Fault Diagnosis of Discrete Event Systems Using Components Fault-Free models

Moamar Sayed-Mouchaweh, A. Philippot, V. Carre-Menetrier, Bernard Riera

Fault Diagnosis of Discrete Event Systems Using Components Fault-Free models

M. Sayed-Mouchaweh, A. Philippot, V. Carré-Ménétrier, Bernard Riera

Université de Reims Champagne-Ardenne - Centre de Recherche en STIC (URCA-CReSTIC)
Moulin de la Housse B.P. 1039, 51687 REIMS Cedex 2, FRANCE,
({moamar.sayed-mouchaweh, alexandre.philippot, veronique.carre, bernard.riera}@univ-reims.fr)

Abstract: This paper presents a Boolean discrete event model-based approach for Fault Detection and Isolation of manufacturing systems. This approach considers a system as a set of components composed of discrete actuators and their associated discrete sensors. Each component model is only aware of its local desired, fault-free, behavior. The occurrence of any fault entailing the violation of the desired behavior is detected and the potential responsible candidates are isolated using event sequences, time delays between correlated events and state conditions, characterized by sensors readings and control signals issued by the controller. An application example is used to illustrate the approach.

Keywords: Discrete-Event Systems, Decentralized approaches, Diagnosis, Automata, Manufacturing System.

1. INTRODUCTION

The complexity of man-made systems, such as communication networks, manufacturing systems, electric power systems, …, increases rapidly with the course of time. It results from the large number of subcomponents of these systems and the large volume of information flow. This increasing complexity enhances the probability of unpredictable faults and failures.

The basic idea of Fault Detection and Isolation (FDI) is to collect sequences of observations (or symptoms), in order to decide whether or not a system is working normally (fault detection). Then if a fault is detected, FDI reports (fault isolation) which fault has occurred (deterministic diagnosis) or the most likely to have occurred (probabilistic diagnosis). Each fault that can result in a certain symptom, or sequence of observations, is considered as a possible fault candidate.

Generally, the FDI approaches are divided into model-reasoning and model-based approaches. The model-reasoning approaches (Devillez et al. 2004, Isermann 1997) construct a model about the system behavior based on an initial human experience, e.g. expert systems, on a set of historical data, e.g. pattern recognition and signal processing methods, etc. The model-based approaches (Cordier et al. 2007, Darkhovski et al. 2003, Hadjicostis 2005, Rozé et al. 2002, Patton el al. 2000, Sampath et al. 1995, Wang et al. 2005) establish a mathematical or analytical model about the behavior of a system. The model can contain the normal or nominal behavior (fault-free behavior) or the normal behavior as well as the system behavior for a predefined set of faults. The model may be quantitative, expressed in e.g. differential, difference equations, or transfer functions, etc., or a qualitative model, e.g. a finite-state automaton, a set of logic expressions, a combination of both, a Petri Net, etc.

The principal advantage of approaches using normal and faulty behaviors is the precision of the fault isolation. However, integrating the system behavior in response to a predefined set of faults increases exponentially the model size. In addition, only predefined faults can be diagnosed. This disadvantage can be avoided using a fault-free model. However, the fault isolation cannot be as precise as the one using normal and faulty behaviors.

Performing the diagnosis of large scale DES by using a global model is unrealistic. In addition, this type of systems is naturally distributed, i.e., they are composed of several subsystems possessing their own local information. Thus in this paper, we propose a distributed fault-free-model-based approach to diagnose plant faults of large scale DES. The global model is described by its components fault-free models. Each component is composed of an actuator and its associated sensors. These models are available in a library and represented as Boolean DES models. Each behavior which does not correspond to a normal one is considered as a faulty behavior. The components' elements (actuators or sensors) responsible of this faulty behavior are considered as potential fault candidates.

The paper is structured as follows. In section 2, the proposed approach is presented. In section 3, a manufacturing system is used to illustrate the approach. The last section concludes the paper and presents future research directions.

2. MODEL-BASED FDI APPROACH

2.1 System components Boolean models

We use Boolean DES (BDES) modeling, introduced in (Wang 2000), to model the equipments (sensors and actuators) behavior of the system. The system model G consists of n local models: $G^1,\ldots, G^n$, each one owns its local observable events responsible of a restricted area of the

process. The model $G = (\Sigma, Q, Y, \delta, h, q_0)$ is represented as Moore automaton and $L = L(G)$ denotes its corresponding prefixed closed language. Σ is a set of finite events and it includes the observable and unobservable events. Q is the set of states, Y is the output space, $\delta: \Sigma^* \times Q \rightarrow Q$ is the state transition function and Σ^* is the set of all event sequences of the language $L(G)$. The transition function $\delta(\sigma, q)$ provides the set of possible next states if σ occurs at q. $h: Q \rightarrow Y$ is the output function. $h(q)$ is the observed output at q. q_0 is the initial state. Let $\Sigma_\Pi = \{\Pi_{F_1}, \Pi_{F_2}, ..., \Pi_{F_r}\}$ be the set of fault partitions. Each fault partition corresponds to some kind of faults in an equipment element (sensor or actuator). It consists of the set of faults which has the same effect according to either the configuration or maintaining procedure. We assume that at most one fault may occur at a time.

(Balemi et al. 1993) defined controllable events $\Sigma_c \subseteq \Sigma$ as controller's outputs sent to actuators, and uncontrollable events $\Sigma_u \subseteq \Sigma$ as the controller's inputs coming from sensors. $(\Sigma_o = \Sigma_c \cup \Sigma_u) \subset \Sigma$ is the set of observable events. Typically, the observable events in a system are one of the following: enabled or disabled commands issued by the controller and changes of sensor readings. The unobservable events are failure events or other events which cause changes in the system state not recorded by sensors.

Let G^i and its corresponding prefixed closed language, $L^i = L(G^i)$, be the local model of the restricted area of the system observed by this model. $G^i = (\Sigma^i, Q^i, Y^i, \delta^i, h^i, q_0^i)$ is represented as Moore automaton. $\Sigma_0^i = \Sigma_c^i \cup \Sigma_u^i$ is the set of local observable events by G^i and $\Sigma_o^i \subset \Sigma_o$. The other notations have the usual definition but for the restricted area observed by G^i. The model G is the synchronous composition of all the local models: $G = G^1 \| G^2 \| ... \| G^n$. G observes the system by one global projection function or mask, $P_L: \Sigma^* \cup \{\varepsilon\} \rightarrow \Sigma_o^*$. Thus P erases the unobservable events in an event sequence. The inverse projection function is defined as: $P_L^{-1}(u) = \{s \in L: P_L(s) = u\}$. It establishes all the event sequences producing the same observable event sequence u. Similarly, a local projection function can be defined for each local model G^i as: $P^i: \Sigma^{i*} \cup \{\varepsilon\} \rightarrow \Sigma_o^{i*}$. Each state q_i of G is represented by an output vector h_j considered as a Boolean vector whose components are Boolean variables. Let d denote the number of state variables of G, the output vector h_j of each state q_j can be defined as: $\forall q_j \in Q, h(q_j) = h_j = (h_{j1}, ..., h_{jp}, ..., h_{jd}), h_{jp} \in \{0,1\}, 1 \le j \le 2^d, h_j \in Y \subseteq IB^d$. A transition from one state to another one is defined as a change of a state variable from 0 to 1, or from 1 to 0. Thus each transition produces an event α characterized by either rising, $\alpha = \uparrow h_{jp}$, or falling, $\alpha = \downarrow h_{jp}$, edges where $p \in \{1,2,..., d\}$.

To describe the effect of the occurrence of an event $\alpha \in \Sigma_o$, a displacement vector E_α is used. It is defined as a Boolean vector $E_\alpha = (e_{\alpha 1}, ..., e_{\alpha p}, ..., e_{\alpha d})$ in IB^d. If $e_{\alpha p} = 1$, then the value of p^{th} state variable h_{jp} will be set or reset when α occurs. While if $e_{\alpha p} = 0$, the value of p^{th} state variable h_{jp} will remain unchanged when α occurs. Consequently we can write:

$$\forall q_i, q_j \in Q, \forall \alpha \in \Sigma_o, q_j = \delta(\alpha, q_i) \Rightarrow h_j = h_i \oplus E_\alpha \qquad (1)$$

The symbol " $\oplus$ " denotes the logical operator Exclusive-OR.

Similarly, we can define the displacement vectors for the other observable events. The set of all the displacement vectors of all the events provides the displacement matrix E. For each event $\alpha \in \Sigma_0$, an enablement condition, $en_\alpha(q_i) \in \{0,1\}$, is defined in order to indicate if the event α can occur at the state q_i, $en_\alpha(q_i) = 1$, or not, $en_\alpha(q_i) = 0$. Consequently, (1) can be re-written as:

$$\forall q_i, q_j \in Q, \forall \alpha \in \Sigma_o, q_j = \delta(\alpha, q_i) \Rightarrow h_j = h_i \oplus (E_\alpha . en_\alpha(q_i)) \qquad (2)$$

The symbol "." denotes the logical operator AND.

2.2 Constrained-system Boolean model

Let $S = (\Sigma, Q_S, Y, \delta_S, h, q_0)$ denote the constrained-system model, characterized as Moore automaton. It defines the global desired behaviour of the system and it is represented by the prefixed closed specification language $K = L(S) \subseteq L(G)$. S can be obtained using different algorithms from the literature as the ones developed in (Phillipot *et al.* 2007, Ramadge and Wonham 1987) and the references therein. To obtain the transition function δ_S, the enablement conditions for all the system events at each state, $\forall \alpha \in \Sigma_0$, must satisfy all the specifications K, representing the desired behaviour:

$$\forall \alpha \in \Sigma_0, \forall q_i, q_j \in Q_S, q_j = \delta_S(\alpha, q_i) \Rightarrow \\ en_\alpha(q_i) = 1, h_j = h_i \oplus (E_\alpha . en_\alpha(q_i)) \qquad (3)$$

Thus, the constrained-system model contains only the authorized events at each state. Each local model G^i has a local constrained model S^i, which is a part of the global constrained model S. S^i is represented by the specification language $K^i = L(S^i)$, which is included in K. S^i is a Moore automaton: $S^i = (\Sigma^i, Q_S^i, Y^i, \delta_S^i, h^i, q_0^i)$ and $Q_S^i \subset Q^i$. All these notations have the usual definition but for the local constrained-system model S^i.

2.3 Events timing delays modelling

The majority of sensors and actuators in manufacturing systems produce correlated events since state's changes are usually effected by a predictable flow of materials (Pandalai and Holloway 2000). Therefore, a temporal model centered on the notion of expected event sequencing and timing relationships can be used.

In this paper, we define a set of expected consequents EC_β for each controllable event, $\beta \in \Sigma_c$, in order to predict uncontrollable but observable consequent events within pre-defined time periods. This EC_β is constructed for observable events and it describes the next events that should occur and the relative time periods in which they are expected. These pre-defined time periods are determined by experts or by learning according to the system dynamic and to the desired behaviour. If $u = \beta\alpha_1\alpha_2...\alpha_k$ is an observable event sequence starting by a controllable event β, and ending by the observable event sequence $\alpha_1\alpha_2...\alpha_k \subset \Sigma_o^*$, then the set of expected consequents $EC_\beta(u)$ is created when the event β occurs. $EC_\beta(u)$ has the following form : $EC_\beta(u) = \{C_{\alpha_1}^\beta, C_{\alpha_2}^\beta, ..., C_{\alpha_i}^\beta, ..., C_{\alpha_k}^\beta\}$. $C_{\alpha_i}^\beta$ is a consequent expected after the enablement of the controllable event β and it is defined as follows: $C_{\alpha_i}^\beta = \{\alpha_j, \alpha_i, (q_{\alpha_i}, [t_{\min}^{\alpha_i}, t_{\max}^{\alpha_i}], l_{q_{\alpha i}}^{\alpha_i})\}, j < i$. This expected consequent means that when α_j occurs, the event α_i should happen at the state q_{α_i} and within the time interval $[t_{min}^{\alpha_i}, t_{max}^{\alpha_i}]$. If it is the case then the expected consequent is satisfied. If the event α_i has occurred before $t_{min}^{\alpha_i}$ or after $t_{max}^{\alpha_i}$ then the expected consequent is not satisfied and it provides the fault labels $l_{q_{\alpha i}}^{\alpha_i}$, as the cause of this non satisfaction. The fault labels indicate one or more of fault candidates and they are defined by an expert. The set of expected consequent $EC_\beta(u)$ is evaluated by an expected function $EF_\beta(u)$. $EF_\beta(u)$ is equal to 1 if one of its expected consequents is not satisfied while it is equal to zero if all its expected consequents are satisfied. The set of expected consequents $EC_\beta(u)$ is deleted when they are satisfied, i.e., $EF_\beta(u) = 0$.

2.4 Fault detection and isolation checking

We adopt the hypothesis that each behavior which does not correspond to a normal one is considered as abnormal one. Thus, a fault can occur starting from any state of the desired behavior. This fault occurrence is unobservable and it leads the system to a faulty state. Each one of these faulty states must be reached within a finite delay for all the event sequences that can lead to this state starting from any other one of the desired behaviour states.

Let Ψ_{F_j} define the set of all the event sequences ending by a fault belonging to the fault partition Π_{F_j}. Thus $\Psi_F = \cup_{j=1}^r(\Psi_{F_j})$ denotes the set of all the event sequences ending by a fault belonging to one of fault partitions of Σ_Π. Consequently, $\Psi_F \subseteq (L - K)$, i.e., all the faulty sequences ending by a fault belonging to one fault partitions of Σ_Π are considered as violation of the specification language *K*. The set of faulty states is defined as $S_F : \cup_{j=1}^r(S_{F_j})$ where S_{F_j} is the set of states reached by the occurrence of a fault belonging to Π_{F_j}. Let H_{F_j} denote the set of all state output vectors of the faulty states belonging to S_{F_j}. Then, the output partition H_{F_j} is defined as : $\forall q' \in S_{F_j}, h' = h(q') \Rightarrow h' \in H_{F_j}$.

In order to ensure the fault detection, the following conditions must hold:

$$\forall \rho \in L, \rho \in K, \forall i \in \{1,2,...,n\}, \forall q \in Q \Rightarrow en_\rho^i(q) = 1 \quad (4)$$

$$\forall \rho \in L, \rho \in L - K, \forall j \in \{1,2,...,r\}, \rho \cap \psi_F \neq \Phi, \exists i \in \{1,2,...,n\}, \exists q \in Q \Rightarrow en_\rho^i(q) = 0 \text{ or } EF_q(P^i(\rho)) = 1 \quad (5)$$

$$|\rho| \leq k \in N \quad (6)$$

The condition (4) means that all the enablement conditions of all the local desired models must be satisfied for any event of a sequence belonging to the global desired behavior. Thus, this condition ensures that no conflict can occur between local desired models for the enablement of events at any state of the desired behaviour. The satisfaction of (5) ensures that any event sequence violating the global desired behavior, due to the occurrence of a fault, must be detected by reaching at least one state *q*. This detection is based on the non satisfaction either of the enablement condition of the latest event in the event sequence ρ or of its expected function. In the both cases, this non satisfaction can provide a set of fault labels F_j, $j \in \{1,2,..,r\}$. This later can contain one fault label, i.e., one fault candidate, or several fault candidates. In the latter case, a preference order can be defined among these fault candidates in order to help the human operators to isolate the original or real one. This preference order can be established using the human experience or by learning, i.e., simulation. Finally (6) guarantees that this detection will be realized in a finite delay or number of event transitions equal to the cardinality of the event sequence ρ.

3. MANUFACTURING SYSTEM EXAMPLE

To illustrate the proposed approach, we use the example of pick and place station of the flexible manufacturing system platform *cellflex* (http://meserp.free.fr/). This station realizes the import and the export of pieces by a gripper between two processes thanks to a pneumatic system of 3 axes (Fig. 1). The symbol ① refers to *Z* axis displacement, ② to *X* axis displacement, ③ to *Y* axis displacement and ④ to the pneumatic system gripper. This station is composed of 4 actuators piloted by 6 pre-actuators produced by different technologies. The information about the behavior of the station is provided by 9 sensors (Fig. 2).

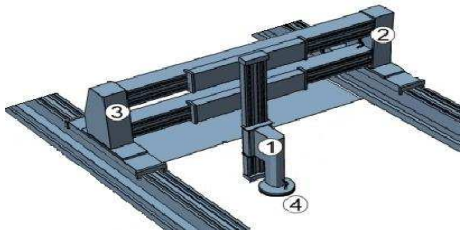

Fig. 1. Pick and place station

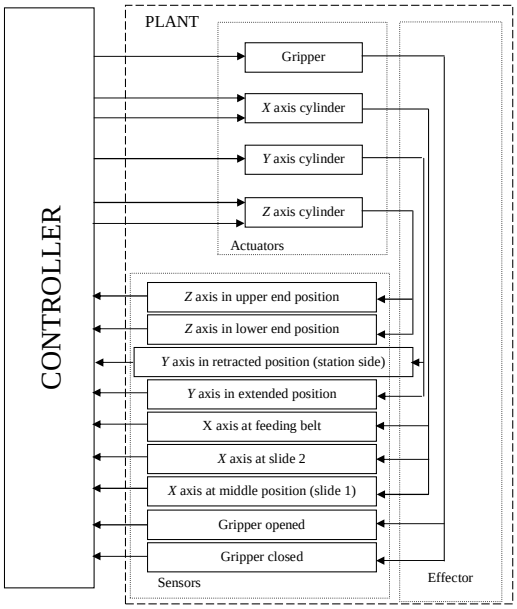

Fig. 2. Actuators and sensors of the pick and place station

3.1 *Y* axis Plant Elements models

We will illustrate the construction of *Y* axis model. Same reasoning can be followed for the construction of the other axis models. The *Y* axis actuator is a Double Acting Cylinder (DAC) where their positions are given by two sensors, retracted y_R and extended y_E positions ones (Fig. 3).

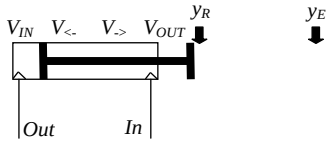

Fig. 3. Elements of *Y* axis

The Fig. 4 and Fig. 5 illustrate the free-fault models of plant elements of *Y* axis. The model G_{DAC} (Fig. 5) evolves from its initial state q_0 towards the states q_1/V_{IN} or q_3/V_{OUT} according, respectively, to the activation of the control signals *In* or *Out*. The states q_1/V_{IN} and q_3/V_{OUT} represent, receptively, the piston rod in home and in fully extended positions. If the model is located in the state q_1, the activation of the control signal *Out* leads the piston rod to move forward. This piston rod movement is represented by the dynamic state $q^*_2/V_{->}$. The output $V_{->}$ indicates that the piston rod is in movement towards its fully extended position. The time required to reach this position, T_s, is assigned to the time variable Δ. In the same time, a local clock *t* is initiated to calculate the spent time during the forward movement. At this dynamic state, two cases can arise. In the first case, the value of *t* becomes equal to the one allocated to Δ. This means that the actuator has reached its fully extended position. Therefore, G_{DAC} reaches the state q_3 with the output V_{out}. In the second case, the control signal *In* is activated. This activation forces the piston rod to stop moving forward in order to return to its home position. Thus, G_{DAC} evolves to the dynamic state q^*_4 with the output $V_{<-}$ indicating that the piston rod is in inversed movement. In this case, the present spent time *t* is assigned to Δ. Then, the local clock is initiated again to calculate the elapsed time in the inverse movement. When this time becomes equal to the one allocated to Δ, the piston reaches its home position indicated by the state q_1/V_{IN}. The same reasoning can be followed for the other states.

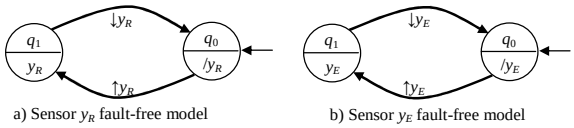

Fig. 4. Sensors y_R and y_E fault-free models

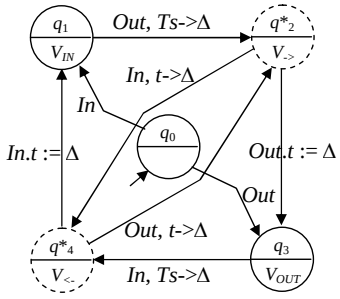

Fig. 5. DAC fault-free model

For each PE, we can enumerate, with the help of an expert, the possible potential faulty or degraded behaviors and their responsible candidates. Faulty behavior causes the production halt while the degraded one disturbs or reduces the optimal production performances. Table 1 shows the labels indicating the faulty candidates of the faulty and degraded behaviors of *Y* axis Plant Elements.

Table 1. Faulty and degraded behaviors and their responsible candidates for the *Y* axis Plant Elements

Type	Label	Description
Faulty behaviors	B_{yR}	sensor y_R blocked at 1
	$B_{/yR}$	sensor y_R blocked at 0
	B_{yE}	sensor y_E blocked at 1
	$B_{/yE}$	sensor y_E blocked at 0
	B_{Vin}	DAC blocked in retracted direction
	B_{Vout}	DAC blocked in extended direction
Degraded behaviors	$D_{V->}$	DAC too slowly acting in extended direction compared to its normal behavior
	$D_{V<-}$	DAC too slowly acting in retracted direction compared to its normal behavior

However, the faulty behaviors caused by these faults are not integrated in the models; only their labels are defined in order to propose fault candidates.

3.2 *Y* axis desired behavior model

The *Y* axis Plant Elements can be represented as a block for which the inputs are the control signals of the controller, *In* and *Out*, and the outputs are the sensors' information, y_R and y_E (Fig. 6). The controller is supposed to be safety and dependable. Consequently, it is not possible to have the activation of *In* and *Out* at the same time. When the control signal *Out* is activated, the normal response is $\downarrow y_R$ followed by $\uparrow y_E$.

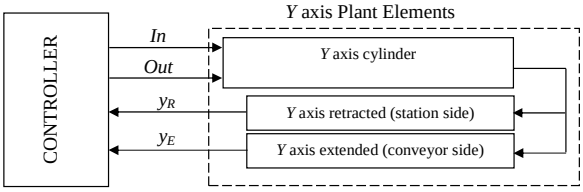

Fig. 6. Observable events of the "*Y* axis" PE

The local constrained-system model S^Y for the sub-model G^Y of the *Y* axis Plant Elements is depicted in Fig. 7. Since any double effect cylinder with 2 positions has the same behavior, the constrained model is obtained from a library given by an expert. In BDES modelling, this desired behavior can be described using two tables; the first one explains the enablement conditions for the occurrence of each event and the second one is the displacement matrix for the estimation of the state output vector of each next state. These tables are shown respectively in Table 2 and, Table 3 for the S^Y. As an example we can notice that the only event allowed to take place in the initial state of S^Y, characterized by the output vector h_1 = (y_R y_E *Out In*)=(1000), is the enablement of the controllable event $\uparrow$*Out* since its enablement condition, $en^Y_{q_1}(\uparrow Out)$, is satisfied and the enablement conditions for all the other events are false (See Table 2). The displacement vector of this event is $E^Y_{\uparrow Out} = (0010)$. The output vector of the next estimated state of S^Y is calculated using (2) : $h_2 = h_1 \oplus (E^Y_{\uparrow Out}.en^1_{\uparrow Out}(q_1)) = (1000) \oplus ((0010).1) = (1010)$.
Similarly, we calculate the next output vector according to the occurrence of each authorized observable event.

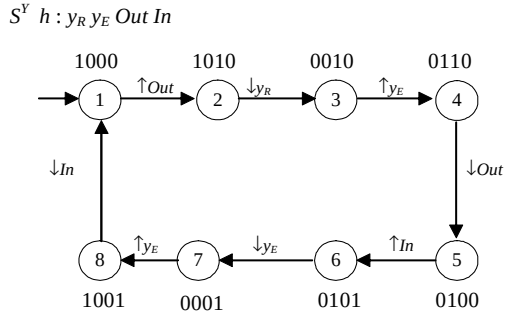

Fig. 7. Local constrained-system model S^Y for the sub model G^Y

Table 2. Enablement conditions for S^Y for the sub model G^Y

Event: σ S^Y	Enable condition: en^Y_σ
$\uparrow y_R$	$/y_R$. $/y_E$. /*Out* . *In*
$\downarrow y_R$	y_R . $/y_E$. *Out* . /*In*
$\uparrow y_E$	$/y_R$. $/y_E$. *Out* . /*In*
$\downarrow y_E$	$/y_R$. y_E . /*Out* . *In*
$\uparrow$*Out*	y_R . $/y_E$. /*Out* . /*In*
$\downarrow$*Out*	$/y_R$. y_E . *Out* . /*In*
$\uparrow$*In*	$/y_R$. y_E . /*Out* . /*In*
$\downarrow$*In*	y_R . $/y_E$. /*Out* . *In*

Table 3. The displacement matrix E^Y for S^Y

State variable	$\uparrow y_R$	$\downarrow y_R$	$\uparrow y_E$	$\downarrow y_E$	$\uparrow$*Out*	$\downarrow$*Out*	$\uparrow$*In*	$\downarrow$*In*
y_R	1	1	0	0	0	0	0	0
y_E	0	0	1	1	0	0	0	0
Out	0	0	0	0	1	1	0	0
In	0	0	0	0	0	0	1	1

3.3 Expected consequents definition

We use expected consequents to model the cylinder response times which can be obtained by learning and/or by technical documentation. For S^Y, we define 2 expected consequents, one for each command enablement: $EC_{\uparrow Out}$ and $EC_{\uparrow In}$. The enablement of *Out*, entails the events $\downarrow y_R$ and $\uparrow y_E$ to occur respectively at the states q_2 and q_3. $\downarrow y_R$ is expected to occur within the time interval [*t*1, *t*2] after the enablement of *Out*, $\uparrow y_E$ within the time interval [*t*3, *t*4] after the occurrence of $\downarrow y_R$ according to the system dynamic. If $\downarrow y_R$ does not occur at q_2 then the cylinder has not responded. Thus, the non satisfaction of the corresponding expected consequent at this state indicates the occurrence of the fault "DAC blocked in retracted direction" indicated by the label B_{Vin}. If $\downarrow y_R$ has occurred but too lately, then the provided fault is "DAC is acting too slowly in extended direction" indicated by the label $D_{V->}$. However when $\downarrow y_R$ occurs, S^Y will transit to the state q_3. If $\uparrow y_E$ has not occurred, then the non satisfaction of the corresponding expected consequent provides the fault candidate $\{y_E\}$ with the label $B_{/yE}$ to indicate that the sensor y_E is blocked at 1, stuck-off, since the piston has responded. Consequently $EC_{\uparrow Out}$ can be written as follows:

$$EC_{\uparrow Out} = \begin{Bmatrix} \{\uparrow Out, \downarrow y_R, (q_2, [t1, t2], \{B_{Vin}, D_{V->}\})\}, \\ \{\downarrow y_R, \uparrow y_E, (q_3, [t3, t4], \{B_{/yE}, D_{V->}\})\} \end{Bmatrix}.$$

The number of candidates can be reduced using a progressive monitoring. The occurrence of new sensors events can lead to eliminate the improbable or inconsistent candidates with this new observation. According to the case if the event $\downarrow y_R$ or $\uparrow y_E$ has occurred too lately or not, one faulty candidate will be validated. If the event $\downarrow y_R$, or $\uparrow y_E$, occurred then the faulty candidate is the DAC which is acting too slowly in extended direction compared to its normal behavior.

Similarly, the expected function for the enablement of the command *In* is written as follows:

$$EC_{\uparrow In} = \begin{Bmatrix} \{\uparrow In, \downarrow y_E, (q_6, [t1,t2], \{B_{Vout}, D_{V<-}\})\}, \\ \{\downarrow y_E, \uparrow y_R, (q_7, [t3,t4], \{B_{/yR}, D_{V<-}\})\} \end{Bmatrix}.$$

However, these abnormal behaviors require the determination of the intervals defining the acceptable time displacement of the DAC. To determine these intervals, we have established a learning phase about the system's normal behavior. The goal of this learning is to obtain realistic time response intervals related to the system dynamic and to the actuators technology. These intervals are obtained by a learning extrapolation of the probability, chance, of the occurrence of an event in this interval. For example, Fig. 7 presents the learning of time interval [$t3$,$t4$] of the occurrence of the event $\uparrow y_E$ after the occurrence of the event $\downarrow y_R$ in response to the activation of the command *Out*. Fig. 8 presents the learning extrapolation when the command *Out* is activated. This activation expects as normal response the events $\downarrow y_R$ and $\uparrow y_E$ within respectively the time intervals [$t1$,$t2$] and [$t3$,$t4$].

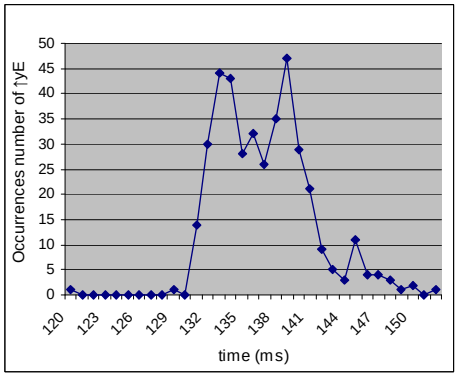

Fig. 7. Learning of the time interval of the occurrence of the event $\uparrow y_E$ after the occurrence of the event $\downarrow y_R$ in response to the activation of the command *Out*

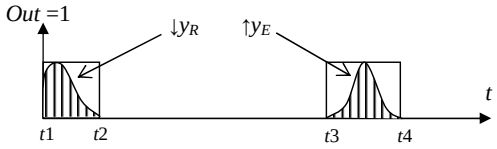

Fig. 8. Learning extrapolation for the time intervals of the sensors events occurrences in response to the activation of the command *Out*

3.4 Fault candidates generation for the *Y* axis

The candidates responsible of the occurrence of a fault in a PE can be determined based on its normal models as well as on its temporal constraints represented by a set of templates or chronicles. The following hypotheses are considered:

- Only one event responsible of a faulty or degraded behavior can occur at the same time,
- The controller is supposed to be dependable and safety. Consequently, the controller cannot be responsible of any fault as the one of sending two opposable control signals,
- The cylinder does not fail during operation, i.e., if it does fail, the fault occurs at the start of operation. This means that a fault cannot occur during the cylinder movement.

A fault is detected either when a non expected event occurs or when an expected event does not occur. In the first case the enablement condition of the event's occurrence is not satisfied. The possible faulty candidates are determined by identifying the state variables responsible of this non satisfaction. As an example, when the cylinder of the *Y axis* is in the initial state (Fig 7) and when the command *Out* is activated, the system transits to the next desired state characterized by the outputs *Out* = 1, *In* = 0, y_R = 1, y_E = 0. If the cylinder responds, then the sensor event $\downarrow y_R$ will be observed within the time interval [$t1$,$t2$] indicating that the cylinder motor is not faulty. If there is no sensor event within the time interval then we can infer that the DAC is blocked in retracted direction. However if $\downarrow y_R$ occurs but too late, then we can infer that the DAC is acting too slowly in extended direction compared to its normal behavior (the fault label $D_{V->}$).

The occurrence of $\uparrow Out$ transits the *Y* axis model to the second state q_2. The output vector for this state is calculated using (1) : $h_2 = (h_1 = 1000) \oplus (E^Y_{\uparrow Out} = 0010) = (1010)$. Since $en^Y_{\uparrow Out}(q_1) = 1$, see Table 2, then this state corresponds to a state of the desired behaviour S^Y. If the event $\uparrow y_E$ occurred at the state q_2 instead of the expected event $\downarrow y_R$, then $en^Y_{\uparrow y_E}(q_2)$ = $/y_R./y_E.Out./In$ = 0. The only reason of this non enablement, based on the conditions of q_2, is the variable state of the sensor y_R. Thus, the faulty candidate is the retracted sensor $\{y_R\}$.

4. CONCLUSION AND FUTURE WORKS

This paper presents a fault-free model-based approach for the Fault Detection and Isolation (FDI) of discrete manufacturing system. This approach considers the plant as a set of plant elements composed of actuators and their associated sensors. The goal is to take benefit of the composite structure of manufacturing systems. The use of fault-free models reduces the model construction complexity and avoids the necessity to define a priori the faults to be diagnosed.

A fault is detected either by the occurrence of non expected event or by the non occurrence of expected event within a predefined time intervals. The later indicate the actuators reactivity and are determined using a learning extrapolation. The occurrence of non expected event is detected when the enablement condition of this event is not satisfied. The state variables, representing the sensors outputs and the control signals, responsible of this non satisfaction are considered as the fault candidates. The non occurrence of expected event, or its occurrence too late, within its predefined time intervals

is detected using a template. The later is created for each control signal and it represents temporal constraints between events occurrences.

A future work of this paper is to define a codiagnosability notion allowing determining the set of faults which can be diagnosed and the time delay required for this diagnosis. This diagnosis is achieved by the set of local diagnosers. Each one of the later is responsible of a restricted area of the system or a specified component. In addition, we aim to use the learning of system dynamic as well as the expert knowledge to achieve an order preference between candidates when the set of fault candidates contains more than one candidate.

ACKNOWLEDGEMENTS

This work is integrated in the regional project MOSYP (Performances Measurements and Optimization of Production Systems). For this, the authors would like to thank the region Champagne-Ardennes within the project MOSYP (CPER ICOS).

REFERENCES

Balemi, S., Hoffmann, G.J., Gyugyi, P., Wong-Toi, H. and Franklin G.F. (1993). Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, vol. 38, N°7, pp.1040-1059.

Cordier, M.O. and Grastien, A. (2007). Exploiting Independence in a Decentralised and Incremental approach of diagnosis. *20th International Joint Conference on Artificial Intelligence*, pp. 292-297.

Darkhovski, B. and Staroswiecki, M. (2003). Theoretic Approach to Decision in FDI. *IEEE Transactions on Automatic Control*, vol. 48, n°5.

Devillez A., Sayed Mouchaweh M. and Billaudel P. (2004), A process monitoring module based on fuzzy logic and Pattern Recognition, *International Journal of Approximate Reasoning*, 37/1, 43-70.

Hadjicostis, C.N. (2005). Probabilistic Fault Detection in Finite-State Machines Based on State Occupancy Measurements. *IEEE Transactions on Automatic Control*, vol. 50, N°12, pp. 2078-2083.

Lamperti, G., Zanella, M. (2008). On Processing Temporal Observations in Monitoring of Discrete-Event Systems. *Book Chapter, Enterprise Information Systems: Part 3*, vol. 3, pp.135-146, ISBN 978-3-540-77580-5.

Isermann, R. (1997). Supervision, Fault Detection and Fault-Diagnosis Methods-an Introduction. *Control Engineering Practice*, Vol. 5, N° 5, pp.639-652.

Patton, R., Clark R.R. and Frank M. (2000). Issues of Fault Diagnosis for Dynamic Systems. *Contributor Ron Patton, Edition: illustrated. Published by Springer.*

Philippot, A., Sayed-Mouchaweh, M. and Carré-Ménétrier, V. (2007). Unconditional Decentralized Structure for the fault diagnosis of Discrete Event Systems. *1st IFAC Workshop on Dependable Control of Discrete-event Systems (DCDS'07)*, Cachan, France.

Qiu, W. (2005). Decentralized/distributed failure diagnosis and supervisory control of discrete event systems. *PhD of the Iowa State University*, USA.

Rozé, L. and Cordier, M.O. (2002). Diagnosing Discrete-Event Systems: Extending the "Diagnoser Approach" to Deal with Telecommunication Networks. *In Discrete Event Dynamic Systems*, vol. 12, n°1, pp.43-81, ISSN 0924-6703.

Sampath, M. (1995). A Discrete Event Systems Approach to Failure Diagnosis. *Thesis, University of Michigan*, Michigan, USA.

Wang, Y. (2000). Supervisory Control of Boolean Discrete-Event Systems. *Thesis of Master of Applied Sciences, University of Toronto*, Canada.

Wang, Y., Yoo, T.S. and Lafortune S. (2005). Decentralized diagnosis of discrete event systems using conditional and unconditional decisions. *Proceeding of CDC'05, 44th IEEE Conference on Decision and Control.*

Wonham, W. M. and Ramadge, P.J. (1987). On the supremal controllable sublanguage of a given language. *SIAM Journal on Control and Optimization*, vol. 25, n°3, pp.637-659.