Decentralized Fault Detection and Isolation of Manufacturing Systems

M. Sayed Mouchaweh

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@univ-reims.fr

ABSTRACT

This paper presents a Boolean discrete event model based approach for Fault Detection and Isolation (FDI) of manufacturing systems. This approach considers a system as a set of independent 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 a 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 sensor readings and control signals. The proposed approach is applied to a flexible manufacturing system.

1 INTRODUCTION

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 or the most likely to have occurred. Each fault that can result in a certain symptom is considered as a possible fault candidate.

The principal advantage of FDI approaches using both normal and fault behaviors, is the precision of the fault isolation. However, integrating the system behavior in response to a predefined set of faults increases significantly the model size. In addition, only predefined faults can be diagnosed. These disadvantages can be avoided using a fault free model (Pucel, *et al., 2009*; Roth, *et al., 2009*). However, the fault isolation cannot be as precise as the one using normal and fault behaviors.

Performing the diagnosis of a large scale Discrete Event System (DES) by using a global model is unrealistic. In addition, this type of systems is naturally decentralized, i.e. they are composed of several subsystems or components possessing their own local information. Decentralized approaches (Debouk, *et al., 2000*; Garcia and Yoo, 2005; Qiu and Kumar, 2006; Wang, *et al., 2007*) are an alternative to achieve the diagnosis of systems of this type. In these approaches, the diagnosis is performed based on a set of local diagnosers. Each local diagnoser is responsible for a restricted area of the system. Since no communication is allowed among the local diagnosers, a global model of the system is required to take into account the links between the interrelated components.

In this paper, we propose a decentralized fault free model based approach to diagnose plant faults of DESs. The independence property between system components is exploited in order to describe the global model by the fault free models of its components. Each component is composed of an actuator and its associated sensors. These models are represented as Boolean DES models. A behavior which does not correspond to a normal one is considered as a fault behavior. Component elements (actuator/sensor), responsible for this fault behavior, are considered as fault candidates.

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

2 PROPOSED APPROACH

2.1 Boolean models of system components

We use Boolean DES (BDES) modeling (Aveyard, 1973) to model the equipment (sensors/actuators) behavior of a system. The system model G consists of n local models: $G^1,\ldots, G^n$; each one owns its local observable events responsible for a particular component c^i, $i \in \{1,...,n\}$. The model $G = (\Sigma, Q, Y, \delta, h, q_0)$ is represented as a Moore automaton and $L = L(G)$ denotes its corresponding

prefixed closed language. Σ is a finite set of events and it includes the observable and unobservable events. Q is the set of states, Y is the output space, $\delta: \Sigma^* \text{ x } Q \rightarrow Q$ is the partial transition function, and Σ^* is the set of all event sequences of the language $L(G)$. The partial transition function $\delta(\sigma, q)$ provides the next state 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 kinds of faults in an equipment element (sensor/actuator).

Controllable events $\Sigma_c \subseteq \Sigma$ are defined as controller outputs sent to actuators and uncontrollable events $\Sigma_u \subseteq \Sigma$ as controller inputs coming from sensors. $(\Sigma_o = \Sigma_c \cup \Sigma_u) \subset \Sigma$ is the set of observable events. Typically, observable events in a system are either enabled/disabled commands or changes of sensor readings. 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 a Moore automaton. $\Sigma_o^i = \Sigma_c^i \cup \Sigma_u^i$ 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 a global projection function or mask, $P_L: \Sigma^* \cup \{\varepsilon\} \rightarrow \Sigma_o^*$, where ε denotes the empty sequence. Similarly, a local projection function can be defined for each local model G^i as: $P^i: \Sigma^{i*} \cup \{\varepsilon\} \rightarrow \Sigma_o^{i*}$. A state q_j of G is represented by an output vector h_j considered as a Boolean vector whose components, $(h_{j1}, ..., h_{jp}, ..., h_{jd}), h_{jp} \in \{0,1\}$, are Boolean variables. 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 $p \in \{1,2,..., d\}$.

In order 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 $\{0,1\}^d$. If $e_{\alpha p} = 1$, then the value of p^{th} state variable h_{jp} will be complemented when α occurs. While if $e_{\alpha p} = 0$, the value of p^{th} state variable h_{jp} will remain unchanged when α occurs. Thus, the output vector can be calculated by:

$$q_j = \delta(\alpha, q_i) \Rightarrow h_j = h_i \oplus E_\alpha \quad (1)$$

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

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

$$q_j = \delta(\alpha, q_i) \Rightarrow h_j = h_i \oplus (E_\alpha . en_\alpha(q_i)) \quad (2)$$

The symbol "." denotes the logical operator AND.

2.2 Constrained Boolean models of system components

Let $S = (\Sigma, Q_S, Y, \delta_S, h, q_0)$ denote the constrained system model, characterized as a 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 (Philippot, *et al., 2007*; Wonham and Ramadge, 1987) and the references therein. To obtain the transition function δ_S, the enablement conditions for all system events, $\forall \alpha \in \Sigma_o$, at each state must satisfy all the specifications K, representing the desired behavior. Thus, the constrained system model contains only the authorized events at each state. When the enablement condition of an event is not satisfied, this indicates the occurrence of a fault.

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 Modelling timing delays of events

In this paper, we define a set of expected consequences EC_β for each controllable event $\beta \in \Sigma_c$, in order to predict uncontrollable but observable consequent events within predefined time intervals. EC_β is constructed for controllable events and it describes the next events that should occur and the relative time intervals in which they are expected. The predefined time intervals are determined by experts or by learning according to the system dynamics and to the desired

behavior. Let βu be an observable event sequence starting by controllable event β, and ending by observable but uncontrollable event sequence $u = \alpha_1\alpha_2...\alpha_k \in \Sigma_o^*$. Then, the set of expected consequences $EC_\beta(u)$ is created when β occurs. $\alpha_1\alpha_2...\alpha_k$ is the longest uncontrollable but observable event sequence in response to β. $EC_\beta(u)$ has the form: $\left\{C_\beta^{\alpha_1}, \overline{C}_\beta^{\alpha_1}, C_\beta^{\alpha_2}, \overline{C}_\beta^{\alpha_2}, ..., C_\beta^{\alpha_i}, \overline{C}_\beta^{\alpha_i}, ..., C_\beta^{\alpha_k}, \overline{C}_\beta^{\alpha_k}\right\}$. $C_\beta^{\alpha_i}$ is a positive consequence which should be satisfied in the case of a normal behavior. $\overline{C}_\beta^{\alpha_i}$ is a negative consequence which should not be satisfied in a normal behaviour. They are defined as follows: $C_\beta^{\alpha_i} = \left\{\alpha_i, (q_{\alpha_i}, t_{\min}^{\alpha_i} \le t_{\alpha_i} \le t_{\max}^{\alpha_i}, l_\beta^{\alpha_i})\right\}, \overline{C}_\beta^{\alpha_i} = \{\alpha_i, (q_{\alpha_i}, t_{\max}^{\alpha_i} < t_{\alpha_i} \le t_{\max}^{\alpha_i} + \Delta^{\alpha_i}, \overline{l}_\beta^{\alpha_i})\}$. Δ^{α_i} is the maximal time period within which event α_i is expected to occur. The positive consequence means that event α_i should happen at state q_{α_i} and within the time interval [$t_{min}^{\alpha_i}$, $t_{max}^{\alpha_i}$]. If it is the case, then the positive expected consequence is satisfied. Otherwise, the positive expected consequence is not satisfied and it provides the set of fault labels $l_\beta^{\alpha_i}$ as the cause of this non satisfaction. The negative expected consequence means that if event α_i occurred after $t_{max}^{\alpha_i}$, then this satisfaction indicates a fault behavior and it provides the set of fault labels $\overline{l}_\beta^{\alpha_i}$ as the cause of this satisfaction. Positive consequences are used to infer the fault candidates in the case of non occurrence of an event. While negative consequences are used to infer the fault candidates in the case of too late event occurrences. The late occurrence of an event characterizes a degraded behavior of a system. Fault behavior causes the production halt while a degraded one reduces the optimal production performance.

Each expected positive/negative consequence $\left\{C_\beta^{\alpha_i}, \overline{C}_\beta^{\alpha_i}\right\}$ is evaluated by an expected function $EF_\beta(\alpha_i)$. $EF_\beta(\alpha_i)$ is equal to 1 if the positive consequence $C_\beta^{\alpha_i}$ related to event α_i is not satisfied or its associated negative consequence $\overline{C}_\beta^{\alpha_i}$ is satisfied. While it is equal to zero if the positive expected consequence is satisfied. If a positive consequence is satisfied, then its associated negative one is not satisfied.

2.4 Fault detection and isolation

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

Let Ψ_{F_j} define the set of all event sequences ending by a fault belonging to fault partition Π_{F_j}. Thus, $\Psi_F = \bigcup_{j=1}^{r}(\Psi_{F_j})$ denotes the set of all event sequences ending by a fault belonging to a fault partition of Σ_Π. Consequently, $\Psi_F \subseteq (L - K)$, i.e. all the fault sequences ending by a fault of Σ_Π are considered as violation of the specification language *K*.

The FDI of a global model *G* is defined as follows. Let $\beta\rho\theta$ be an event sequence starting by controllable event β. ρ is an event sequence that ends by a failure event and θ is a continuation of ρ. In order to ensure the fault detection, the following condition must hold:

$$\left.\begin{array}{l}(\forall \beta\rho\theta \in L : \rho \in \psi_F)(P(\theta) = \alpha_1...\alpha_k, \\ \|\theta\| = k \in IN)(\exists q \in Q) \\ EF_\beta(\alpha_k) = 1 \text{ or } en_{\alpha_k}(q) = 0\end{array}\right] \Rightarrow \quad (3)$$

The satisfaction of (3) ensures that any event sequence violating the global desired behavior, due to the occurrence of a fault, must be detected by:

-) the non satisfaction of the positive expected consequence related to event α_k,

-) the satisfaction of the negative expected consequence related to event α_k,

-) the non satisfaction of the enablement condition of the latest observable but uncontrollable event α_k in the event sequence.

This detection is performed in a finite time delay; specifically at *k* event transitions after ρ.

In this paper, we assume that at most one fault may occur at a time. The set of fault candidates can be determined as follows. Let h_{jp} denote the state variable describing the discrete status (on/off) of sensor *p*. Let $\left\{\uparrow h_{jp}, \downarrow h_{jp}\right\}$ be the events produced by this state variable at state q_j. The non occurrence of $\downarrow h_{jp}$ in the expected normal time interval generates the following fault candidates as an explanation: l_β^α = {"sensor *p* blocked at 1", "actuator associated with sensor *p* stuck-off", "actuator associated with sensor *p* acting too slowly according to its normal behavior"}. The too late occurrence of $\downarrow h_{jp}$ generates the fault candidate: $\overline{l}_\beta^\alpha$ = {"actuator associated with sensor *p* acting too slowly according to its normal behavior"}. If $en_\alpha(q_j) = 0$

because h_{jp} = 1, then the fault candidate $l^{\alpha}_{q_j}$ is {"sensor p blocked at 1"}. In the contrary case, i.e. h_{jp} = 0, the fault candidate $l^{\alpha}_{q_j}$ is {"sensor p blocked at 0"}.

2.5 Progressive monitoring

Progressive monitoring aims at reducing the number of fault candidates thanks to the occurrence of new events. The fault candidates which are no more consistent with the occurrence of a new event will be deleted of the set of fault candidates. In addition, only the common fault candidates explaining together the non satisfaction of positive consequences and enablement conditions or the satisfaction of negative consequences are kept. This is justified since one fault may occur at a time. Let $FCAN_{\beta}$ be the set of fault candidates in the case of the occurrence of controllable event β. If $en_{\alpha_i}(q_j)=0$, then $FCAN_{\beta}=l^{\alpha_i}_{q_j}$. If $\alpha_i=\uparrow h_{jp}$ (respectively $\alpha_i=\downarrow h_{jp}$) and its enablement condition is satisfied: $en_{\alpha_i}(q_j)=1$, then sensor p is not blocked at 0 (respectively sensor p is not blocked at 1). Thus, the fault candidate: "sensor p blocked at 0" (respectively "sensor p blocked at 1") will be removed from the fault candidates. The same reasoning is applied for the satisfaction of the positive expected consequence or the non satisfaction of the negative expected consequence related to the occurrence of an event α_i. Therefore, the set of fault candidates is reduced by both eliminating the fault candidates which are no more consistent with the occurrence of an event and by keeping the candidates explaining together the non satisfaction of positive consequences and enablement conditions and the satisfaction of negative consequences. Let $NFCAN_{\beta}$ be the set of fault candidates to be removed from $FCAN_{\beta}$. The set of fault candidates is reduced as it is depicted in Figure 1.

2.6 Decentralized fault detection and isolation

The system considered in this paper is composed of a set of independent components: c^i, $i\in\{1,...,n\}$. Two components c^i and c^j, respectively, their local models G^i and G^j, are considered as independent if they verify the following two proprieties: the state-independent and the transition-independent ones. The first property (Cordier and Grastien, 2007) states that if G^i and G^j are a decomposition of their global model $G^{ij}=G^i \,||\, G^j$, then each one of them has a unique initial state. Thus, the global model can be retrieved by their synchronous composition. The transition-independence property is satisfied between two local models G^i and G^j if their automata do not have any common synchronisation event.

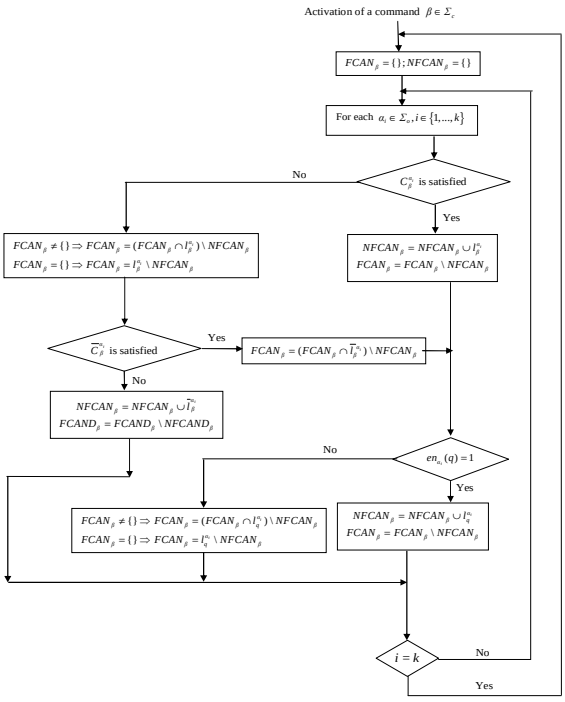

Figure 1: Progressive monitoring flowchart. "$\cup$" is the union set operation, "$\cap$" is the intersection set operation, and "\" is the difference set operation

We adapt the notion of decentralized diagnosis (Sengupta and Tripakis, 2002), defined for models containing both normal and fault behaviors, for the case of normal behavior models. A system is decentrally FDI iff each fault occurrence can be detected and its associated set of responsible candidates can be generated based on a set of local models and a set of inter-local models message events. In order to ensure the decentralized FDI, the following conditions must hold:

$$(\forall\beta\theta\in L)(\forall\theta\in K:P^i(\theta)=\alpha_1...\alpha_k)\\(\forall i\in\{1,...,n\})(\forall q\in Q)\Rightarrow en^i_{\alpha_k}(q)=1 \quad (4)$$

$$\left.\begin{array}{l}(\forall\beta\rho\theta\in L)(\rho\in L-K)(P^i(\theta)=\alpha_1...\alpha_k)\\(\exists i\in\{1,2,...,n\})(\exists q\in Q)\end{array}\right]\Rightarrow\\ en^i_{\alpha_k}(q)=0 \text{ or } EF_{\beta}(\alpha_k)=1 \quad (5)$$

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

behavior. 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. At this state, the detection is based on the non satisfaction of the enablement condition of the latest event α_k in the event sequence, of its positive expected consequence or on the satisfaction of its negative expected consequence.

If the system is composed of independent components, there is no need for inter-models messages to ensure a decentralized FDI equivalent to the centralized FDI.

3 PICK AND PLACE STATION EXAMPLE

To illustrate the proposed approach, we use the example of Figure 2 which presents a flexible manufacturing system platform called *cellflex* (http://meserp.free.fr/).

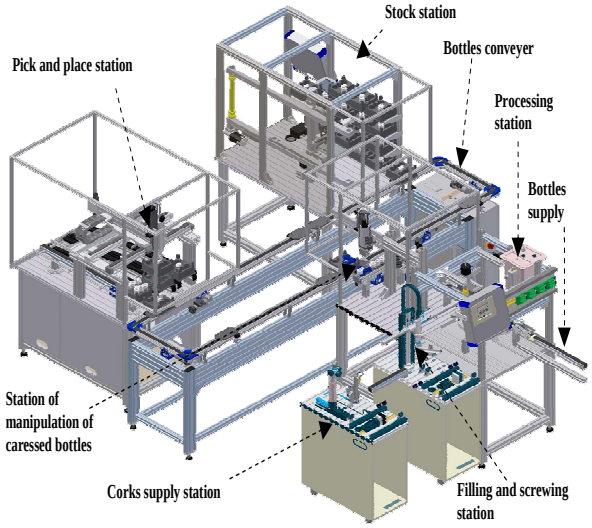

Figure 2: Flexible manufacturing system

We focus on the pick and place station; the other stations can be treated by the same reasoning. Pick and place station performs import and export of pieces by a gripper using a pneumatic system of 3 axes. This station is composed of 4 actuators piloted by 6 pre-actuators. The information about the behavior of the station is provided by 9 sensors (Figure 3). For each plant element (sensor/actuator), we can enumerate, with the help of an expert, the potential fault or degraded behaviors and their responsible candidates. We illustrate the application of the proposed approach using the *Y* axis. The same reasoning can be followed for the construction of the other axis. The *Y* axis actuator is a Double Acting Cylinder (DAC) where positions are given by two sensors, retracted y_R and extended y_E positions. Table 1 shows the fault candidates for the *Y* axis.

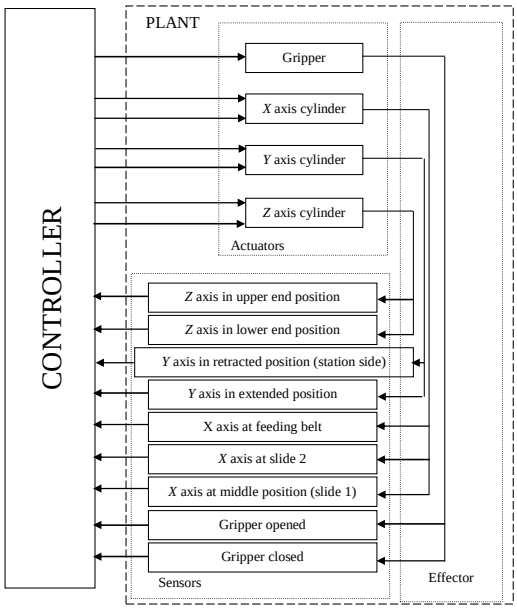

Figure 3: Actuators and sensors of pick and place station

Table 1: Fault candidates for the *Y* axis plant elements

Type	Label	Description
Fault 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 acting too slowly in extended direction compared to normal behavior
	$D_{V<-}$	DAC acting too slowly in retracted direction compared to normal behavior

3.1 Fault free models of the Y axis plant elements

The *Y* axis actuator is a Double Acting Cylinder (DAC) where retracted and extended positions are indicated respectively by two sensors y_R and y_E (Figure 4).

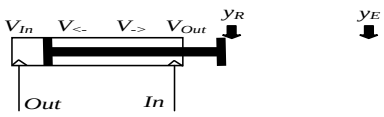

Figure 4: Elements of the *Y* axis

Figure 5 and Figure 6 illustrate the fault free models of the *Y* axis plant elements. The model G_{DAC} (Figure 6) evolves from its initial state q_0/V_{In} after the occurrence of $\uparrow Out$. State q_0/V_{In} indicates that the piston rod is in home position. The occurrence of $\uparrow Out$ leads the piston rod to move forward. This piston rod movement is

represented by dynamic state $q^*_1/V_{->}$. The output $V_{->}$ indicates that the piston rod is in movement towards its fully extended position. The time T_s required to reach this position 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, when 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 state q_2 with the output V_{Out}. The deactivation of *Out* leads the system to be in state q_3 with the output V_{Out}. At this state, control signal $\uparrow In$ can occur. This activation leads the piston rod to return to its home position. Thus, G_{DAC} evolves to dynamic state q^*_4 with the output $V_{<-}$ indicating that the piston rod is in inverse movement. The time T_s 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 arrives to its home position indicated by reaching state q_5/V_{In}. The deactivation of *In* transits the system to its initial state.

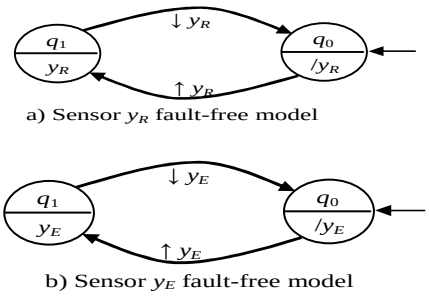

Figure 5: Fault free models of sensors y_R and y_E

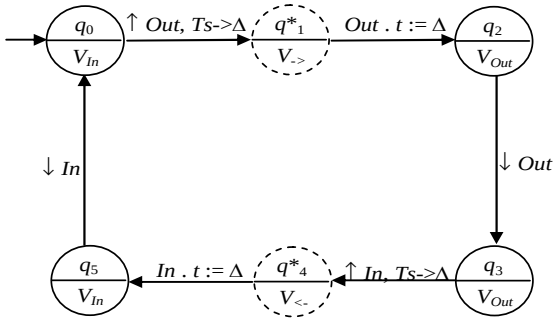

Figure 6: DAC fault free model

3.2 Desired behavior model of the Y axis

The *Y* axis plant is represented as a block; its inputs are control signals *In* and *Out*, and its outputs are sensor readings y_R and y_E (Figure 7). When $\uparrow Out$ occurs, the normal response is $\downarrow y_R$ followed by $\uparrow y_E$.

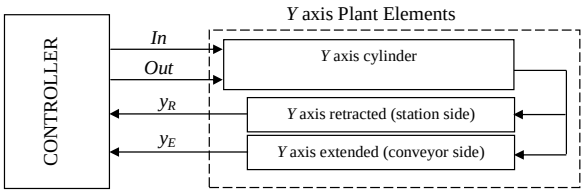

Figure 7: Observable events of theY axis plant elements

Local constrained system model S^Y for submodel G^Y of the *Y* axis is depicted in Figure 8. Since any DAC with 2 positions has always the same desired behavior, the constrained model can be obtained from a library. In the case of DAC with *n* positions, several constrained models can be obtained according to the global desired behavior. In (Philippot, *et al.*, *2007*) an algorithm is defined to extract the local desired behavior based on the global one.

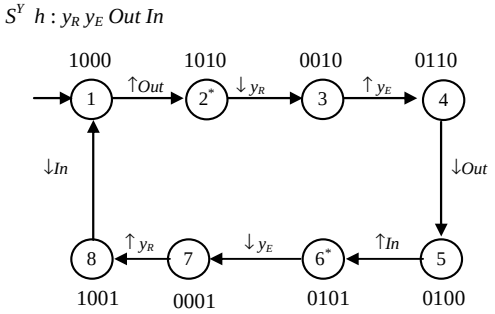

Figure 8: Local constrained-system model S^Y for submodel G^Y

In BDES modelling, the 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 S^Y.

Table 2: Enablement conditions for S^Y

Event: σ of 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: Displacement matrix E^Y for S^Y

	$\uparrow y_R$	$\downarrow y_R$	$\uparrow y_E$	$\downarrow y_E$	$\uparrow Out$	$\downarrow Out$	$\uparrow In$	$\downarrow In$
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 Definition of expected consequences

We use expected consequences to model cylinder response times. For S^Y, we define 2 expected consequences: $EC_{\uparrow Out}$ and $EC_{\uparrow In}$, one for each command enablement: $\uparrow Out$ and $\uparrow In$. The enablement of *Out* entails events $\downarrow y_R$ and $\uparrow y_E$ to occur respectively at states q_2 and q_3. After the occurrence of $\uparrow Out$, $\downarrow y_R$ is expected to occur within the time interval [$t1$, $t2$] and $\uparrow y_E$ within the time interval [$t3$, $t4$]. These time intervals depend on system dynamics. We define the maximum time $\Delta^{\downarrow y_R}_{\max}$ accepted for the cylinder response, in the case of degraded behavior, entailing the occurrence of event $\downarrow y_R$. If $\downarrow y_R$ does not occur at q_2 within [$t1$, $t2$], then either: -) the cylinder has not responded, -) the cylinder is acting too slowly, or -) sensor y_R is blocked at 1. Thus, the non satisfaction of the corresponding positive expected consequence at this state provides three fault candidates: "DAC blocked in retracted direction" indicated by the label B_{Vin}, "DAC acting too slowly in extended direction" indicated by the label $D_{V->}$, and "sensor y_R blocked at 1" indicated by the label B_{yR}. If $\downarrow y_R$ occurred but too lately, then the provided fault is "DAC acting too slowly in extended direction" indicated by the label $D_{V->}$. The same reasoning can be followed for event $\uparrow y_E$. Consequently $EC_{\uparrow Out}$ can be written as follows:

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

Similarly, the expected consequences for the enablement of command *In* is defined.

To determine the acceptable time of displacement of the DAC in the case of normal and degraded behaviors, we have established a learning phase about the system normal behavior. The goal of this learning is to obtain realistic time response intervals related to the system dynamics and to the actuators technology. These intervals are obtained by a learning extrapolation of the probability of the occurrence of an event in this interval. Figure 9 presents the learning extrapolation after the occurrence of $\uparrow Out$.

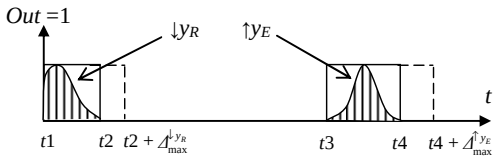

Figure 9: Learning extrapolation for time intervals of sensor event occurrences in response to ↑Out

3.4 Generation of fault candidates for the Y axis

The candidates responsible for the occurrence of a fault in a plant element (sensor/actuator) can be determined based on its normal models as well as on its temporal constraints represented by a set of positive/negative expected consequences. The following hypotheses are considered:

-) all components fail independently with equal likelihood,
-) one fault may occur at a time,
-) the controller is supposed to be dependable and safe,
-) 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.

The fault candidates are generated as follows. When the *Y axis* cylinder is in the initial state (Fig 4) and when $\uparrow Out$ occurs, the system transits to the next desired state characterized by *Out* = 1, *In* = 0, y_R = 1, y_E = 0. This state output vector is calculated using (1): $h_2 = (h_1 = 1000) \oplus (E^Y_{\uparrow Out} = 0010) = (1010)$. If the cylinder responds, then sensor event $\downarrow y_R$ will be observed within [$t1$, $t2$] indicating that the cylinder motor is not faulty. 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 event $\uparrow y_E$ occurred at the state q_2 instead of expected event $\downarrow y_R$, then $en^Y_{\uparrow y_E}(q_2)$ = $/y_R./y_E.Out./In$ = 0. The only reason for this non enablement, based on the conditions of q_2, is the state variable of sensor y_R. Thus, the fault candidate is $FCAN_{\uparrow Out} = \{B_{yR}\}$. If there is no sensor event within [$t1$, $t2$], then positive expected consequence $C^{\downarrow yR}_{\uparrow Out}$ is not satisfied and the fault candidates are generated as follows: $FCAN_{\uparrow Out} = I^{\downarrow yR}_{\uparrow Out} = \{B_{Vin}, D_{V->}, B_{yR}\}$. The same reasoning can be followed for the other cases of generation of fault candidates.

3.5 Progressive monitoring

The number of candidates can be reduced using a progressive monitoring (Fig 1). The occurrence of new sensor events can lead to eliminate the inconsistent candidates with this new observation. As an example, if $\downarrow y_R$ is observed within the time interval [$t1$, $t2$], then positive expected consequence $C^{\downarrow yR}_{\uparrow Out}$ as well as the enablement condition of $\downarrow y_R$ are satisfied. Thus, we can obtain: $NFCAN_{\uparrow Out} = \{B_{y_R}, B_{V_{in}}, D_{V->}\}$ and $FCAN_{\uparrow Out} = \{\}$. If $\uparrow y_E$ did not occur, then the non satisfaction of $C^{\uparrow yE}_{\uparrow Out}$ generates the fault candidates: $FCAN_{\uparrow Out} = (l^{\uparrow yE}_{\uparrow Out} = \{B_{y_E}, B_{V_{in}}, D_{V->}\}) / \{B_{yR}, B_{V_{in}}, D_{V->}\} = \{B_{yE}\}$. If events $\downarrow y_R$ and $\uparrow y_E$ both did not occur, then the fault candidates are generated and then are reduced as it is depicted in Figure 10.

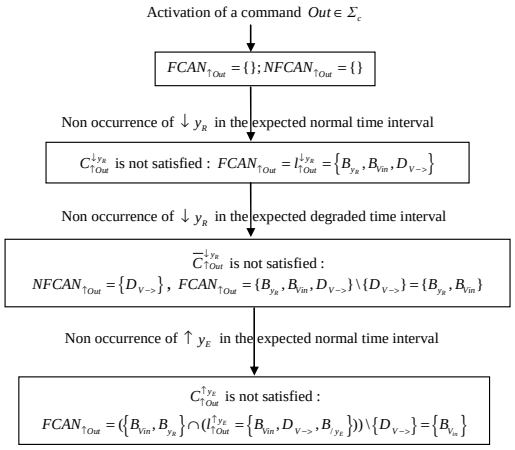

Figure 10: Progressive monitoring in the case of non occurrence of both $\downarrow y_R$ and $\uparrow y_E$

4 CONCLUSION

This paper presents a fault free model based approach for the Fault Detection and Isolation of discrete manufacturing system. The use of fault free models reduces the model construction complexity since there is no need to integrate fault behaviours in the system models.

In this paper, the components are assumed to be independent. Thus, a future work is to develop the proposed approach for the case of a system of interrelated components.

ACKNOWLEDGMENT

This work is supported by the Scientific Interest Group surveillance, safety and security of the big systems (GIS3SGS).

REFERENCES

(Aveyard, 1973) R. L. Aveyard. A boolean mode1 for a class of discrete event systems. *IEEE Trans. Systems, Man, and Cybernetics*, vol. SMC4-3, pp. 249-258, 1973.

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

(Debouk *et al., 2000*) R. Debouk, S. Lafortune and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems, *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 10 (1-2), pp. 33–86, 2000.

(Garcia and Yoo, 2005) H. E. Garcia and T. S. Yoo. Model-based detection of routing events in discrete flow networks, *Automatica*, Volume 41 (4), pp. 583-594, 2005.

(Philippot *et al., 2007*) A. Philippot, M. Sayed-Mouchaweh and V. Carré-Ménétrier. Unconditional Decentralized Structure for the fault diagnosis of Discrete Event Systems, in *Proceeding of 1st IFAC Workshop on Dependable Control of Discrete-event Systems*, 2007.

(Pucel and Stumptner, 2009) X. Pucel, W. Mayer and M. Stumptner. Diagnosability analysis without fault models, 20th International Workshop on Principles of Diagnosis (DX'09). Stockholm, Sweden, pp. 67-74, 2009.

(Qiu and Kumar, 2006) W. Qiu and R. Kumar. Decentralized failure diagnosis of discrete event systems, *IEEE Transactions on systems, man and cybernetics*-Part A, Vol. 36 (2), pp. 628-643, 2006.

(Roth *et al., 2009*) M. Roth, J.-J. Lesage, L. Litz, A residual inspired approach for fault localization in DES, in *Proceedings of the 2nd IFAC Workshop on Dependable Control of Discrete Event Systems* (DCDS'09), pp. 347-352, 2009

(Sengupta and Tripakis, 2002) R. Sengupta and S. Tripakis. Decentralized diagnosability of regular languages is undecidable, 40th IEEE Conf. on Decision and Control, pp. 423-428, 2002.

(Wang and Lafortune, 2007) Y. Wang, T. S. Yoo and S. Lafortune. Diagnosis of discrete event systems using decentralized architectures, *Discrete Event Dynamic Systems*, Vol. 17 (2), pp. 233-263, 2007.

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