \mathscr{L}_{\text{in}}^{N_l} \wedge Pc_{\Omega_{N_l}}(y_{l,j}^i) \notin \mathscr{L}_{\text{out}}^{N_l}$ or

(iii) $Pc_{\Omega_{N_l}}(u_{l,s}^i) \notin \mathscr{L}_{\text{in}}^{N_l} \wedge Pc_{\Omega_{N_l}}(y_{l,j}^i) \notin \mathscr{L}_{\text{out}}^{N_l}$ or

(iv) $Pc_{\Omega_{N_l}}(u_{l,s}^i) \in \mathscr{L}_{\text{in}}^{N_l} \wedge Pc_{\Omega_{N_l}}(y_{l,j}^i) \in \mathscr{L}_{\text{out}}^{N_l}$ but $t^{\text{ev}} \notin [a, b]$; $\int_a^b f(t) \geq (1-\alpha)$; $(1-\alpha)$ is the confidence level and $f(t)$ is the density function of normal event, $\omega_l^i \in \mathscr{L}^{N_l}/s$.

In cases (i), (ii), and (vi) the fault event is observable and in case (iii) the fault event is unobservable. Fault language can be defined as follows.

Definition 14 (fault language, $\mathscr{L}^{F_l}$). Given a timed event sequence s, the fault language is $\mathscr{L}^{F_l} = \{st \mid s \in \mathscr{L}^{N_l} \wedge t \in \mathscr{L}/s \wedge \exists \omega_l^{i*} \in t \mid \omega_l^{i*} \notin \mathscr{L}^{N_l}\}$, that is, the set of all timed event sequences with at least one timed fault event in the postlanguage of a normal sequence.

4. Diagnosis Method

The diagnoser proposed in this paper works without any previous knowledge of the system language. The diagnoser construction starts with the identification of the normal behavior, which results in a set of st-IPNs that generate the observed normal language. $\mathscr{L}^{N_l}$ is the language generated by the identified st-IPN for subsystem l. The diagnosis task is carried out by comparing the current event trace t with $\mathscr{L}^{N_l}$. If $t \notin \mathscr{L}^{N_l}$, a timed fault event has been detected. The algorithm creates a language model recognizer for this new situation and t is considered as part of a fault language, $\mathscr{L}^{f_i}$.

Once a fault has been detected, a fault filtering algorithm allows the full diagnosis. This algorithm takes into account flow sharing (data, materials, or energy) between subsystems. The consequence of flow sharing is that a subsystem that operates without fault could reach an erroneous state, not described in $\mathscr{L}^{N_l}$. This problem happens when the subsystem does not receive the prospective service (the flow) of another subsystem linked to it. Flow stopping could be due to failures in another subsystem up or down the line [37]. In order to include this fact in the diagnoser the notion of shared flow sensor (SFS) is introduced.

When a fault has been detected and it has not been eliminated by the filtering algorithm, the structure of diagnoser proposed allows isolating and identifying the fault; at this time a new fault trace of $\mathscr{L}^{f_i}$ is learned by the diagnoser. So, the diagnosis skills of the diagnoser grow over time.

4.1. Architecture for the Diagnostic Method. As it was mentioned in the previous section, the diagnoser is based on a set of identified st-IPNs. The diagnoser also includes color in order to compare languages and detect faults. So the

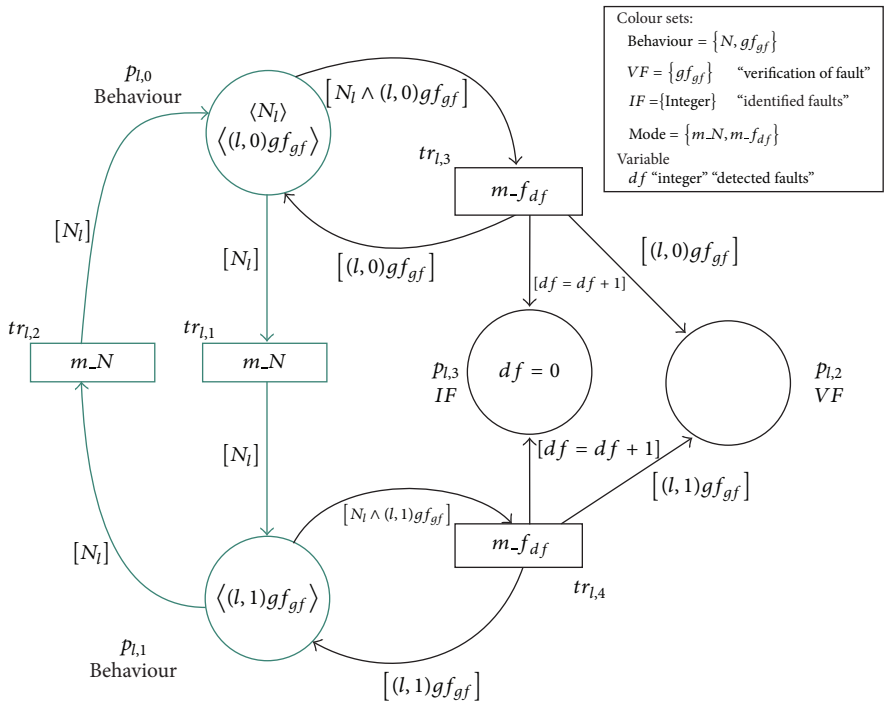

Figure 3: st-DICPN architecture.

diagnoser is a *stochastic timed interpreted Coloured Petri Net to diagnosis* (st-DICPN), which is shown in Figure 3.

The set of places is partitioned into $P_l = P_{LN_l} \cup P_{VF_l} \cup P_{IF_l}$, where P_{LN_l} represents the set of latent nestling places, that is, places with nominal behavior in which a fault can happen; P_{VF_l} is the set of places that verify the detected fault kind; P_{IF_l} is a place that counts the identified faults.

The set of transitions is partitioned into $TR_l = TR_{N_l} \cup TR_{F_l}$, where TR_{N_l} represents the set of normal transitions that fire following the normal language and TR_{F_l} represents the set of fault transitions whose size can be increased each time a fault event is detected; they fire when a fault f_{df} is detected.

The set of colour classes is $C_l = \{\text{Behavior}, VF, IF, \text{Mode}\}$, where Behavior $= \{\langle N_l \rangle, \langle (l,q) gf_{gf_l} \rangle\}$: $\langle N_l \rangle$ is the normal token, $\langle (l,q) gf_{gf_l} \rangle$ is the generic fault token, l stands for the subsystem, q stands for the place, and the subscript gf_l is a fault identification index; $VF = \{\langle (l,q) gf_{gf_l} \rangle\}$; $IF = \{\langle \text{integer} \rangle\}$ is variable token that depends on variable df, and the Mode colour class set includes the colours assigned to transitions, where Mode $= \{m\_N, m\_f_{df}\}$: m_N is "normal mode" and m_f_{df} is "detected fault mode df."

In Figure 4, the part of the net in green colour represents the normal behavior which is identified online from the observed legal sequences. The part of the net in black colour represents the identified fault behavior by means of applying the diagnostic algorithm. This net has a variable structure because the diagnosis process learns the fault languages.

In Figure 4, when a fault is detected on place $p_{l,0}$, $df = 1$, the transition $tr_{1,3}$ fires in m_f_1 mode and then a colour token $\langle (l,0) gf_1 \rangle$ is reached in $p_{l,2}$ (VF place) and an integer $\langle 1 \rangle$ is reached in $p_{l,3}$ (IF place). In this moment, the *fut* function is executed and the arcs and transitions structure increases, so $pre(p_{l,0}, tr_{l,3}) = [(l,0) gf_1; (l,0) gf_{gf}]$, $post(p_{l,0}, tr_{l,3}) = [(l,0) gf_{gf}]$, $post(p_{l,2}, tr_{l,3}) = [(l,0) gf_1]$, $post(p_{l,3}, tr_{l,3}) = [1]$, and the token in transition $tr_{l,3}$ is $[m\_f_1; m\_f_{df}]$.

The incidence matrix entries are represented by vectors [35]. If a fault f_1 is detected in place ($p_{l,0}$), then *Pre* and *Post* matrices will be updated as shown as follows.

Pre

$$\begin{array}{c} \\ p_{l,0} \\ p_{l,1} \\ p_{l,2} \\ p_{l,3} \end{array} \begin{array}{c} \begin{array}{cccc} tr_{l,1} & tr_{l,2} & tr_{l,3} & tr_{l,4} \end{array} \\ \begin{bmatrix} N_l & 0 & \begin{bmatrix} N_l \wedge (l,0)\, gf_1 \\ N_l \wedge (l,0)\, gf_{gf} \end{bmatrix} & 0 \\ 0 & N_l & 0 & N_l \wedge (l,1)\, gf_{gf} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \quad \begin{array}{cccc} m\_N & m\_N & \begin{bmatrix} m\_f_{df} \\ m\_f_1 \end{bmatrix} & [m\_f_{df}] \end{array}. \tag{2}$$

Post

$$\begin{array}{c} \\ p_{l,0} \\ p_{l,1} \\ p_{l,2} \\ p_{l,3} \end{array} \begin{array}{c} \begin{array}{cccc} tr_{l,1} & tr_{l,2} & tr_{l,3} & tr_{l,4} \end{array} \\ \begin{bmatrix} 0 & N_l & (l,0)\, gf_{gf} & 0 \\ N_l & 0 & 0 & (l,1)\, gf_{gf} \\ 0 & 0 & \begin{bmatrix} (l,0)\, gf_1 \\ (l,0)\, gf_{gf} \end{bmatrix} & (l,1)\, gf_{gf} \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \quad \begin{array}{cccc} m\_N & m\_N & \begin{bmatrix} m\_f_{df} \\ m\_f_1 \end{bmatrix} & [m\_f_{df}] \end{array}. \tag{3}$$

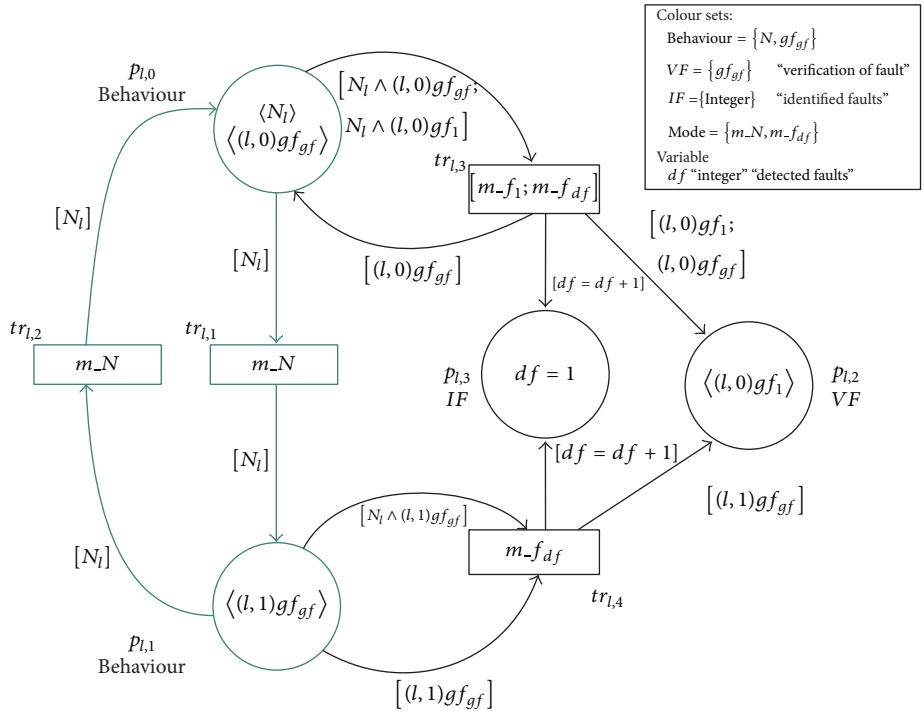

Figure 4: st-DICPN with fault.

Definition 15 (stochastic timed interpreted Coloured Petri Net to diagnosis (st-DICPN)). A st-DICPN for a subsystem l is defined as $stDCQ_l = (stCQ_l, fut)$, where $stCQ_l = (CQ_l, \Omega_l, \delta_l)$ is a st-ICPN defined as in Definition 10 for a subsystem l; $P_l = P_{LN_l} \cup P_{VF_l} \cup P_{IF_l}$; $TR_l = TR_{N_l} \cup TR_{F_l}$; $C_l = \{\text{Behavior}, VF, IF, \text{Mode}\}$. The set of colour domain of places are $cd(P_{LN_l}) = \{N_l, (l,q)gf_{gf_l}\}$, $cd(P_{VF_l}) = \{(l,q)gf_{gf_l}\}$, and $cd(P_{IF_l}) = \{\text{Integer}\}$. The set of colour domain of transitions are $cd(TR_{N_l}) = \{m\_N_l\}$ and $cd(TR_{F_l}) = \{m\_f_{df}\}$. $Pre, Post \in \beta^{|P|\times|TR|}$ are incidence matrices based on Definition 9. The normal transitions labeling function is $\lambda_l : TR_{N_l} \times m\_N \to U_l \cdot \delta_l$. $\delta_l := TR_l \times OM \to f(t_{TR_l \times OM})$ is the transition firing time density function for each OM. The normal places output function is $\varphi_l : R(CQ_l) \times cd(P_{LN_l}) \to Y_l / dY_l$. And $fut : M_l \times Y_l \to TR_l \times Pre_l \times Post_l$ is a function that updates $TR_l, Pre_l, Post_l$.

The incidence matrices are $Pre[p, tr] : cd(tr) \to cd(p)$ and $Post[p, tr] : cd(tr) \to cd(p)$. That is, $Pre[p^{LN}_{l,q}, tr^N_{l,r}]m\_N_l = \langle N_l \rangle$; $Pre[p^{LN}_{l,q}, tr^F_{l,r}]m\_f_{df} = \langle N_l \rangle \wedge \langle (l,q)gf_{gf_l} \rangle$; $Post[p^{LN}_{l,q}, tr^N_{l,r}]m\_N_l = \langle N_l \rangle$; $Post[p^{LN}_{l,q}, tr^F_{l,r}]m_{f_{df}} = \langle (l,q)gf_{gf_l} \rangle$; $Post[p^{VF}_{l,q}, tr^F_{l,r}]m\_f_{df} = \langle (l,q)gf_{gf_l} \rangle$; $Post[p^{IF}_{l,q}, tr^F_{l,r}]m\_f_{df} = \{x+1\}$.

This structure of incidence matrices allows the net evolution under normal conditions or nonpredefined event traces.

4.2. Online Diagnosis Process. This section proposes a procedure that specifies the online work of the diagnoser. This process has five steps and it can be seen in Figure 5.

The first step is the configuration of the system. The system has to be split into subsystems, I/O signals must be defined, and the starting event for each subsystem will be $\omega^0_l = (u_{l,s} y_{l,j})\tau_0$, where $u_{l,s}$, $y_{l,j}$ stand for the starting values of control commands and sensor readings, at time τ_0, and the set of operation modes (o_{om}) has to be stated.

The second step is the observation and learning of the normal behavior. The identification algorithm [33] builds a set of st-IPNs generating the observed language.

The third step is the computing of the initial diagnoser which transforms the st-IPNs into st-DICPNs. The proposed algorithm modifies the net structure as follows.

(i) A normal token $\langle N_l \rangle$ will be added to each at initial place.

(ii) The set of P_{VF_l}, P_{IF_l}, and TR_{F_l} will be added as well as the arcs required to complete the diagnoser architecture (as the one shown in Figure 4).

(iii) To represent generic faults, a coloured fault token, $\langle (l,q)gf_{gf_l} \rangle$, must be added to each $p_{l,q} \in P_{LN_l}$.

The fourth step is the online fault detection and fault isolation. More precisely the process consists of the following.

Initialize the variables $df_l = 0$, $gf_l = 0$, and $\mathscr{L}^{f_{df_l}} = \{\}$ (set of identified fault traces).

Being the current state of the net $p_{l,q} \in P_{LN_l}$ and given an observed event, $\omega^{i*}_l = (u^{i*}_{l,s} y^{i*}_{l,j}) \cdot t^{\text{ev}}_{l,o_{om}}$.

(i) If $Pc_{\Omega_{N_l}}(\omega^{i*}_l) = \omega^{i*}_l$ (see Definition 4), then $\omega^{i*}_l \in \mathscr{L}^{N_l}$. $tr_{l,r} \in TR_{N_l}$ is firing in normal mode with

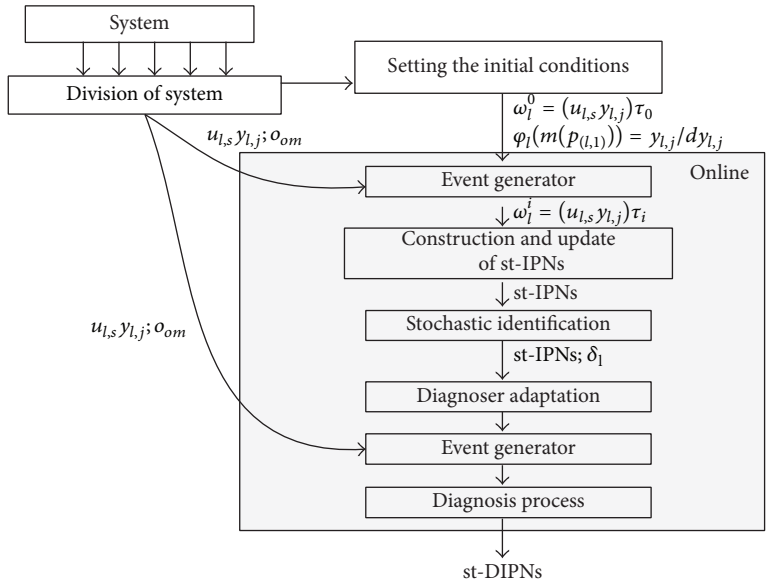

Figure 5: Diagnosis process.

Inputs: $\mathscr{L}^{N_l}$, $st\text{-}DICPN_l$; $\omega_l^i = (u_{l,s} y_{l,j}) \cdot t^{ev}$, for $i = 0, \ldots, k$;
Outputs: IF_l; $\mathscr{L}^{IF_l}$; update $st\text{-}DICPN_l$
Initial Conditions: $\omega_l^0 = (u_{l,s} y_{l,j}) \cdot t^{ev}$; $IF_l = []$; $\mathscr{L}^{f_{df_l}} = \{\}$.
(1) $i = 0$;
(2) for $l = 1, \ldots, c$
(i) $q_l = 1$; $pf_l = 0$; $pf_l = 0$; $f_l = 0$; $df_l = 0$.
(ii) read $u_{l,s}$, $y_{l,j}$ at time τ_0
(iii) $\varphi_l(m(p_{l,q_l})) = y_{l,j}/\vec{0}$; $m(p_{l,q_l}) = \langle N_l \rangle$;
end for
(3) $i = i + 1$;
(4) Wait for a new ω_l^i; read $\omega_l^i = (u_{l,s} y_{l,j})\tau_i$;
(5) $sub = l$
(a) if $\omega_l^i = (u_{l,s} y_{l,j}) \cdot t^{ev} \in \mathscr{L}^{N_{sub}}$ then $m(p_{sub,q_l+1}) = \langle N_{sub} \rangle$ and $\sigma^{N_{sub}} = tr_{sub,1}, \ldots, tr_{sub,i}$.
(b) else $\omega_l^i = (u_{l,s} y_{l,j})$ is a generic fault in $p_{sub,q_{sub}}$; $gf_{sub} = gf_{sub} + 1$;
then a new $tr_{l,r}$ is generated as:
(i) compute a *pre* vector of zeros except in position $p_{sub,q_{sub}}$ which has a value of $N_l \wedge (sub, q_{sub}) gf_{gf_{sub}}$.
(ii) compute a *post* vector of zeros except in position $p_{sub,q_{sub}}$, $p_{sub,VF_{sub}}$ which has a value of $(sub, q_{sub}) gf_{gf_{sub}}$ and position $p_{sub,IF}$ which has a weight of df_l.
(iii) the new $tr_{l,r}$ fire, the net reaches the marking: $m(p_{sub,VF_{sub}}) = (sub, q_{sub}) pf_{pf_{sub}}$; $m(p_{sub,q_{sub}}) = (sub, q_{sub}) gf_{gf_{sub}}$ and $df_l = df_l + 1$. $m(p_{l,IF_l}) = \langle df_l \rangle$.
(iv) $f_{df_{sub}} = f_{df_{sub}} + 1$.
(v) Execute Algorithm 2.
end if
(6) Wait for the next event and return to Step (3)

Algorithm 1: Diagnosis process.

$\lambda_l(tr_{l,r}) = u_{l,s}.f(t_{tr_{l,r},o_{om}})$ and a normal token is placed at $m(p_{l,q+1}) = \langle N_l \rangle$, with $\varphi_l(m(p_{l,q+1})) = y_{l,j}/dy_{l.j}$ and wait for a new ω_l^{i*}.

(ii) Else, if $Pc_{\Omega_{N_l}}(\omega_l^{i*}) = \varepsilon$ then $\omega_l^i \notin \mathscr{L}^{N_l}$, and a fault trace has been detected in l: $df_l = df_l + 1$:
(a) a $tr_{l,r} \in TR_{F_l}$ is firing in f_{df_l} mode, $gf_l = gf_l + 1$;
(b) a generic fault token is reached in $m(p_{l,VF_l}) = m(p_{l,q_l}) = \langle (l, q) gf_{gf_l} \rangle$ and another integer token is reached in $p_{l,IF_l} = \langle df_l \rangle$.

This process is shown in Algorithm 1.

At the end of this step a fault has been detected over a time line, as well as its fault trace.

Input: SFS_r; $\omega_{l,q}^{i*} = (u_{l,s}^{i*} y_{l,j}^{i*}) \cdot t^{ev} \subset \mathscr{L}^{f_l}$; $s\omega_l^p \in \mathscr{L}^{N_l}$ is the expected sequence.
Output: $ISFS_r$.
Initial Conditions: $ISFS_r = \{\}$.
If $P_{\Omega_{N_l}}(y_{l,j}^{i*}) = \varepsilon$ then $\exists i \mid sr_{l,i}^* \in y_{l,j}^* \neq sr_{l,i} \in y_{l,j}$, therefore $sr_{l,i} = sr_{l,i}^{ex}$ is the expected sensor reading.
If $sr_{l,i}^{ex} \in SFS_r$ then
For $l : 1, \ldots, c$
if $\exists sr_{l,n_l} \in ISFS_r$ then, delete $\omega_{l,q}^{i*}$ because is a fault propagation.
else include $sr_{l,i}^{ex}$ in $ISFS_r$
end if
end for
end if
else there is not a fault propagation.
end if

Algorithm 2: Fault filtering.

The fifth step eliminates the fault if it has been generated by a false alarm caused by coupling. Coupling of *st-DICPN*$_l$ and faults propagation ([38]) are generated by interactions among subsystems. Therefore, a fault filtering algorithm is proposed, which is described below.

The algorithm analyzes the shared flow sensor reading set, $SFS_r = \cup sr_{l,n_l}$, where n_l is the number of sensors in subsystem l. Given an identified fault event $\omega_{l,q}^{i*}$ at current time τ_{ac}, $\mathscr{L}_{\text{out}}(\omega_{l,q}^{i*}) = y_{l,j}$, and $y_{l,j} = sr_{l,1} \cdots sr_{l,n_l}$, the algorithm compares the values of the shared flow sensors to decide if the fault is due to propagation (Algorithm 2).

Once Algorithm 2 filters each fault, the st-DICPN architecture is updated and *fut* function updates the st-DICPN as follows.

(i) Detected fault colour tokens are added in the verification place.

(ii) The integer colour token is updated.

(iii) A colour token, m_f_{df}, is added in the fired fault transition, the same way the colour token $(l,q)gf_{gf_l}$ is added in the involved arcs.

(iv) The fault has been isolated and then $s\omega_l^{i*}$ is included in the set of identified fault trace, $s\omega_l^i \subset \mathscr{L}^{f_{df_l}}$.

Therefore, the fault trace is learned by the st-DICPN and the fault is identified.

This trace contains information about the faulty subsystem as well as the unexpected behavior and it is possible to distinguish the subsystem that made the fault as well as the faulty signals.

4.3. Properties. The structure of a $stDCQ_l$ is variable because the number of coloured tokens, as well as the number of transitions, grows with each new fault trace. Nevertheless, the size of each $stDCQ_l$ is bounded by the number of sensors.

Let σ^{N_l} be a transition firing sequence, such that, for all $tr_{l,r} \in \sigma^{N_l}, tr_{l,r} \in TR_{N_l}$; let s be the generated event sequence by σ^{N_l} such that $s \in \mathscr{L}^{N_l}(stDCQ_l)$, where $s = \omega_l^0, \ldots, \omega_l^k$; if $\varphi(m(p_{l,0})) = \varphi(m(p_{l,k}))$, then $s^* \in \mathscr{L}^{N_l}(stDCQ)$; that is, s is a cycle.

Given a sequence of events $s\omega_l^{i*}$, such that $s \in \mathscr{L}^{N_l}$, $s = \omega_l^0, \ldots, \omega_l^k$, at times $\tau_1 \leq \cdots \leq \tau_k$ and $s\omega_{l,q}^{i*} \in \mathscr{L}^{IF_l}$, then the transition firing sequence σ^{F_l} that generates $s\omega_l^{i*}$ is $\sigma^{F_l} = tr_{l,1}, \ldots, tr_{l,k}, tr_{l,k+1}, tr_{l,k}$, where $tr_{l,1}, \ldots, tr_{l,k} \in TR_{N_l}$, $tr_{l,k+1} \in TR_{F_l}$, and $M_{l,0}\lceil \sigma^{F_l} > M_{l,k}$, where $m(p_{l,VF_l}) = \langle (l,q)gf_{gf_l} \rangle; m(p_{l,k}) = \langle N \rangle, \langle (l,q)gf_{gf_l} \rangle. (p_{l,k} \in P_{LN_l})$. That is, the fault language $\mathscr{L}^{IF_l}(stDCQ) = \omega_l^0, \ldots, \omega_l^k \omega_{l,q}^{i*} \omega_l^{k+1}$ reaches a normal token in a latent nesting place, from which the system can evolve when the fault has been repaired.

4.4. Detectability. The analysis of detectability presented in this proposal is based on language theory and prior works on temporal observability. Detectability proves if the system can detect the occurrence of a fault in a finite number of observable events.

Based on Definition 13 n-detectability is defined as follows.

Definition 16 (n-detectable). Let $s\omega_l^{i*}$ be an event sequence ending with a fault event and let t be an observable event sequence after $s\omega_l^{i*}$, with $n = |t|$. Given $s_1 = s\omega_l^{i*} t$, $\omega_l^{i*} = (u_{l,s} y_{l,j}) \cdot t^{\text{ev}}$ is n-detectable if $\exists \omega_l^i \in t$ such that (i) $Pc_{\Omega_{N_l}}(u_{l,s}) \notin \mathscr{L}_{\text{in}}^{N_l} \wedge Pc_{\Omega_{N_l}}(y_{l,j}) \in \mathscr{L}_{\text{out}}^{N_l}$ or (ii) $Pc_{\Omega_{N_l}}(u_{l,s}) \in \mathscr{L}_{\text{in}}^{N_l} \wedge Pc_{\Omega_{N_l}}(y_{l,j}) \notin \mathscr{L}_{\text{out}}^{N_l}$ or (iii) $Pc_{\Omega_{N_l}}(u_{l,s}) \in \mathscr{L}_{\text{in}}^{N_l} \wedge Pc_{\Omega_{N_l}}(y_{l,j}) \in \mathscr{L}_{\text{out}}^{N_l}$ but $t^{\text{ev}} \notin [a,b]$ in n steps.

Necessary and Sufficient Conditions for Detectability

Theorem 17 (detectability). *Given $s\omega_l^p$, an event sequence ending with a fault event, ω_l^p, let t_1, t_2 be two observable events sequences, where $s, t_1 \in \mathscr{L}^{N_l} \wedge t_1, t_2 \in \mathscr{L}/s$; ω_l^p is detectable if $Pt_{\Omega_{N_l}}(st_1) \neq Pt_{\Omega_{N_l}}(s\omega_l^p t_2)$.*

Proof. As $t_1 \in \mathscr{L}^{N_l}/s \wedge t_1 \in \mathscr{L}^{N_l}$ then $st_1 \in \mathscr{L}^{N_l}$. And as ω_l^p is a fault event then $s\omega_l^p t_2$ is a fault trace of the $stDCQ_l$, $(\mathscr{L}^{f_i} = s\omega_l^p t_2)$. Give $s = \omega_l^0, \ldots, \omega_l^k$ and $t_2 = \omega_l^q, \ldots, \omega_l^x$.

Necessary Condition. If ω_l^p is detectable then $Pt_{\Omega_{N_l}}(st_1) \neq Pt_{\Omega_{N_l}}(s\omega_l^p t_2)$.

As $t_1 \in \mathscr{L}^{N_l}$ then, for all $\omega_l^i \in t_1, \omega_l^i \in \Omega_{N_l}$. If ω_l^p is detectable then $\exists \omega_l^{i*} \in t_2 \mid \omega_l^{i*} \in \Omega_{o_l} \wedge \omega_l^{i*} \notin \Omega_{N_l}$ (see Definition 16); therefore $t_1 \neq t_2$ and $Pt_{\Omega_{N_l}}(st_1) \neq Pt_{\Omega_{N_l}}(s\omega_l^p t_2)$.

Sufficient Condition. If $Pt_{\Omega_{N_l}}(st_1) \neq Pt_{\Omega_{N_l}}(s\omega_l^p t_2)$, then ω_l^p is detectable.

$stDCQ_l$ represents only observed languages. If st_1 and $s\omega_l^p t_2$ are represented in a $stDCQ_l$, then st_1 and st_2 must be observed event sequences.

Assuming that $Pt_{\Omega_{N_l}}(st_1) = Pt_{\Omega_{N_l}}(s\omega_l^p t_2)$, then $t_2 \in \mathscr{L}^{N_l}$ and ω_l^p is not detectable. □

5. Application Case

As application case, it has used the centralized air heating system, AHS; it has been identified as a set of st-IPNs in [33].

The system includes three heating subsystems. Each heating subsystem has a fan creating an air flow that is heated with hot water. The water flow is controlled by pump-valve systems. Moreover, there are a central heater providing hot water to each heating subsystem and two valves (v_{c1} and v_{c2}) controlling the water flow through the whole system. The system can be split into five subsystems (1, 2, 3, 4, and 5). Subsystems 3, 4, and 5 are the local heaters, subsystem 2 is the distribution subsystem (v_{c1} and v_{c2}), and subsystem 1 is the main heating subsystem (heater, main pump (p_m), and reflux valve (v_r)). The heater works in three modes (0, 1, 2), each state is defined by the number of resistances it has activated; so modes 0, 1, and 2 will represent no activation of resistance, activation of h_1, and activation of both h_1 and h_2.

The system has a set of sensors. Each subsystem l includes a flow sensor (F_l) that measures the presence or absence of flow. Nevertheless, flow level is affected when other subsystems are activated. So, a software sensor (NF_l) is designed to measure the deviation over a normal operation flow taking into account the activation of other subsystems. The system also includes binary temperature sensors, pressure sensors, and a position sensor for valve Vc_1.

The initial conditions in each subsystem are

$$\begin{aligned}
\omega_1^0 &= \left[\overline{v}_r \overline{p}_g \overline{v}_g \overline{h}_1 \overline{h}_2, T_o \overline{T}_1 \overline{T}_2 \overline{F}_1\right] = \left(u_{1,0} y_{1,8}\right);\\
\omega_2^0 &= \left[\overline{v}_{c1} \overline{v}_{c2}, \overline{P}_o \overline{F}_2 \overline{NF}_2\right] = \left(u_{2,0} y_{2,0}\right);\\
\omega_3^0 &= \left[\overline{p}_3 \overline{v}_3, \overline{F}_3 \overline{NF}_3\right] = \left(u_{3,0} y_{3,0}\right);\\
\omega_4^0 &= \left[\overline{p}_4 \overline{v}_4, \overline{F}_4 \overline{NF}_4\right] = \left(u_{4,0} y_{4,0}\right);\\
\omega_5^0 &= \left[\overline{p}_5 \overline{v}_5, \overline{F}_5 \overline{NF}_5\right] = \left(u_{5,0} y_{5,0}\right).
\end{aligned} \tag{4}$$

Table 1: Identified density functions for AHS system.

Transition	o_8	o_{12}
$tr_{1,1}$	$N(16.6, 0.51)$	
$tr_{1,2}$	27	
$tr_{1,3}$		$N(115.1, 0.32)$
$tr_{1,4}$		12
$tr_{1,5}$	$N(593, 1.5)$	
$tr_{1,6}$	2	
$tr_{1,7}$	$N(91.5, 1.2)$	
$tr_{1,8}$	9	
$tr_{1,9}$	1	
$tr_{3,1}$		$N(158.08, 2.52)$
$tr_{3,2}$		$N(26.93, 1.201)$
$tr_{3,3}$	$N(565.06, 1.201)$	
$tr_{3,4}$	3	
$tr_{3,5}$	1	

The system globally starts with the external event "*Son*"; the heating subsystems are locally started with events "*Ca3,*" "*Ca4,*" and "*Ca5.*" These events are external events that change the controller strategy. Each combination of external events generates a system operation mode. For example, $o_{12} = Son, Ca3, \overline{Ca4}, \overline{Ca5}$.

We have simulated the system, including some changes in the operation modes: o_0–o_8–o_{12}–o_8–o_0. That sequence means that "*Son*" works from time 1 to 85 and "*Ca3*" works from time 15 to 75.

Step 1. The identified normal languages for each subsystem are:

$$\begin{aligned}
\mathscr{L}^{N_1} &= \left(u_{1,0} y_{1,8}\right)\left(u_{1,30} y_{1,9}\right)\left(u_{1,30} y_{1,13}\right)\left(u_{1,15} y_{1,13}\right)\\
&\quad \left(u_{1,15} y_{1,15}\right)\left(u_{1,30} y_{1,15}\right)\left(u_{1,30} y_{1,13}\right)\left(u_{1,0} y_{1,9}\right)\\
&\quad \left(u_{1,0} y_{1,8}\right) \quad \text{at times } \tau_0, \tau_1, \tau_2, \tau_3, \tau_5, \tau_8, \tau_9, \tau_{12}, \tau_{13},\\
\mathscr{L}^{N_2} &= \left(u_{2,0} y_{2,0}\right) \quad \text{at time } \tau_0,\\
\mathscr{L}^{N_3} &= \left(u_{3,0} y_{3,0}\right)\left(u_{3,3} y_{3,2}\right)\left(u_{3,3} y_{3,3}\right)\left(u_{3,3} y_{3,2}\right)\left(u_{3,3} y_{3,0}\right)\\
&\quad \left(u_{3,0} y_{3,0}\right) \quad \text{at times } \tau_0, \tau_4, \tau_6, \tau_7, \tau_{10}, \tau_{11},\\
\mathscr{L}^{N_4} &= \left(u_{4,0} y_{4,0}\right) \quad \text{at time } \tau_0,\\
\mathscr{L}^{N_5} &= \left(u_{5,0} y_{5,0}\right) \quad \text{at time } \tau_0.
\end{aligned} \tag{5}$$

The identified transition firing time density functions for each o_{om}, (δ_l) are shown in Table 1.

Step 2. The set of the flow sensors is $SFS_r = \{sr_{1,4}, sr_{2,1}, sr_{2,2}, sr_{3,1}, sr_{4,1}, sr_{5,1}\}$.

Generic fault in all places is assumed, in all involved subsystems of the identified operation modes sequence.

Step 3. Assuming a fault in the main pump at time 60 min and a fault in the heater at time 70 min at subsystem 1, the

Table 2: Detected faults.

Fault	Subsystem	Place	Expected sr	Fault sr	Meaning	$sr_{l,i}^{ex}$	$ISFS_r$
$(1,5)gf_1$	1	5	[1111]	[1110]	$T_oT_1T_2\overline{F}_1$	$sr_{1,4}$	$sr_{1,4}$
$(3,3)gf_1$	3	3	[10]	[00]	$\overline{F}_3\overline{NF}_3$	$sr_{3,1}$	$sr_{1,4}, sr_{3,1}$
$(1,5)gf_2$	1	5	[1111]	[1101]	$T_oT_1\overline{T}_2\overline{F}_1$	$sr_{1,3}$	

observed event sequence at subsystem 1 is $s_1 = (u_{1,0}y_{1,8})(u_{1,30}y_{1,9})(u_{1,30}y_{1,13})(u_{1,15}y_{1,13})(u_{1,15}y_{1,15})(u_{1,15}y_{1,13})(u_{1,30}y_{1,8})(u_{1,0}y_{1,8})(u_{1,15}y_{1,9})$, at times $\tau_0, \tau_1, \tau_2, \tau_3, \tau_5, \tau_7, \tau_9, \tau_{11}, \tau_{12}$ with $\delta = \{0, 17, 27, 114, 12, 454, 268, 39, 95\}$ and at subsystem 3: $s_2 = (u_{3,0}y_{3,0})(u_{3,3}y_{3,2})(u_{3,3}y_{3,3})(u_{3,3}y_{3,0})(u_{3,0}y_{3,0})$, at times $\tau_0, \tau_4, \tau_6, \tau_8, \tau_{10}$ with $\delta = \{0, 159, 28, 520, 46\}$.

Step 4

Inputs. $\mathscr{L}^{N_l}$, $st\text{-}DICPN_l$.

Initial Conditions. Consider $\omega_1^0 = (u_{1,0}y_{1,8})\tau_0$; $\omega_3^0 = (u_{3,0}y_{3,0})\tau_0$; $\varphi_1(m(p_{1,1})) = y_{1,8}/0000$; $m(p_{1,1}) = \langle N_1\rangle$; $\varphi_3(m(p_{3,1})) = y_{3,0}/00$; $m(p_{3,1}) = \langle N_3\rangle$.

In τ_1, $\omega_1^1 = (u_{1,30}y_{1,9}) \cdot 17$; $Pt_{\Omega_{N_1}}(\omega_1^1) = (u_{1,30}y_{1,9}) \cdot 17$; then $\lambda_1(tr_{1,1}) = u_{1,30} \cdot f(t_{tr_{1,1},8})$ and $m(p_{1,2}) = \langle N_1\rangle$ with $\varphi_1(m(p_{1,2})) = y_{1,9}/0001$.

In τ_2, $\omega_1^2 = (u_{1,30}y_{1,13}) \cdot 27$; $Pt_{\Omega_{N_1}}(\omega_1^1) = (u_{1,30}y_{1,13}) \cdot 27$; then $\lambda_1(tr_{1,2}) = u_{1,30} \cdot f(t_{tr_{1,2},8})$ and $m(p_{1,3}) = \langle N_1\rangle$ with $\varphi_1(m(p_{1,3})) = y_{1,13}/0100$.

In τ_3, $\omega_1^3 = (u_{1,15}y_{1,13}) \cdot 114$; $Pt_{\Omega_{N_1}}(\omega_1^3) = (u_{1,15}y_{1,13}) \cdot 114$; then $\lambda_1(tr_{1,3}) = u_{1,15} \cdot f(t_{tr_{1,3},12})$ and $m(p_{1,4}) = \langle N_1\rangle$ with $\varphi_1(m(p_{1,4})) = y_{1,13}/0000$.

In τ_4, $\omega_3^4 = (u_{3,3}y_{3,2}) \cdot 159$; $Pt_{\Omega_{N_3}}(\omega_3^4) = (u_{3,3}y_{3,2}) \cdot 159$; then $\lambda_3(tr_{3,1}) = u_{3,3} \cdot f(t_{tr_{3,1},12})$ and $m(p_{3,2}) = \langle N_3\rangle$ with $\varphi_3(m(p_{3,2})) = y_{3,2}/10$.

In τ_5, $\omega_1^5 = (u_{1,15}y_{1,15}) \cdot 12$; $Pt_{\Omega_{N_1}}(\omega_1^5) = (u_{1,15}y_{1,15}) \cdot 12$; then $\lambda_1(tr_{1,4}) = u_{1,15} \cdot f(t_{tr_{1,4},12})$ and $m(p_{1,5}) = \langle N_1\rangle$ with $\varphi_1(m(p_{1,5})) = y_{1,15}/0010$.

In τ_6, $\omega_3^6 = (u_{3,3}y_{3,3}) \cdot 28$; $Pt_{\Omega_{N_1}}(\omega_3^6) = (u_{3,3}y_{3,3}) \cdot 28$; then $\lambda_3(tr_{3,2}) = u_{1,3} \cdot f(t_{tr_{3,2},12})$ and $m(p_{3,3}) = \langle N_3\rangle$ with $\varphi_3(m(p_{3,3})) = y_{3,3}/01$.

In τ_7, $\omega_1^7 = (u_{1,15}y_{1,13}) \cdot 454$; $Pt_{\Omega_{N_1}}(\omega_1^7) = \varepsilon$; then ω_1^7 is a generic fault at the subsystem 1; $t_{\text{proy}} = 454$; a new fault mode m_f_1 is added in $tr_{1,10} \in TR_{F_1}$, with $pre = [0000\langle(1,5)gf_1 \wedge N_1\rangle 000000]$, $post = [0000\langle(1,5)gf_1\rangle 0000\langle(1,5)gf_1\rangle\langle x\rangle]$ and it adds a fault token to $m(p_{1,vf_1}) = m(p_{1,5}) = \langle(1,5)gf_1\rangle$ and $m(p_{1,if_1}) = \langle 1\rangle$. This fault is added in Table 2.

Algorithm 2 *Execution*. Consider $sr_{l,i}^{ex} = sr_{1,4}$ and $sr_{1,4} \in SFS_r$.

In τ_8, $\omega_3^8 = (u_{3,3}y_{3,0}) \cdot 520$; $Pt_{\Omega_{N_3}}(\omega_3^8) = \varepsilon$; then ω_3^8 is a generic fault at the subsystem 3; a new fault mode m_f_1 is added in $tr_{3,6} \in TR_{F_3}$, with $pre = [00\langle(3,3)gf_1 \wedge N_3\rangle 0000]$, $post = [00\langle(3,3)gf_1\rangle 00\langle(3,3)gf_1\rangle\langle x\rangle]$ and it adds a fault token to $m(p_{3,vf_3}) = m(p_{3,3}) = \langle(3,3)gf_1\rangle$ and $m(p_{3,if_3}) = \langle 1\rangle$. This fault is added in Table 2.

Algorithm 2 *Execution*. Consider $sr_{l,i}^{ex} = sr_{3,1}$ and $sr_{3,1} \in SFS_r$; then the detected generic fault is a propagation fault and therefore is eliminated.

In τ_9, $\omega_1^9 = (u_{1,15}y_{1,9}) \cdot 268$; $Pt_{\Omega_{N_1}}(\omega_1^9) = \varepsilon$; then ω_1^9 is a generic fault at the subsystem 1; $t_{\text{proy}} = 454 + 268$; $tr_{1,10} \in TR_{F_1}$ is fired in mode m_f_2 with $pre = [0000\langle(1,5)gf_1 \wedge N_1; (1,5)gf_2 \wedge N_1\rangle 000000]$, $post = [0000\langle(1,5)gf_1; (1,5)gf_2\rangle 0000\langle(1,5)gf_1; (1,5)gf_2\rangle\langle x\rangle]$ and $m(p_{1,vf_1}) = m(p_{1,5}) = \langle(1,5)gf_2\rangle$ and $m(p_{1,if_1}) = \langle 2\rangle$. This fault is added in Table 2.

Algorithm 2 *Execution*. Consider gf_2 is not a fault propagation.

Therefore $\omega_1^0\omega_1^1\omega_1^2\omega_1^3\omega_1^5\omega_1^7$ is a fault trace; that is, $\omega_1^0\omega_1^1\omega_1^2\omega_1^3\omega_1^5\omega_1^7 \subset \mathscr{L}^{f_1}$.

Interpretation. Based on Table 2, $(1,5)gf_1$ is a detected fault that has happened in subsystem 1. It has been detected because there is no flow. Therefore the fault may have occurred due to a main pump fault since the sensor $s_{1,4}$ measures the flow after the pump. $(3,3)gf_1$ is not fault and $(1,5)gf_2$ is a fault in the heater.

6. Conclusions

The presented method detects and isolates faults in stochastic DES without previous behavior model. To achieve this goal a diagnoser that represents the system languages (normal and fault behaviors) is defined. The diagnoser is a deterministic language generator that avoids the combinatorial explosion of states. The proposed fault identification works online and it uses PN as the language generator. Moreover, the proposed structure generates stochastic timed languages. The st-DICPN has a structure without deadlocks and allows a detectability test based on the timed projection operation, which is defined, too. The diagnosis methodology guarantees the detection and isolation of nonmodeled faults. It has been shown that the system learns the fault languages, so the next time the same fault happens, the diagnoser will be able to conclude that the same fault has occurred.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by a grant from the Universidad del Cauca, Reference 2.3-31.2/05 2011.

References

[1] S. Jiang and R. Kumar, "Failure diagnosis of discrete-event systems with linear-time temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 49, no. 6, pp. 934–945, 2004.

[2] J. Zaytoon and S. Lafortune, "Overview of fault diagnosis methods for Discrete Event Systems," *Annual Reviews in Control*, vol. 37, no. 2, pp. 308–320, 2013.

[3] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[4] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis, "Failure diagnosis using discrete-event models," *IEEE Transactions on Control Systems Technology*, vol. 4, no. 2, pp. 105–124, 1996.

[5] A. P. Estrada-Vargas, E. López-Mellado, and J.-J. Lesage, "A comparative analysis of recent identification approaches for discrete-event systems," *Mathematical Problems in Engineering*, vol. 2010, Article ID 453254, 21 pages, 2010.

[6] M. P. Cabasino, A. Giua, and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," *Automatica*, vol. 46, no. 9, pp. 1531–1539, 2010.

[7] J. Prock, "A new technique for fault detection using petri nets," *Automatica*, vol. 27, no. 2, pp. 239–245, 1991.

[8] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard, "Fault detection and diagnosis in distributed systems: an approach by partially stochastic Petri nets," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 8, no. 2, pp. 203–231, 1998.

[9] C. N. Hadjicostis and G. C. Verghese, "Monitoring discrete event systems using PETri net embeddings," in *Application and Theory of Petri Nets 1999*, vol. 1639 of *Lecture Notes in Computer Science*, pp. 188–207, Springer, Berlin, Germany, 1999.

[10] A. Benveniste, E. Fabre, S. Haar, and C. Jard, "Diagnosis of asynchronous discrete-event systems: a net unfolding approach," *IEEE Transactions on Automatic Control*, vol. 48, no. 5, pp. 714–727, 2003.

[11] S. Genc and S. Lafortune, "Distributed diagnosis of discrete-event systems using petri nets," in *Applications and Theory of Petri Nets 2003*, W. Aalst and E. Best, Eds., vol. 2679 of *Lecture Notes in Computer Science*, pp. 316–336, Springer, Berlin, Germany, 2003.

[12] S. Gene and S. Lafortune, "Distributed diagnosis of place-bordered petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 2, pp. 206–219, 2007.

[13] A. Giua and C. Seatzu, "Fault detection for discrete event systems using Petri nets with unobservable transitions," in *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC '05)*, pp. 6323–6328, December 2005.

[14] A. Ramírez-Treviño, E. Ruiz-Beltrán, I. Rivera-Rangel, and E. López-Mellado, "Online fault diagnosis of discrete event systems. A petri net-based approach," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 31–39, 2007.

[15] M. Dotoli, M. P. Fanti, A. M. Mangini, and W. Ukovich, "On-line fault detection in discrete event systems by Petri nets and integer linear programming," *Automatica*, vol. 45, no. 11, pp. 2665–2672, 2009.

[16] M. P. Fanti, A. M. Mangini, and W. Ukovich, "Fault detection by labeled petri nets in centralized and distributed approaches," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 2, pp. 392–404, 2012.

[17] F. Basile, P. Chiacchio, and G. de Tommasi, "An efficient approach for online diagnosis of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 54, no. 4, pp. 748–759, 2009.

[18] M. Roth, J.-J. Lesage, and L. Litz, "The concept of residuals for fault localization in discrete event systems," *Control Engineering Practice*, vol. 19, no. 9, pp. 978–988, 2011, Proceedings of the 2nd IFAC Workshop on Dependable Control of Discrete Systems (DCDS '09).

[19] M. Roth, S. Schneider, J.-J. Lesage, and L. Litz, "Fault detection and isolation in manufacturing systems with an identified discrete event model," *International Journal of Systems Science*, vol. 43, no. 10, pp. 1826–1841, 2012.

[20] C.-H. Kuo and H.-P. Huang, "Failure modeling and process monitoring for flexible manufacturing systems using colored timed petri nets," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 3, pp. 301–312, 2000.

[21] Y. Li, T. Melliti, and P. Dague, "A colored petri nets model for the diagnosis of semantic faults of bpel services," in *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE '09)*, Paris, France, June 2009.

[22] A. S. Staines, "A compact cpn representation for embedded and control systems fault diagnosis and recovery," in *Proceedings of the 8th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS '09)*, pp. 78–83, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wis, USA, 2009.

[23] E. García, L. Rodriguez, F. Morant, A. Correcher, and E. Quiles, "Latent nestling method: a new fault diagnosis methodology for complex systems," in *Proceedings of the 34th Annual Conference of the IEEE Industrial Electronics Society (IECON '08)*, pp. 253–258, November 2008.

[24] A. Ramírez-Treviño, E. Ruiz-Beltrán, I. Rivera-Rangel, and E. López-Mellado, "Diagnosability of discrete event systems. A Petri Net based approach," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '04)*, vol. 1, pp. 541–546, May 2004.

[25] A. Ramírez-Treviño, E. Ruiz-Beltrán, J. Arámburo-Lizárraga, and E. López-Mellado, "Structural diagnosability of des and design of reduced Petri net diagnosers," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 42, no. 2, pp. 416–429, 2012.

[26] M. P. Cabasino, A. Giua, and C. Seatzu, "Diagnosability of discrete-event systems using labeled petri nets," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 144–153, 2014.

[27] M. P. Cabasino, A. Giua, L. Marcias, and C. Seatzu, "A comparison among tools for the diagnosability of discrete event systems," in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE '12)*, pp. 218–223, August 2012.

[28] L. Yao, L. Feng, and B. Jiang, "Fault diagnosis and fault tolerant control for non-gaussian singular time-delayed stochastic distribution systems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 937583, 9 pages, 2014.

[29] C. Girault and R. Valk, *Petri Nets for Systems Engineering—A Guide to Modeling, Verification, and Applications*, Springer, Berlin, Germany, 2003.

[30] T. Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[31] M. Dotoli, M. P. Fanti, and A. Mangini, "Real time identification of discrete event systems using Petri nets," *Automatica*, vol. 44, no. 5, pp. 1209–1219, 2008.

[32] K. Hernández and M. Meda-Campana, "Fault diagnosis using petri nets. A case study," in *Proceedings of the 10th Latin American and Caribbean Conference for Engineering and Technology*, July 2012.

[33] D. M. Muñoz, A. Correcher, E. García, and F. Morant, "Identification of stochastic timed discrete event systems with st-IPN," *Mathematical Problems in Engineering*, vol. 2014, Article ID 835312, 21 pages, 2014.

[34] J.-I. Latorre-Biel, E. Jiménez-Macías, M. P. de la Parte, J. Blanco-Fernández, and E. Martínez-Cámara, "Control of discrete event systems by means of discrete optimization and disjunctive colored PNs: application to manufacturing facilities," *Abstract and Applied Analysis*, vol. 2014, Article ID 821707, 16 pages, 2014.

[35] C. Girault and R. Valk, *Petri Nets for System Engineering: A Guide to Modeling, Verification, and Applications*, Springer, New York, NY, USA, 2001.

[36] M. P. Cabasino, A. Giua, S. Lafortune, and C. Seatzu, "A new approach for diagnosability analysis of Petri nets using verifier nets," *IEEE Transactions on Automatic Control*, vol. 57, no. 12, pp. 3104–3117, 2012.

[37] E. Garcia, F. Morant, R. Blasco-Gimenez, A. Correcher, and E. Quiles, "Centralized modular diagnosis and the phenomenon of coupling," in *Proceedings of the 6th International Workshop on Discrete Event Systems*, pp. 161–168, 2002.

[38] S. Abdelwahed, G. Karsai, N. Mahadevan, and S. C. Ofsthun, "Practical implementation of diagnosis systems using timed failure propagation graph models," *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 2, pp. 240–247, 2009.