Diagnosis of critical embedded systems: application to the control card of a railway vehicle braking systems

R. Saddem, A. Toguyeni, and M. Tagina

Abstract— **Embedded systems are more and more used to control critical systems. In this paper, we propose a diagnostic approach to increase the security of control of critical embedded system based on digital components. This is a part of a study to design of an electronic card to control a railway vehicle braking system. Because of the critical aspect, it is necessary to diagnose the failures of the control card to process them online for safety purposes. In this paper, we propose to use diagnoser technique based on timed automata. But since this technique suffers of combinatorial explosion and because digital devices are characterized by a lot of input/output, our approach proposes to make an abstraction of the system behavior to reduce the size of the models and to implement a kind of distributed diagnosers.**

I. INTRODUCTION

Nowadays, embedded systems are more and more used to control complex systems. In this study, we are interested in critical embedded systems that can be used in the control of transport systems like railway systems or autonomous intelligent vehicles. These systems are critical since in case of misbehavior, it is necessary to react very quickly in order to keep them in a safety working mode.

Generally, the implementation of embedded systems is based on COTS (commercial off-the-shelf) such as microcontrollers or FPGA. These components can have failures that can bring the controlled system to an unsafe state. For that reason, one tries to develop redundant control architecture that enables to implement fault-tolerant control. This requires the necessity to detect and isolate quickly the failures of the control system. In this paper, we will focus on the example of the control of a railway vehicle braking system given in [1].

Electronic devices with Boolean inputs/outputs are discreet event systems (DES hereafter). This study aims to propose an efficient method based on model checking technique and the notion of diagnoser to identify online the failures of such systems. But the large number of input-output leads to a combinatorial explosion which renders a priori unenforceable the diagnoser approach. In this paper, we propose an approach based on temporal abstraction of system behavior and a kind of distributed implementation to solve this combinatorial problem.

The paper is structured as follows. In section 2, we will first recall the main techniques used for the diagnosis of DES. In section 3, we will present the modeling tool adopted: the formalism of timed automata that will be used in the rest of the paper. In section 4, we will propose a generic model of the complete behavior of an electronic control card. This model is one of the main contributions of this paper since it will give the capacity to build the diagnoser to survey this card. In section 5, we will propose a method to build and implement the diagnoser of the fault-tolerant control architecture.

Manuscript was received on March 11, 2011.

R. Saddem is with Université Lille Nord de France/Ecole Centrale de Lille, Laboratoire d'Automatique, Génie Informatique et Signal, Lille, France and with Université de la Manouba, Ecole Nationale des Sciences de l'Informatique, Stratégies d'Optimisation des Informations et de la connaissancE, Manouba, Tunisia (ramla.saddem@ec-lille.fr).

A. Toguyeni is with Université Lille Nord de France/Ecole Centrale de Lille, Laboratoire d'Automatique, Génie Informatique et Signal, Lille, France (armand.toguyeni@ec-lille.fr).

M. Tagina is with Université de la Manouba, Ecole Nationale des Sciences de l'Informatique, Stratégies d'Optimisation des Informations et de la connaissancE, Manouba, Tunisia, (moncef.tagina@ensi.rnu.tn).

II. DIAGNOSIS OF DES: A STATE OF THE ART

In DES, there are observable events, coming from information sent by the sensors or/and command orders, and other unobservable events such as failures. The objective of the diagnostic procedure is to detect and/or isolate (when possible) those failures from the occurrences of observable events. In DES, there are two basic approaches for diagnosis; the diagnoser based approach [2] and the chronicles based approach [3].

A. Diagnoser

A diagnoser is an observer automaton in charge of identifying faults in a system. An observer automaton models all observable behaviors of the system. Strictly speaking, an observer is an automaton whose alphabet can be partitioned into two types of events: the set of observable events Σ and that of unobservable events τ.

The objective of an observer is to identify the current state of the system given the observable events that it generates. There are many studies to observe the process [2]. Most of them are based on finite state automata or timed automata [4]. Traditionally there are two major classes of observers; observers for the states recognition and observers for events recognition or sequences of events recognition.

The observers for the recognition of states also imply two categories of approaches. The first category is inspired by Lin and Wonham's works about controllers' synthesis [5]. It requires the identification of the current state after each reception of the event produced by the process [6] [7]. The second category deals with the identification of the state reached after a finite sequence of events [8].
Some other works are also interested in the recognition of events sequence leading from one state of the system to

another. For example, works relating to the concept of invertibility [9]. A language is called invertible if at any time, using the knowledge from observed events' sequences, one can reconstruct the entire sequence (including unobservable events) over a bounded number of past events [9]. This concept of invertibility can be very useful for failure isolation, but must not be mistaken for the concept of diagnosability proposed by Sampath [10]. Diagnosability is more dedicated to the problem of failure detection.

The major limitation of diagnoser approaches 1ies in the management of the combinatorial explosion related to the formalism of automata. Indeed, the classical approach is to decompose the system into components. Then, an automaton model is associated with each component. The system diagnoser in a centralized approach is then obtained by the synchronous product of automata of its components. This synchronous product leads to combinatorial explosion when the system is complex. To cope with this drawback, some studies are based on Petri Nets (PN) [11]. In section 4, we will propose a modeling based on timed automata and a distributed implementation to solve this problem.

B. Chronicles

Chronicles model an evolution of an observed system. They are adapted to the interpretation over time of occurrence of events produced by a system. In fact, a chronicle consists of a set of partially ordered observable events. These events are usually linked together by temporal constraints on their occurrences. In practice, a chronicle is a description of a temporal pattern defining a partial order on events described by their type and date of occurrence. The relations between events can be logical (conjunction) or temporal (sequence, missing , etc).

Early works on the chronicles have been proposed by McDermott [12]. Other studies have been conducted by A. Toguyéni at LAGIS [13], Malik Ghallab and Christophe Dousson at LAAS [14] [15]…
Thus each abnormal situation can be described by one or more chronicles. In that case, the events can be symptoms or alarms constrained by their date of occurrence. Diagnosis based on chronicles allows to interpret online the occurred events and then instantiate some chronicles patterns. A chronicle is recognized when all its events have occurred with respect to their constraints. This determines whether the system works normally or not.

Generally, a chronicle pattern can be described by "(1)":

$$(In, A, nct)* (In, C, nct) *(A,B, ct1) * (A,D, ct2) * (C, D, ct3)*(D, not E, ct4) \rightarrow G \qquad (1)$$

'A', 'B', 'C',' D', 'E' are events; 'In' is the always-occurring event used as the reference of events that are not constrained ; 'G' is a failure or an unobservable event that can be used in other chronicle; 'cti' is a time constraint that can model a date, a delay or a duration; 'nct' means no time constraint.

Let (A, B, ct1) be a triplet with 'A' as the reference event and 'B' the constrained event. (In, A, nct) means that 'A' is an unconstrained event. (D, not E, ct4) means that 'E' must not occur after 'D' with respect to the temporal constraint 'ct4'. The meaning of (1) is as follows. If one has the occurrences of event 'A' (respectively the occurrence of event 'C) and after that the occurrences of event 'B' meeting the constraint 'ct1' and event 'D' with respect to constraint 'ct2', and if one has the occurrences of event 'D' from 'C' in compliance with the 'ct3' constraint, and if one does not have event 'E' from 'D' with respect to the 'ct4' constraint, then one concludes to 'G'. For recognition, the chronicles can be represented as temporal graphs (Fig. 1) [14][15].

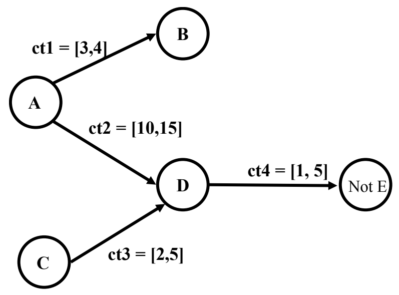


Fig. 1. Temporal Graph

The main advantage of this approach is the relative simplicity to write chronicles from Failure Mode, Effects, and Criticality Analysis (FMECA) completed by an analysis of the temporal behavior of the process. A major limitation in this context is first to be able to guaranty the completeness of the knowledge encoded as abnormal. A second level of difficulty is in interpreting some sequences of events at the input of the diagnostic system under the hypothesis of multiple failures [16]. The problem in this context is the interweaving of the events produced by different failures which can lead to diagnose a superset of failures. In fact, chronicles are generally regarded as surface models knowledge as opposed to deep models that are based on physical properties. Deep models enable to explicit systems' behaviors. They are difficult to obtain. But, the advantage of chronicles is that they allow direct interpretation of the observed events.

III. The Modeling Tool Adopted: Timed Automata

In this work, our objective is to monitor digital systems by the diagnoser approach. To combat the risk of combinatorial explosion, it is necessary to make an abstraction of the behavior of the monitored system to simplify it. Our modeling approach is based on a discrete and temporal abstraction of its behavior that consist in specifying the delay where is expected its report after an order send by the control part. This led us to choose the formalism of timed automata. Not only it allows explicit modeling of temporal behaviors, but it also allows for more compact models than finite state automata. Timed automata

(TA hereafter) are also useful because it exists software tools (like UPPAAL, KRONOS, etc.) to verify the models.

Following its definition in [5], a timed automaton A is a tuple (L, ℓ_0, X, Σ, E, Inv) with: L is a finite set of states; ℓ_0 is the initial state; X is a finite set of clocks to positive real values; Σ is a finite set of actions (that we also called events hereafter); $E \subseteq L \times C(X) \times \Sigma \times 2^X \times L$ is a finite set of transitions; $Inv \in C(X)^L$ associates a constraint to each state.

The synchronous product of timed automata is defined as standard: the automata synchronize on common actions except for the unobservable action.

Let $A_i = (L_i, \ell^i_0, X_i, \Sigma_i, E_i, Inv_i)$, $i \in \{1,2\}$, two timed automata as $X_1 \cap X_2 = \phi$. The synchronized product of A_1 and A_2 is the $A_1 \times A_2 = (L, \ell_0, X, \Sigma, E, Inv)$ automaton defined by : $L = L_1 \times L_2$; $\ell_0 = (\ell^1_0, \ell^2_0)$; $\Sigma = \Sigma_1 \cup \Sigma_2$; $X = X_1 \cup X_2$; $E \subseteq L \times C(X) \times \Sigma \times 2^X \times L$ and $((\ell_1, \ell_2), g_{1,2}, \sigma, R, (\ell'_1, \ell'_2)) \in E$ if :

- Whether $\sigma \in \Sigma_1 \cap \Sigma_2$ and i) $(\ell_k, g_k, \sigma, r_k, \ell'_k) \in E_k$ for k=1 and k=2; ii) $g_{1,2} = g_1 \wedge g_2$; iii) $R = r_1 \cup r_2$;
- Whether for k = 1 or k = 2, $\sigma \in (\Sigma_k \backslash \Sigma_{3-k})$ and i) $(\ell_k, g_k, \sigma, r_k, \ell'_k) \in E_k$; ii) $g_{1,2} = g_k$ and iii) $R = r_k$;
- $Inv(\ell_1, \ell_2) = Inv(\ell_1) \wedge Inv(\ell_2)$.

IV. THE BEHAVIORAL MODEL OF A CONTROLLER CARD

A. *Description of the case study*

In this study we consider the example of a redundant architecture based on three controller cards and a 2/3 voter [1]. Fig. 2 summarizes the main part of this architecture. 'DCi' (with i in {1,2,3}) is a controller card based on a digital COTS (microcontroller, FPGA, etc).

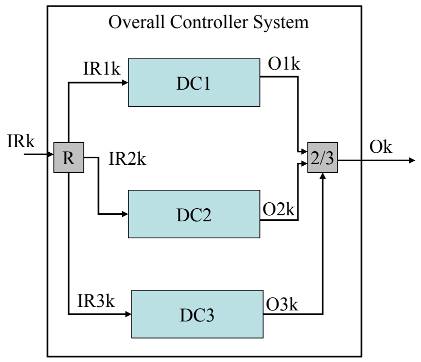


Fig. 2. The three daughter cards system

The overall controller system is composed of three identical 'DCi' cards. It is characterized by inputs and outputs. Each system boolean output, noted 'Ok' here, is a function of its inputs. This function depends on the code that one implements in each 'DCi' card. As an example, if an output depends on 3 boolean inputs, one can write Ok=f(Ix,Iy,Iz). In fig.2, functional block 'R' is a distributer of an input signal to each 'DCi' card. Considering the connector between 'R' and 'DCi', we assume that each daughter card receives an internal signal called 'IRik' at its inputs. For this study, we assume that this internal signal is not observable. The 'Oik' output of each 'DCi' daughter card is processed by the 2/3 voter. Output 'Ok' is equal to 1 if at least two 'Oik' are equal to 1.

B. *The behavioral model*

To build diagnoser one requires a behavioral model of the system. Such type of model can be based on the control code implemented in the 'DCi' card. But such solution presents two drawbacks. First the model and then the diagnosers would not be independent with regard this code. The second drawback would be the high size of the model caused by the complexity of the control. This would induce a combinatorial explosion.

To avoid these drawbacks we must make several abstractions to construct this behavioral model. First, we decouple each of the outputs of the system and we study it separately. After, for each output, the first abstraction consists in avoiding considering directly each inputs of the overall card. In the behavioral model related to an output 'Ok', one considers a reference signal that is noted 'IRk'. To understand this abstraction, let us consider the following example. For Ok=f(Ix,Iy,Iz), let us assume that 'Ok' takes the value 1 after Ix and Iy rising edge move to 1, and Iz falling edge move to 0 (Ix, Iy and Iz are boolean inputs). We are not interested in the order of these events. Once a sequence of these events is obtained, the reference signal, 'IRk' is issued (fig. 2). This initial behavior is modeled by a Mealy automaton in fig. 3. In Mealy automaton, the interpretation of a transition ei/eo from state x to state y is as follows : when the system is in state x, if the automaton receives input event ei, it will make a transition to state y and in that process will emit the output event eo.

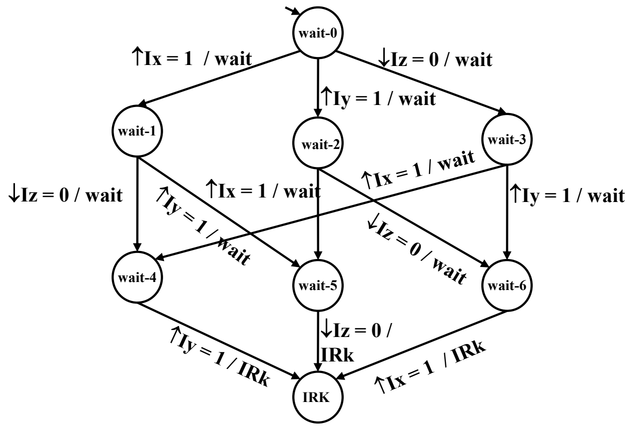


Fig. 3. Reference input signal's generation

To avoid the combinatorial explosion, the second abstraction consists in reducing the normal functioning of a 'DCi' card to a temporal specification between the reference signal 'IRk' and the output 'Oik'. Let us assume that this output is given between 2 and 9 time units (t.u. hereafter) after the occurrence of the reference event. Fig. 4 gives the temporal specification (Fig. 4. a) and the corresponding timed automaton that can be built (Fig. 4. b).

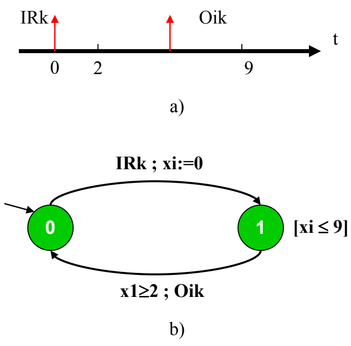


Fig.4. Abstraction of the normal behavioral model of 'DCi'

To construct the diagnoser to survey this system, one must complete this model taking into account its failures. We will call 'fi' the failure of 'DCi'. Fig. 5 represents the behavioral model of 'DCi' with 'fi' which is an unobservable event that can occur before or after 'IRk'. If it occurs before 'IRk', the model reaches state 3. If it occurs after 'IRk' the system goes from state 1 to state 4. In all cases, it will reach state 4. After 9 t.u., if 'Oik' does not occur, we conclude that DCi fails. The guard of the transition between states 4 and 5 models the timeout. 'ri' is an event that models the repair of the card. It is an observable event that brings back the 'DCi' to its initial state. In the diagnoser technique, one cannot process transition without event as transition from state 4 to state 5 (Fig. 5). To solve this problem, let us take more consideration about how the whole system works.

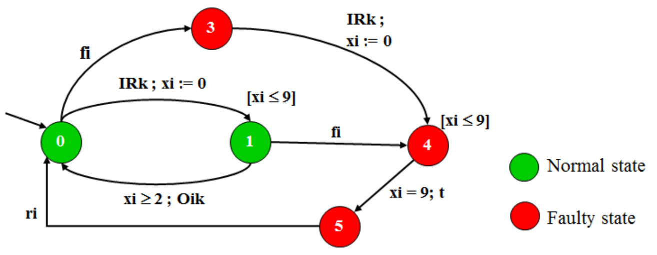
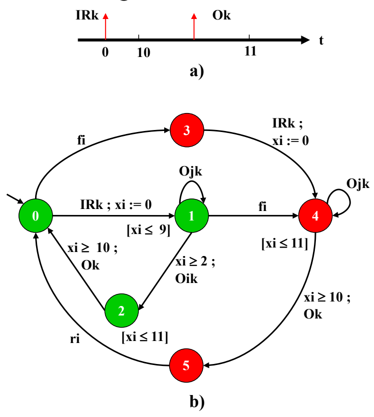


Fig.5. First behavioral model of 'DCi'

The 2/3 voter gives us information that makes the behavioral model of each card better. In fact, in case of CDi failure, after the occurrence of 'IRk', all 'Ojk' (j≠i) occur and the voter can produce 'Ok' between 10 and 11 t.u. This information is then added to our model. The model represented in Fig. 6. b. is then obtained. The transition between states 4 and 5 is fired between 10 t.u. and 11 t.u as soon as 'Ok' occurs. In section VI.C we will discuss the validity of this model.

One performs a synchronous product to construct the behavioral model of the general controller system. Tools like KRONOS allow obtaining timed automaton synchronous products. The global model contains 54 states and 124 transitions. It is therefore not represented here. For simplicity purposes, let us consider here a system composed of only two 'DCi' daughter cards with the ad hoc functional block to replace the voter. The behavioral model obtained is shown in Fig. 7.



Evt	Description	Observable?
IRk	Reference input of OK	Yes
Ok	Voted output number k	Yes
Oik	Output of Dci	Yes
Ojk	Output of DCj	Yes
fi	Failure of DCi	No
ri	Reparation of DCi	Yes

Fig.6. Second behavioral model of 'DCi'

V. CONSTRUCTION OF THE DIAGNOSTIC OF THE FAULT-TOLERANT CONTROL SYSTEM

The objective of this section is to propose a method to construct the diagnostic task of the fault-tolerant control system (the controller card). In that case, two problems must be addressed:

Problem 1 (Modeling problem): Which model must be used to construct the whole system diagnoser? The local model of each 'DCi' daughter card or the general model of the fault-tolerant system?

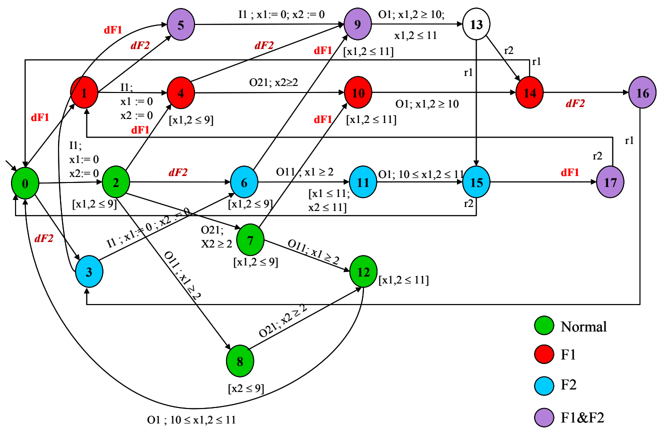


Fig.7. Behavioral model of the two daughter cards system

Problem 2 (Implementation problem): Can we have the same diagnostic with several local diagnosers that evolve in parallel or a single general one (Centralized approach).

The last problem has been already addressed in the literature. It is close to the problem of centralized diagnose approach vs decentralized approach without or with coordinator vs distributed approach with communication among local diagnosers. These different approaches depend on the nature of the behavioral model of the considered system (local or general model) and the nature of the observations (local or general). Decentralized approaches are based on a general model of the system and a local or partial observation of the system observable events. This means that the diagnoser is local since it is constructed from a projection of the general model in the space defined by the set of local observations. Distributed approaches are based on local models and local observations. Thus, they require communications among the local diagnosers to be able to obtain a coherent understanding of the global state of the system.

In this study, the way each 'DCi' daughter card is built implies that each sub-system uses a general observation. Consequently, is it possible to construct a general diagnoser from local diagnoser of each 'DCi' daughter card? Is it possible to implement each local diagnoser independently of the other?

A. Synchronous product of sub-system diagnosers

The idea here is to build a diagnoser resulting from the synchronous product of the diagnoser of each daughter card of the system. For DES modeled by finite state automaton, one uses the determination technique to build the diagnoser. But, timed automata are not always determinable [17] and the test to prove if one is determinable is undecidable [18]. So we built the behavioral model of daughter card 'DCi' to be able to apply determination technique. Application of this technique to the model of fig. 6 gives us the diagnoser of fig. 8. The initial state 'a' is uncertain because it is a meta-state that groups state 0 and state 3 of the behavioral model (noted <0,3> in fig. 8). Indeed, because fault 'fi' is an unobservable event, the 'DCi' can go from state 0 to state 3 without any observable event.

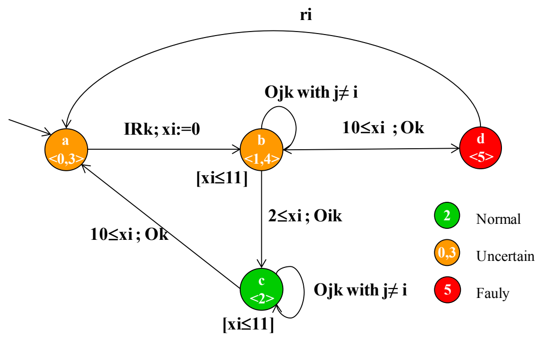


Fig.8. Diagnoser of 'DCi' with repect to output 'Ok'

The same reasoning enables us to build state 'b' (it groups together the basic state 1 and state 4). The occurrence of the 'IRk' observable event causes the transition from state 'a' to state 'b'. If 'Oik' occurs before 9 t.u. the diagnoser reaches state 'c' (equivalent to state 2) which is a normal state. On the opposite, the occurrence of event 'Ok' before 'Oik' implies a transition to state 'd' (equivalent to state 5). It is a faulty state and the diagnoser identified in this state the fault of 'DCi'.

Figure 9 shows the synchronous product of the diagnoser of 'DC1" with the diagnoser of 'DC2'. State 5 also noted <a,d> corresponds to the localization of a fault of 'DC2'. State 6 also noted <d,a> is a localization of a fault of 'DC1'. Consequently, the synchronous product gives a diagnoser of the system composed of 'DC1' and 'DC2'.

B. General diagnoser from the system behavioral model

In this section, we want to build the diagnoser of the general system from its behavioral model given in fig. 7. One applies again the determination technique. For 'DC1' and 'DC2' cards, the result is exactly the same as the diagnoser of' fig.9. For example, the initial state is a meta-state that groups together states 0, 1, 3, and 5 of the model given by fig. 7. It is uncertain because some of these states are faulty. The rest of the diagnoser is built in the same way.

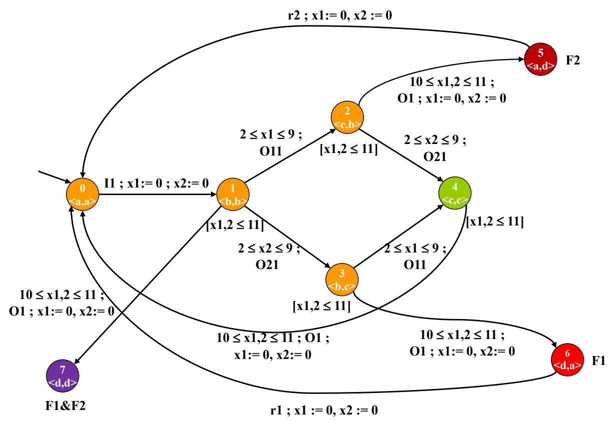


Fig.9. The synchronous product of the diagnosers of two daughter cards

C. Discussion and proposals

Sections V.A and V.b show us that for our example the synchronous product of the diagnosers of 'DC1' and 'DC2' gives exactly the diagnoser of the fault-tolerant control system. As underlined in the introduction of this part, this case is different from classical decentralized or distributed approaches. Indeed each local diagnoser and the general diagnoser used the same observation set. Another difference with classical techniques is the fact that each local diagnoser can locate only the faults of its corresponding 'DCi' daughter card.

Proposal 1: Considering a system A and its sub-systems A_i ($i \in \{1..n\}$) such as :

$$A = A_1 \|_s A_2 \|_s \ldots \|_s A_n$$

Σ is the set of observable events (or actions) of A and of each Ai,

fi is a fault of Ai (this means that Ai is a timed automaton on $\Sigma_{fi} = \Sigma \cup \{fi\}$),

and Θ_i is a (Σ, Δ_i)-diagnoser of Ai with , $\Delta_i \in \Re$

then $\Theta = \Theta_1 \|_s \Theta_2 \|_s \ldots \|_s \Theta_n$ is a (Σ, Δ)-diagnoser of A wih $\Delta = \min(\Delta_i)$ with $i \in \{1..n\}$

Proof: For each faulty timed sequences of events ρ of A, there is fi a faulty event belonging to ρ. Consequently Ai exits as a sub-system of A with fi the faulty event of Ai. Let us call $P_{/\Sigma_{fi}}$ the classical projection from $\Sigma \cup \{f_1, f_2, \ldots, f_n\}$ to Σ_{fi}. Consequently $P_{/\Sigma_{fi}}(\rho)$ is a faulty timed sequence of Ai and $P_{/\Sigma}(\rho)$ is the timed sequence obtained by projection of ρ on the set of observable event Σ (deletion of fi). Because each Θ_i is a (Σ, Δ_i)-diagnoser, $P_{/\Sigma}(\rho)$ is a timed sequence of Θ_i and then of Θ.

As Θ_i is a (Σ, Δ_i)-diagnoser of Ai, there exists a decision function Di defined on TW*(Σ) the set of timed sequences based on alphabet Σ (Di: TW* (Σ) -> {0,1}) such as $D_i(P_{/\Sigma}(\rho)) = 1$. This means that for each faulty timed sequence ρ we can find a decision function D: TW* (Σ) -> {0,1}, such as:

if fi belongs to ρ then $D(P_{/\Sigma}(\rho)) = D_i(P_{/\Sigma}(\rho)) = 1$

This means that Θ is a diagnoser of A.

Let us called $Faulty_{\geq \Delta i}(A_i)$ the set of faulty time sequences on Ai with the minimal time before a fault fi can be diagnosed. If we consider $\Delta_i^{'} \geq \Delta_i$ then $Faulty_{\geq \Delta_i^{'}}(A_i) \subseteq Faulty_{\geq \Delta i}(A_i)$. Consider Δ=min($\Delta_i$) with i ∈{1..n}. According to previous relation and the definition of A, for each i ∈{1..n} $Faulty_{\geq \Delta i}(A_i) \subseteq Faulty_{\geq \Delta}(A)$. Consequently Δ is the minimal time to diagnose each fault fi of A, then Θ is (Σ, Δ)-diagnoser of A.

Proposal 1 generalizes the result given by the example of fault-tolerant systems built from two 'DCi' daughter cards to the case of a system with n sub-system working in parallel. This result has been checked by a simulation of the fault-tolerant system on UPPAAL. It proves that instead of implementing a single general diagnoser of the whole system, it is sufficient to implement a local diagnoser for each sub-system and the n diagnosers evolving in parallel.

VI. Conclusion

In this paper, we have proposed a method to identify the faults of electronic cards in case of fault-tolerant architecture. We have shown that the diagnoser technique can be used in that purpose. The main problem one must solve in that case is the combinatorial explosion in case of attempt to build a single general behavioral model of fault-tolerant system. In this paper a method is proposed that consists in building a diagnoser to survey each card with regard to an output of the system. This method allows identifying the failure of one of the daughter cards and has been extended to a system with n sub-systems evolving in parallel.

In this study we have focused on the use of the diagnoser technique to identify a fault. For implementation, it is important to see that from diagnosers we can extract Causal Temporal Signatures and implement them directly [19]. It is the first perspective of this work. Another perspective is the use of Functional Graph to be able to identify the root cause of any fault [20].

Acknowledgment

This work was supported by Fond Unique Interministeriel and I-TRANS in the framework of FerroCOTS project (Use of COTS in railway systems), from FUI Vague 7 program.

References

[1] R. Johansson, "A fault-tolerant architecture for computer-based railway vehicle brake systems," in *Journal of Rail and Rapid Transit*, vol. 218, Num 3, 2004, pp. 189-20.

[2] M. Sampath, R. Sengupta, S. Lafortune, K.Sinnamohideen, and D. Teneketzis, "Diagnosability of Discrete Event Systems", IEEE Transactions on Automatic Control, vol. 40, n° 9, september 1995.

[3] C. Dousson, P. Gaborit, and M. Ghallab, "Situation Recognition: Representation and Algorithms", International Joint Conference on Artificial Intelligence, IJCAI, , Chambéry, France, August 1993.

[4] R. Alur and D.L. Dill, "A theory of timed automata", Theoretical Computer Science, vol. 126, 1994, pp. 183–235.

[5] F. Lin and W. M. Wonham, "Decentralized control and coordination of discrete-event systems," in Proc. 27th IEEE Conf Decision and Control (Austin, TX), pp. 1125-1130, December 1988.

[6] Shu S., Lin F, Ying H., and Chen X., "State estimation and detectability of probabilistic discrete event systems", Automatica (Journal of IFAC), vol. 44, Issue 12, December 2008, pp. 3054-3060.

[7] L. Feng, "Analysis of temporal performance of supervised discrete event systems", Automatica (Journal of IFAC), Vol. 30, Issue 12, March 1994, pp. 3054-3060.

[8] C.M. Ozveren, "Observability of Descret Event Dynamic system". IEEE Transactions on Automatic Control, vol. 35, n° 7, 1990, pp. 797-806.

[9] C.M. Ozveren and A.S. Willsky, "Invertibility of Descret Event Dynamic system", Mathematics of Control, Signals, and Systems, vol. 5, n°4, 1992, pp. 365-390.

[10] M. Sampath, R. Sengupta, S. Lafortune, K.Sinnamohideen and D. Teneketzis, "Failure Diagnosis Using Discrete-Event Models", IEEE Transactions on Control System technology, vol. 4, n° 2, 1996, pp. 105-124.

[11] M. Fanti and C. Seatzu, "Fault diagnosis and identification of discrete event systems using Petri nets", Proceedings of the 9th International Workshop on Discrete Event Systems, Göteborg, Sweden, May 2008.

[12] D.V. Mcdermott, "A temporal logic for Reasoning about Processes and Plans", Cognitive Science, vol. 6, 1982, pp. 101-155.

[13] A. Toguyéni, E. Craye, and J.C. Gentina, "Time and reasoning for on-line diagnosis of failures in flexible manufacturing systems", in Proceedings of the 15th IMACS world congress on scientific computation, modeling, and applied mathematics, vol. 6. Berlin, Germany, 1995, pp. 709-714.

[14] C. Dousson, P. Gaborit and M. Ghallab, "Situation Recognition: Representation and Algorithms", In Proceedings of IJCAI-93, Chambéry, 1993, pp. 166-172.

[15] C. Dousson, Suivi d'évolution et reconnaissance de chroniques, Phd these , Paul Sabatier' University of Toulouse, 1994.

[16] J. De Kleer, B.C. Williams, "Diagnosing Multiple Faults", Artificial Intelligence, vol. 32 (1), 1987, pp. 97-130.

[17] R. Alur, L Fix, and T.A. Henzinger, "A Determinizable Class of Timed Automata", Proceedings of the *6th International Conference on Computer Aided Verification(CAV'94)*, vol. 818 of Lecture Notes in Computer Science, Springer Verlag, 1994, pp. 1-13.

[18] S. Tripakis, « Folk theorems on the determinization and minimization of timed automata », *Information Processing Letters*, vol. 99 (6), Elsevier, North-Holland, Inc., 2006, pp. 222-226.

[19] R. Saddem, A.K.A Toguyeni, and M. Tagina, "Consistency's checking of chronicles' set Using Time Petri Nets", in *18th Mediterranean Conference on Control & Automation*, Marrakech, Morocco, 23-25 June 2010, pp. 1520–1525.

[20] R. Saddem, A.K.A Toguyeni, and M. Tagina, "From a Functional Graph to a T-Timed Petri Nets Model for the diagnosis of Complex System", in *The 2011 International Conference on Communications, Computing and Control Applications*, Hammamet, Tunisia, 3-5 March 2011.