Adaptive online fault diagnosis of manufacturing systems based on DEVS formalism.

Conference Paper · July 2017

CITATIONS
0

READS
17

2 authors, including:

Lala Rajaoarisoa
Institut Mines-Télécom
35 PUBLICATIONS **67** CITATIONS

SEE PROFILE

Adaptive online fault diagnosis of manufacturing systems based on DEVS formalism

L. Rajaoarisoa * **M. Sayed-Mouchaweh** *

* *Computer Systems and Automation, Mines Douai - IA, 59 500 Douai, France, (e-mail: lala.rajaoarisoa@mines-douai.fr).*

Abstract: In this paper, an adaptive fault diagnosis approach is proposed in order to perform the fault diagnosis of manufacturing systems. The desired (normal) behavior is represented by a set of temporal specifications while faults are considered to be the execution of specific fault behavior violating one or more of these specifications. The inference of the fault type of each fault is achieved by a diagnosis module called diagnoser. Diagnoser's model is based on DEVS formalism and two conditions is given to verify if the diagnoser covers fully and optimally the observable events. The approach considers that only normal behavior is known initially and therefore the diagnoser will be generate firstly with just this normal behavior. Since, it is hard to include in advance all abnormal behaviors, then the approach adapts the diagnoser in order to integrate new specific fault behaviors iteratively into its inference engine. This adaptation allows increasing the diagnosis capacity, called diagnosability, over time. Real-time tests are performed on an automated system available into our lab to allow an online validation.

Keywords: Online diagnosis, Real-time diagnoser, Discrete event systems, Fault detection and isolation, DEVS formalism.

1. INTRODUCTION

The production automation systems relies on different techniques and advanced technologies. Thus, maintenance of these systems can become complex, hard and expensive and therefore the implementation of a suitable diagnoser is arise. For the sake of operability, maintenance and cost, having available tool capable to detect and isolate faults in real-time has great place in the company's economy. Moreover, such a tool plays a major role in system safety and is useful for optimizing the production cost.

Today, several diagnostic methods are present in the literature. These methods are based mainly on a normal and/or faulty behavioral model of the system. However, each diagnostic method differs according to different criteria. The latter can be for instance, the dynamics of the process (discrete, continuous or hybrid) [Louarji et al., 2013], [Zhao et al., 2005], the implementation of diagnostic (online and/or offline) [Seera et al., 2013], [Dotoli et al., 2009], the distribution (centralized, decentralized or distributed) [Sayed Mouchaweh and Lughofer, 2015], [Keliris et al., 2014], and so one. Therefore, each diagnostic approach must guarantee the following characteristics [Philippot et al., 2011]: diagnosis should be easy to implement, must be conceivable algorithmically, and finally should be feasible in real time.

Therefore, to reach all or part of these characteristics, researchers use three main approaches to design a diagnostic module. Basically, the difference between each approach is related to the place of the diagnostic module with respect to the control system. Thus, the first approach is called the integrated approach, where the diagnosis is taken into account during the control system designing and integrates fault states. The normal and faulty situations are handled, in this case, by the control system [Diao and Passino, 2002], [Xianzhi and Wen, 2010]. The second one is the separate approach which considers the diagnosis as a function of its own. Then, any failures analysis techniques and diagnosis module must be added to the process control [Paoli et al., 2011], [Chakkor et al., 2014]. This separation has the advantage of using the diagnosis models and techniques as an appropriate and effective tools for diagnosis and control.

Finally, the last one is the mixed approach which is halfway between integrated and separate approaches [Combacau, 2001], [Combacau et al., 1998], [Zamai, 1998]. It is particularly suitable for manufacturing processes because it uses an integrated sensor that adapts to running command sequences with a separate diagnostic module using the most appropriate reasoning techniques. As an example, we include in this work few references for successful industrial applications. In [Baggiani and Marsili-Libelli, 2009], authors describe an online fault detection and isolation algorithm based on the principal component analysis and related statistics. The algorithm proved also capable of dealing with three possible kinds of faults: outright sensor failure, spikes, and process anomalies. In [Jung et al., 2006], an online induction motor diagnosis system using motor current signature analysis with advanced signal-and-data-processing algorithms is proposed. The diagnoser can isolate four types of faults such as breakage of rotor bars and end rings, short-circuit of stator windings, bearing cracks, and air-gap eccentricity. Regarding the discrete event system framework, online fault diagnosis is realized often by using Petri nets formalism [Ramirez-Trevino et al., 2007], [Basile et al., 2008]. However, this approach needs the formulation of the fault diagnosis problems in terms of mathematical programming which can be provided a fast online diagnosis but at a price of excessive memory requirements. The main disadvantage of these approaches is the need to behavioral models for components incorporating all

nominal and faulty possible behaviors. This requires that the faults to be diagnosed must be defined in advance and must be included in the components models or in the global model. Therefore, the built diagnoser is static and cannot be adapted in the case of fault occurrence of new types.

An alternative that we propose to deal with this issue is the implementation of an adaptive diagnoser using DEVS (discrete event system specification) formalism. Indeed, DEVS formalism is a structure which can describe the different aspects of deterministic and causal systems in classical systems theory [Zeigler, 1976]. Furthermore, its structure allows the description of system behavior at different levels. As the DEVS formalism is a general formal method, it fundamentally satisfies common advantages of the formal method (e.g. finite state automaton (FSA), Petri nets), such as consistency, completeness or verification. In [Jacques and Wainer, 2002, Zheng and Wainer, 2003] we can see that DEVS is mostly easy to implement compared to standard tool like FSA or Petri-Nets, and offers an immediate practical application for testing new technical approach and/or analyzing weapon performance with various scenarios. Thus, with this approach, we will be able to implement a diagnoser which can be adapted in order to integrate new unknown specific fault behaviors into its inference engine. This adaptation allows increasing the system diagnosability over time. Therefore, the built diagnoser by the proposed approach is dynamic in the sense that new unknown specific fault behaviors is integrated into its inference engine.

So, this paper is organized as follows. In Section 2 we formulate the problem to build a DEVS model of an individual component of the system. Then, in Section 3, we describe the method to design an adaptive diagnoser model based on temporal behavior for a class of discrete event system. In Section 4, we present the benchmark that we used to validate our diagnoser and give some experimental results. And at the end, in Section 5, we'll give the conclusion of this work.

2. DOUBLE ACTING CYLINDER MODEL

2.1 The atomic DEVS formalism

Several DEVS formalisms evolved from the original one to introduce specializations for different purposes ([Kofman and Castro, 2006], [Chen et al., 1999]). In this paper, we introduce the original DEVS formalism. This one defines a DEVS model to be a structure:

$$M \equiv \langle X, Y, S, \lambda, \delta_{int}, \delta_{ext}, ta \rangle \quad (1)$$

where X is the set of input event values. Y is the set of output event values. S is the set of state values. $\lambda, \delta_{int}, \delta_{ext}$ and ta are functions which define the system dynamics.

Each sequential state $S_i (S_i \in S)$ has an associated *dwell-time* calculated by the time advance function $ta(S_i)$ ($ta(S_i) : S \rightarrow \mathbb{R}_0^+$). The time advance function defines the times saying how long the system remains in a given sequential state in absence of input events. This leads on the following definition:

Definition 1. (Dwell-time). The elapsed time e takes on values ranging from 0 to $ta(S_i)$. Often, the Dwell-time σ in a state is used: $\sigma = ta(S_i) - e_i(t)$. In other terms, dwell-time is based on the comparison between the time to perform a task to a reference time. It can be useful to tell us whether the system is in fail operating mode or not.

Thus, if the state adopts the value S_i at time t_i, after $ta(S_i)$ units of time (i.e., at time $ta(S_i) + t_1$) the system performs an internal transition, going to a new state S_j. The new state is calculated as $S_j = \delta_{int}(S_i)$, where $\delta_{int} : S \rightarrow S$ is called internal transition function. When the state goes from S_i to S_j an output event is produced with value $Y_i = \lambda(S_i)$, where $\lambda : S \rightarrow X \cup \{\phi\}$ is called output function. Functions ta, δ_{int}, and λ define the autonomous behavior of a DEVS model. When an input event arrives, the state changes instantaneously. The new state value depends not only on the input event value but also on the previous state value and the elapsed time since the last transition. If the system goes to the state S_k at time t_k and then an input event arrives at time $t_k + e$ with value X_k, the new state is calculated as $S_{(k+1)} = \delta_{ext}(S_k, e, X_k)$ (note that $ta(S_k) > e$). In this case, we say that the system performs an external transition. Function δ_{ext} ($\delta_{ext} : S_i \times X \rightarrow S_j$) is called the external transition function. No output event is produced during an external transition. This allows us to define the transition relation T which is a subset of $(S \times X \times S)$ as follows: $T = \left\{ \langle S_i, e, X_k, S_j \rangle : \exists \text{ transition from } S_i \text{ to } S_j \text{ by } X \right\}$ computed over arbitrary time intervals (e).

2.2 DEVS model for double acting cylinder

In this section, we give a DEVS formulation to build a model for a double acting cylinder (DAC) system. Basically, a DAC has two main states. The first one defines the behavior when the cylinder is completely opened which we call by S_1 and the second one defines the behavior when the cylinder is completely closed which we call by S_2. More precisely, each state is defined according to the information gets via two sensors placed on each other of the actuator. In other terms, input events mark respectively the High (extension mode) , V_s, and the Low position (retraction mode) , V_r, of the actuator. For diagnosis purposes, we use another level of abstractions for DAC model while considering the command orders, O_r and O_s, as two other input events. These are two control inputs that we use to lead the actuator respectively in Low and High position. In this case the set of input event values X can be defined by the combination of each input event variable, $\mathscr{C}\{O_s, O_r, V_s, V_r\}$. In other terms, the set of input events can be defined as follows: $X = \{X_0, X_1, \cdots, X_{15}\}$, which are associated with the enable events set $Ev = \{ev_1, ev_2, ev_3, ev_4\}$ where

- $ev_1 = \downarrow O_s, \downarrow O_r, \downarrow V_s, \uparrow V_r$, $ev_2 = \downarrow O_s, \downarrow O_r, \uparrow V_s, \downarrow V_r$,
- $ev_3 = \downarrow O_s, \uparrow O_r, \downarrow V_s, \downarrow V_r$, $ev_4 = \uparrow O_s, \downarrow O_r, \downarrow V_s, \downarrow V_r$.

Here, each enable event is characterized by either rising $\uparrow X_k$, or falling $\downarrow X_k$ input event. So, the complete equivalent model of DAC taking into account all possible states where the system can be evolved is illustrated in Figure 1 below. Herein the reader can see that DAC model is composed by 16-states described by the following set: $S = \{S_0, S_1, \cdots, S_{15}\}$. If any event not occurs before the defined time-advance of each state, $ta(S_i)$, the transition from one sequential state to another one is performed by the internal transition function $\delta_{int}(S_i)$. If an external input event occurs, then a new state is computed by the resulting of the current state, the elapsed time, and the input event. Thus, the external transition function can be re-defined as: $\delta_{ext}(S_j) = \left\{ \forall ev_i \in Ev, \forall S_k \in S, S_k \oplus ev_i \rightarrow S_j \in S \right\}$. Here, the symbol '$\oplus$' denotes the logical operator Exclusive-OR.

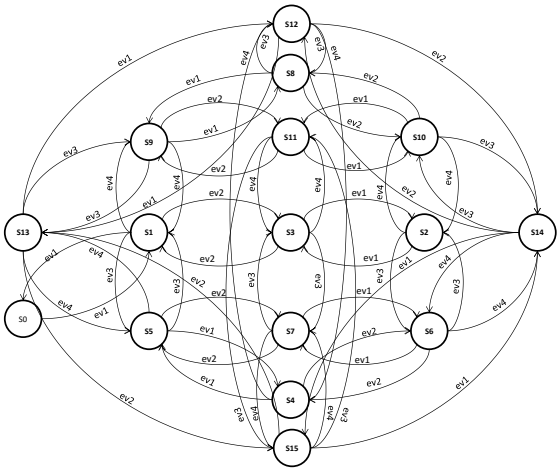

Fig. 1. DAC model functioning with all possible states.

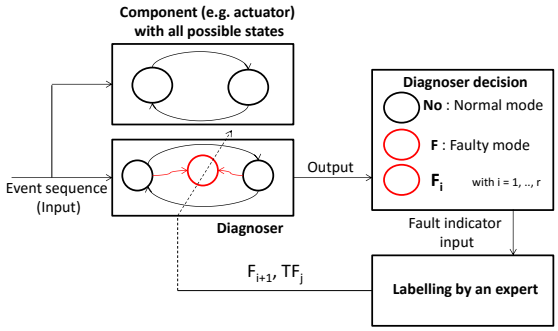

Fig. 2. Proposed approach scheme

3. ADAPTIVE DIAGNOSER DESIGN

Knowing that for a pure discrete-event systems, the temporal notion consists just of temporal ordering constraints among state transitions. Therefore, to design an online adaptive diagnoser model, it is possible to explore this temporal information to know whether the system is in fault operating mode or not. Indeed, the consequences of the fact that the system is in a specific (normal or faulty) state can be represented as a violation of some temporal specifications or constraints between the generated events by the system. This means the diagnoser should take into account the observations and their temporal information. Therefore, a failure is specified as an execution of an event leading to a violation of a temporal specification $C(t)$ defined based on the input events X. Globally, the proposed approach is illustrated in figure 2. The diagnoser is compiled using the local models of system components. Initially, it includes only the normal (desired) behavior. Then, it is enriched by integrating new specific fault behaviors represented as a supervision patterns. This is particularly interesting since it is hard to determine in advance all the faulty behaviors with their fault partition labels. The desired behavior is considered in the sequel as a set of specification, $R = (X, C(t))$, generating a switching sequence S. A fault, F_i, is the result of the execution of a specific behavior represented by $S_f \in S$. S_f violates specification R and can be seen as a class representing the fault behaviors of type TF_j. Briefly, knowing the type of the faults labelling by an expert, we propose to adapt the diagnoser model to take into account this isolation due by the expert to make easy the maintenance procedure in the future.

So, the system we consider is composed of a set of $\Gamma = \{1,...,c\}$ components. The local model, M_c, of a component, c, represents all the feasible behaviors that this component can execute in response to a control command. Let M be the system global model. Thus, in this section the objective is to design an optimal temporal real-time diagnoser model defined by the following structure:

$$D(t) = w(t) \cup M_c^+. \tag{2}$$

In this equation $w(t)$ gives an abstraction of the temporal behavior of the actuator and M_c^+ relates the constraints to perform a given task. The formal definitions both of the temporal behavior abstraction and the structure constraints are given in the following paragraph.

so, let us suppose that for any observation in state domain Ω, the structure is subject to the following constraints:

$$M_c^+ \equiv \langle X_c^+, Y_c^+, S_c^+, \lambda^+, \delta_{int}^+, \delta_{ext}^+, ta_c^+ \rangle \tag{3}$$

where $.^+$ is a subset of $\Omega \cup \{obs\}$ such that

$$X_{obs,i} : X_i \to X_{obs}; \quad S_{obs,i} : S_i \to S_{obs}; \quad Y_{obs,i} : Y_i \to Y_{obs}.$$

These constraints lead on the following definition.

Definition 2. (Sequential state observability). We consider that the sequential state $S_i \in S^+$ is observable in real-time case if there exists a deterministic transitions between sequential states as well as how it reacts to an external input (events) X^+ and how it generates an output (events) Y^+. Otherwise, it is a non-observable sequential state.

This definition relates that sequential state setting may be split into two sets and can be re-written as: $S = S^+ \cup S^-$; where S^+ defines the set of states belonging to the desired behavior, while S^- includes all the states reached due to the occurrence of faults. To determine each sub-element of each set, an iterative reachability search is performed during a normal operating mode and for few observations N. So, the iteration step can be computed by:

$$S^+(k+1) = S^+(k) \bigcup \{S_i : \exists S_j \in S^+ : \langle S_j, S_i \rangle \in T\} \tag{4}$$

with $i = 1,...,n$ and n the state number according to the input events combination. The observable (resp. reachable) set $S^+(k)$ is repeatedly expanded by adding all states reachable in one step from $S^+(k)$. So, to explain simply how this procedure works, let us considerate the following definitions.

Definition 3. Let us assume that the state observable set $S^+(k)$ can be replaced by a flag function $\chi(k, S_i)$ defined as:

$$\chi(k, S_i) = \begin{cases} 1 & S_i \in S^+ \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

and by the same way, as the transition relation T is a subset of $S \times S$, we suppose that it can be replaced by a boolean flag function $\tau(S_j, S_i)$. Thus, using these definitions, equation (4) becomes:

$$\chi(k+1, S_i) = \chi(k, S_i) \wedge [\exists S_j : \chi(k, S_j) \vee \tau(S_j, S_i)] \tag{6}$$

Here, the characteristic functions χ and τ can both be directly represented as a pure logical expression and therefore can constitute a well-defined operation if for each iteration, $k = 1,...,N$, the sequential state S_i is included in observable set S^+ or not. We obtain finally the following result:

Proposition 1. (Full covering diagnoser). The temporal diagnoser model, defined by equation (2), is called full covering diagnoser if : for N, for M^+ and $\forall S^+$ where $S^+ =$

$\{S_i \in S \mid \forall S_j \in S\}$, we have $\lambda(S_i) \neq \lambda(S_j)$ if $\exists X_i \in X^+$, and $e_i \in \mathbb{R}^+$ such that $S_j = \delta^+_{ext}(S_i, e_i, X_i)$.

Proof 1. This condition allows us to verify if during a few cycles, a set of an observable (resp. reachable) state can be computed over arbitrary intervals. For this purpose, an iterative reachability search is started by using equation (6). So, for each cycle n, $(n = 1,...,N)$, and that for N normal operating cycles, all reachable states are added progressively in the observable state S^+ if the transition from S_i to S_j exists and performed according to the defined time-advance function. In this case, M^+ relates the real observation during overall cycles by S^+, then the diagnoser can cover fully the overall observation. This closes this proof.

However, to obtain an optimal full covering Diagnoser, special care has to taken for the temporal behavior of the system. For this purpose, let us consider the following definition.

Definition 4. (Temporal constraints). By definition the set of temporal constraints expresses constraints between maximum and minimum time for each actuator of the system to evolve from one state to another one. The temporal constraints are given by:

$$C(t) = \{[\underline{t}a(S_1), \bar{t}a(S_1)], ..., [\underline{t}a(S_k), \bar{t}a(S_k)]\} \quad (7)$$

where k is the maximum state number of the system, $\underline{t}_a(S_i)$ and $\bar{t}_a(S_i)$ are respectively the smallest and the largest time remaining until internal transition. Notice that each time advanced $ta(S_i) = t_i$ value will be computed by experimental way during a few normal operating mode N.

These allow one to account for the possible timestamps of observed events, if available, in order to find out their reciprocal order. Notice that if the system evolves in unobservable sequence, one or more of its state transitions are abnormal.

Definition 5. (Temporal behavioral model). So, the temporal behavioral model $w(t)$ can be defined as a formula

$$w(t) = b(S_i, \sigma) \wedge C(t) \rightarrow S^+ \vee S^-, \quad (8)$$

where $b(S_i, \sigma)$ assumes behavioral mode of the state S_i during the dwell-time σ. Temporal Behavior may be associate with such a behavior, e.g:

$$w(t) \rightarrow \exists \sigma'(\sigma' \; meets \; C_1(ta(S_1)) \vee, ..., \vee \sigma' \; meets \; C_c(ta(S_c)))$$

$$\Rightarrow w(t) \rightarrow S^+ \text{ otherwise } \; w(t) \rightarrow S^-$$

Proposition 2. (Optimal full covering Diagnoser). The temporal diagnoser model describe by relation (2) covers the overall observation and the diagnosis is optimal after transition if:

- (i) the diagnoser covers fully the desired behavior,
- (ii) for $S^+ = \{S_i \in S | \forall S_j \in S\}$, $\exists \sigma \in \mathbb{R}^+$ where $b(S_j, \sigma) \wedge C(t) \rightarrow S^+ \vee S^-$, i.e every switching sequence takes a finite time interval,

Proof 2. The first condition is trivially required to obtain a full covering diagnoser according to the proposition 1. This is necessary to reconstruct the complete cycle to perform a given task according to the complete observation of the sequential state. This condition is mainly necessary for a real-time computation feasibility. In the other hand, the second condition allows one to determine all states which are non-deterministic time advance function, as silent cycle and/or unobservable sequence state. If right, for each observation n, $(n = 1,...,N)$, and that for N normal operating cycles, all reachable (resp. not-reachable) states are added progressively in the observable (resp. unobservable or faulty) state setting S^+ (resp. S^-). In this case, the diagnoser $D(t)$ relates the optimal observation during overall cycles defined by S^+ and based on the set of specification R. End of proof.

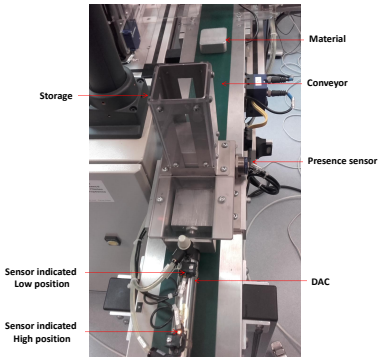

Fig. 3. DAC real system mounted onto the benchmark

As the desired behavior is considered as a set of specification, R, generating a switching sequence $S = S^+ \cup S^-$, a fault f is then the consequence of a specific behavior represented by the faulty (resp. non observable) set S^-. In other terms, a fault f_i occurrence can generate a faulty set $Sf_i \subseteq S - S^+$ with $i = 1,...,r$. Since, the diagnoser $D(t)$ is computed in order to confirm whether the system is under normal operation conditions (No) or not (F). In the case of fault behavior, $X_f \subseteq S - S^+$, and if $D(t)$ contains the fault behaviors corresponding to the occurrence of a fault of type TF_j, with $j = 1,..,r$, then $D(t)$ issues the decision TF_j indicates that a fault belonging to one of the predefined fault setting S^- occurred. When $X_f \subseteq S - S^+$ does not belong to any of the fault behaviors already included in the diagnoser $D(t)$, then the expert is asked to provide a fault label to this fault sequence. Two cases can occur. Either this fault sequence belongs to already existing S^- or it belongs to a new fault type TF_{j+1}. In the former case, the sequence $Sf_i \subseteq S - S^+$ of S^- will be updated by integrating X_f as follows:

$$Sf_{j+1} = Sf_j \cup X_f, \quad (9)$$

To associate with the diagnoser $D(t)$ a new type of faults, an iterative approach is performed during the whole observations. So, the iteration step can be computed by:

$$D(t+1) = D(t) \cup Sf_{j+1} \quad (10)$$

with $t \in \mathbb{R}^+$ and $j = 1,...,r$ and r the fault state number according to the specification constraints. The adaptive diagnoser $D(t+1)$ is repeatedly expanded in this case by adding all possible faulty states labeled by the expert in one step from $D(t)$.

4. EXPERIMENTAL RESULTS

Real-time tests are performed on an automated system available into our lab . This system is designed for heating treatment of metal parts. The DAC system studied is shown in Figure 3. This DAC is equipped with two sensors to indicated if the DAC is in Low or High position. The low position request is performed when the presence sensor informed the controller that there exists a material in the storage part to push on the conveyor. The control of the system is achieved by a TSX Micro PLC (programmable logic controller).

Regarding the diagnoser, it is implemented by using Stateflow into Matlab^{TM}/Simulink, and it is linked with the PLC via the OPC factory server (OFS) of Schneider. Now for configuring

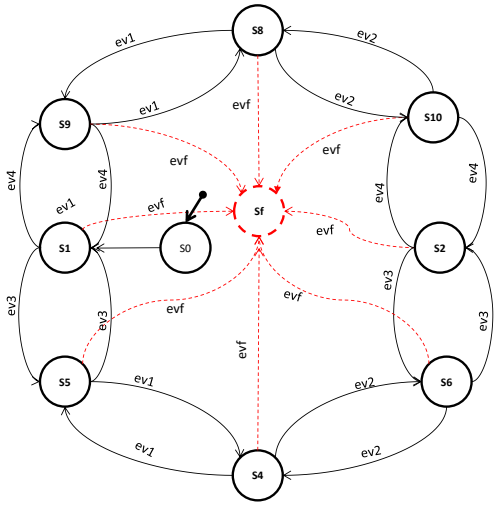

Fig. 4. Optimal full covering DAC diagnoser functioning with a global faulty set. Black solid line as expected behavior. Red dashed line as abnormal behavior

Table 1. Temporal constraints for each state

Dwell Time	t_1	t_9	t_8	t_{10}
Numerical value (ms)	$[0-\infty[$	$[5-15]$	$[20-85]$	$[0-1]$
Dwell Time	t_2	t_6	t_4	t_5
Numerical value (ms)	$[0-1]$	$[7-15]$	$[28-60]$	$[0-1]$

the OFS, we just need to update the symbol table onto Unity Pro, export it and load it by using the OFS utility. Into Simulink the configuration of the OPC client is necessary to realize the connection between OFS and Matlab^{TM}. Herein, the first level is then the plant floor devices. The PLC is used to control and to acquire data for the application. The second level is the place of the OFS/OPC server which let us to configure and save data that we have need for diagnosis purposes. And the last level is the communication between Matlab^{TM} software and the PLC. In Matlab^{TM}, we construct our diagnoser model based on DEVS formalism and generate all informations which indicate us the system status.

Figure 4, represents the optimal full covering DAC diagnoser according to the two propositions mentioned before and that has been implemented in your case. Each normal sequential state which defines the normal desired behavior is considered here as a passive state (black solid line). In other terms, the switching from one state to another one depends on the computation of the external transition function. Moreover, in absence of input events the switching is performed by the internal transition function. In this case, the system switch into an abnormal situation defined by the set $S^- = S_f$. So, each value of the set of temporal constraints for each normal passive state is computed by experimental way, i.e we let the system run for a few cycles (here 100 cycles) during a normal operating mode and we save after that the minimum and maximum of the dwell-time. The normal dwell-time is given in table 1 for our DAC.

In the sequel, we apply onto the system two kinds of actuator faults. The first one is applied when the DAC is completely opened with close order enabled and the second one is applied when the DAC is in completely closed position with open order enabled. In figure 5 we can see the first default simulation after a few normal cycles. Here, the DAC is completely opened and switches from S_1 to S_8, and it is blocked into this state. As it shown the time of detection of a blocked state during an extension phase is detected less than $1s$. Thus, the system switches into the faulty state S_f when the temporal constraints defined in table 1 are not satisfied. Moreover, the detection time of the abnormal situation is acceptable for this class of system.

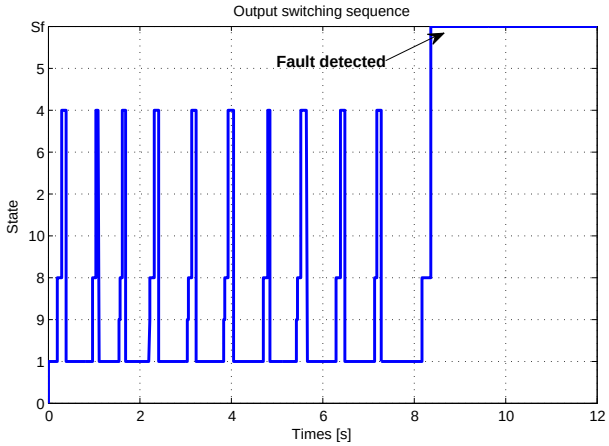

Fig. 5. DAC control with fault simulation during extension action

Since fault is detected by the diagnoser module, the expert is alarmed by this event and start to label this occurrence in order to integrate new unknown specific fault behaviors into its inference engine. Consequently the diagnoser model is updated as well as the faulty set $S^- = \{Sf, Sf1\}$.

To confirm this result we apply another default on the system. We blocked the actuator into the state S_4. The result is given in figure 6 below. Once again, we see that the time of detection of

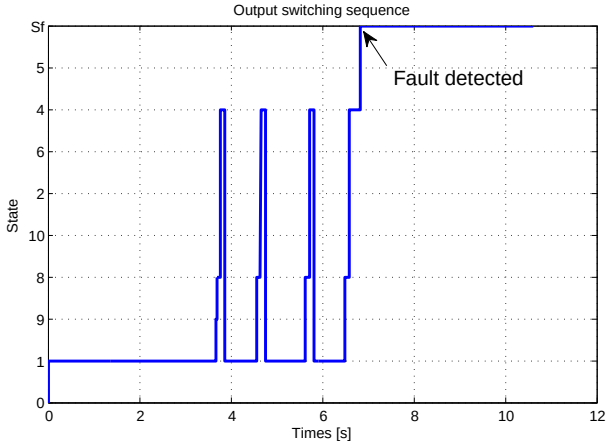

Fig. 6. DAC control with fault simulation during retraction action

a blocked state during this phase is detected at least $1s$. More precisely the system switches into the faulty state S_f when the normal switching sequence is not reached after $60ms$, which is the maximum dwell-time authorized for the state. Notice that the detection time of the faulty situation depends mainly on the communication time between the PLC and Matlab software but not on the diagnoser model. So, in this case the faulty set is defined by $S^- = \{Sf, Sf1, Sf2\}$. The adaptive diagnoser module is finally given in figure 7, taking into account the two scenarios that we have produced before. Here we can see that

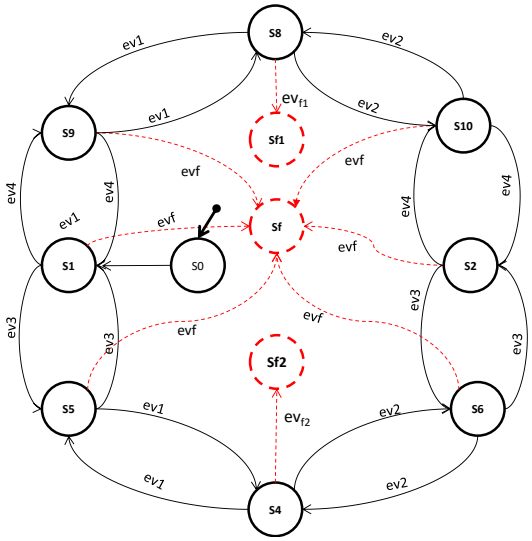

Fig. 7. Adaptive diagnoser according to a stuck-on and stuck-off faults.

the approach can adapt online the diagnoser in order to integrate new specific fault behaviors into its inference engine over time.

5. CONCLUSION

Components of industrial processes are often affected by un-permitted or un-expected deviations from normal operation behaviour. The fault detection task consists of determination of the fault present in a system and the time of detection. In this paper, an adaptive fault diagnosis approach is proposed in order to perform the fault diagnosis of manufacturing systems, in particular actuators faults (stuck-on, stuck-off). The desired (normal) behavior is represented by a set of temporal specifications while faults are considered to be the execution of specific fault behavior violating one of these specifications. The inference of the fault type of each fault is achieved by a diagnosis module called diagnoser. Diagnoser's model is based on DEVS formalism and two conditions are given to verify if the diagnoser covers fully and optimally the observation.

REFERENCES

F. Baggiani and S. Marsili-Libelli. Real time fault detection and isolation in wwtpse. *Water Science and Technology - WST*, 60(11), 2009.

F. Basile, P. Chiacchio, and G. De Tommasi. Online diagnosis of discrete event systems based on petri nets. In *9th International Workshop on Discrete Event Systems. WODES 2008*, pages 436–442, May 2008.

S. Chakkor, M. Baghouri, and A. Hajraoui. Real time remote monitoring and fault detection in wind turbine. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 8(9), 2014.

L. Chen, S. Xia, and P. Luker. A conceptual framework for modeling and simulation of variable structure hybrid systems. *AAAI Technical Report SS-99-05*, 1999.

M. Combacau. Approche discrète de la surveillance des systèmes de production. *Traité IC2*, 11, 2001.

M. Combacau, E. Zamai, and A. Subias. A discrete events model of a monitoring strategy as a control structure of a monitoring system. In *Proceeding on Multiconference IMACS CESA'98*, Hammamet, Tunisie, April 1998.

Y. Diao and K. M. Passino. Intelligent fault-tolerant control using adaptive and learning methods. *Control Engineering Practice*, 10:801–817, 2002.

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*, 45(11):2665–2672, 2009.

C. Jacques and G. Wainer. Using the cd++ devs toolkit to develop petri nets. In *Proceedings of the 2002 Summer Computer Simulation Conference*, San Diego, CA, 2002.

J-H. Jung, L. Jong-Jae, and B-H. Kwon. Online diagnosis of induction motors using mcsa. *Industrial Electronics, IEEE Transactions on*, 53(6):1842–1852, Dec 2006.

C. Keliris, M.M. Polycarpou, and T. Parisini. A distributed fault diagnosis approach utilizing adaptive approximation for a class of interconnected continuous-time nonlinear systems. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 6536–6541, Dec 2014.

E. Kofman and R. Castro. Stdevs, a novel formalism for modeling and simulation of stochastic discrete event systems. In *Proceedings of AADECA 2006*, Buenos Aires, Argentina, 2006.

H. Louarji, M. Sayed Mouchaweh, and C. Labarre. Diagnoser with hybrid structure for fault diagnosis of a class of hybrid dynamic systems. *Chemical Engineering Transactions*, 33 (1):85–90, 2013.

A. Paoli, M. Sartini, and S. Lafortune. Active fault tolerant control of discrete event systems using online diagnostics. *Automatica*, 47(4):639 – 649, 2011.

A. Philippot, M. Sayed Mouchaweh, V. Carré-Ménétrier, and B. Riera. Generation of candidates' tree for the fault diagnosis of discrete event systems. *Control Engineering Practice (CEP)*, 19(9):1002–1013, 2011.

A. Ramirez-Trevino, E. Ruiz-Beltran, I. Rivera-Rangel, and E. Lopez-Mellado. Online fault diagnosis of discrete event systems. a petri net-based approach. *IEEE Trans. on Automation and Engeenering*, 4(1):31–39, Jan 2007.

M. Sayed Mouchaweh and E. Lughofer. Decentralized fault diagnosis approach without a global model for fault diagnosis of discrete event systems. *International Journal of Control*, 0(0):1–14, 2015.

M. Seera, C.P. Lim, D. Ishak, and H. Singh. Offline and online fault detection and diagnosis of induction motors using a hybrid soft computing model. *Applied Soft Computing*, 13 (12):4493 – 4507, 2013.

Z. Xianzhi and S. Wen. Design and implementation of an embedded diagnosis system for radar built-in test equipment. In *International Symposium on Intelligence Information Processing and Trusted Computing (IPTC), 2010*, pages 173–176, Oct 2010.

E. Zamai. A distributed fault diagnosis approach utilizing adaptive approximation for a class of interconnected continuous-time nonlinear systems. In *IFAC symposium on Information Control in Manifacturing*, Nancy, June 1998.

B.P. Zeigler. Theory of modeling and simulation. *Wiley and Sons*, 1976.

F. Zhao, X. Koutsoukos, H. Haussecker, J. Reich, and P. Cheung. Monitoring and fault diagnosis of hybrid systems. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 35(6):1225–1240, Dec 2005.

T. Zheng and G. Wainer. Implementing finite state machines using the cd++ toolkit. In *Proceedings of the 2003 Summer Computer Simulation Conference*, Montreal, QC. Canada, 2003.