Implementation of a fault diagnosis method for timed discrete-event systems

Bérangère Suiphon, Zineb Simeu-Abazi, E. Gascard

International Conference
on Industrial Engineering and Systems Management
IESM'2013
May 20 - May 22
RABAT - MOROCCO

Implementation of a fault diagnosis method for timed discrete-event systems ⋆

Berangere SUIPHON [a], Zineb SIMEU-ABAZI [a], Eric GASCARD [b],

[a] *G-SCOP laboratory, 46 avenue Felix Viallet 38031 Grenoble Cedex, France*

[b] *TIMA laboratory, 46 avenue Felix Viallet 38031 Grenoble Cedex, France*

Abstract

To date, fault diagnosis is more and more studied due to its essential role in equipment availability and human safety. This paper proposes a model-based method for event-discrete systems, using the characteristic times in its normal behavior. Indeed, a failure can be detected is these times are not respected. Then the failure is isolate using fault signature analysis. A failure can be distinguished from each other by a set of variables, called diagnostic variables. The choice of these particular variables is essential to diagnose efficiently a system. The diagnosis performance can also be quantified through two parameters : the detection and isolation times.

Key words: fault diagnosis, detection, isolation, characteristic times, timed event-discrete system, timed automaton

1 Introduction

Fault diagnosis plays an important role in equipment availability. In fact, it enables to improve performances, but also to minimize harmful consequences which can be catastrophic for equipment and human safety. This explains why industry devotes a significant part of its work to fault diagnosis. Industrial systems are more and more intricate and consequently more and more difficult to diagnose.

To date, many diagnostic methods have been implemented. They can be divided into two main types : *model-free* and *model-based* methods. Model-free method consists in analyzing monitoring variables to diagnose the system. In model-based method, a model of the system is established. This model includes the knowledge of normal and faulty system behavior. In this approach, the diagnosis can be decomposed into two steps : *detection and isolation*.

In the case of discrete-event systems (DES), the most used approach is the model-based method. The diagnosis is often based on logical models, like Petri Nets. In [3], inputs and outputs are monitored in order to detect and isolate the source of the fault.
But quickly, it appeared that DES models are sometimes insufficient to diagnose all faults. Indeed, the order of the system steps is sometimes not affected by a fault, but the step times are. In this case, it is necessary to

⋆ This paper was not presented at any other revue. Corresponding author B. Suiphon Tel. +33000000000
Email addresses: berangere.suiphon@g-scop.grenoble-inp.fr (Berangere SUIPHON), zineb.simeu-abazi@g-scop.grenoble-inp.fr (Zineb SIMEU-ABAZI), eric.gascard@imag.fr (Eric GASCARD).

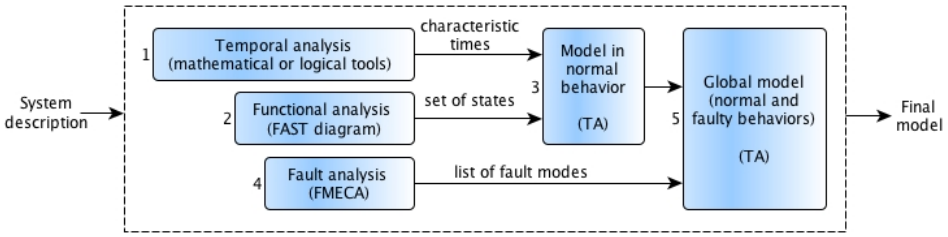

Fig. 1. Steps for modeling a system as a Timed-Automaton

add a new variable : *time*. That is why the recent diagnostic methods are based on timed Petri nets, stochastic automata [7], timed automata [4], template languages [5] or Semi-Markov processes.

In this paper, a model-based method is presented for timed discrete-event systems. This approach is based on using characteristic times of the system. After analysis and modeling of the system, a diagnostic automaton (or *diagnoser*) is build. The model and the diagnoser are Timed Automata (TA).
In this first part, the subject and the context of the paper have been introduced. The second part deals with the tools used to analyze and model a system. In a third part, the necessary steps for building the diagnoser are detailed. The results and validation of our method are exposed in the fourth part. A conclusion with different perspectives of this work is given in the end of the paper.

2 Analysis and modeling tools

The approach described in this paper is a model-based method. It means that the fault diagnosis goes by the differences between the real system and its model. In this part, we expose the successive steps to model the system (Figure 1).
The first step is to analyze the system, first by a *temporal analysis* (1) in order to estimate the characteristic times of the system, and second by a *functional analysis* (2) using FAST diagram. It allows to model the system in its *normal behavior* (3). Then, to consider the fault effects, the *fault analysis* (4) is made, using the FMECA. Finally, the *global model* (5) of the system is build.

2.1 Temporal analysis : search of characteristic times

The diagnostic method detailed in this paper is based on characteristic times of the system. They correspond often to the stage starts (Figure 2). If these times aren't respected, a fault is detected. This subsection explains how to collect these particular moments.

The first step consists in establishing the system dynamics. The objective is to know how the system behaves in the course of time. Many methods can be used : mathematical tools, like differential equations, or symbolic approaches, for example timed Petri nets or timed Markov processes. Once the method chosen, the dynamics is simulated. The simulation allows to collect the characteristic times.
To make it easier to understand, you can refer to the following example.This instance will be used in the whole article.

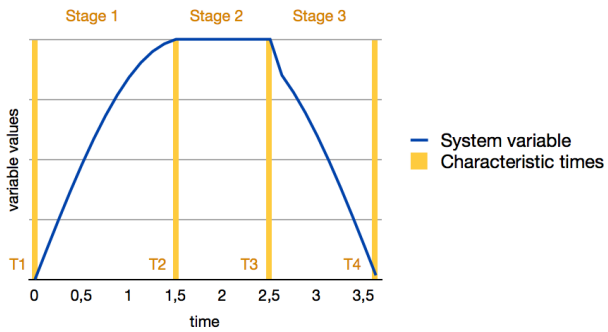

Fig. 2. Example of characteristic times (T1 to T4)

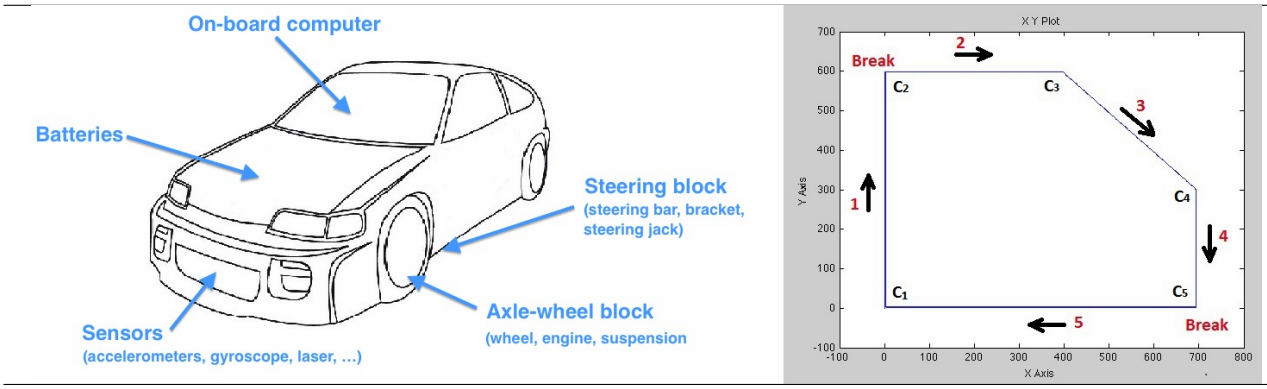

Fig. 3. General diagram of an autonomous electric vehicle and its route

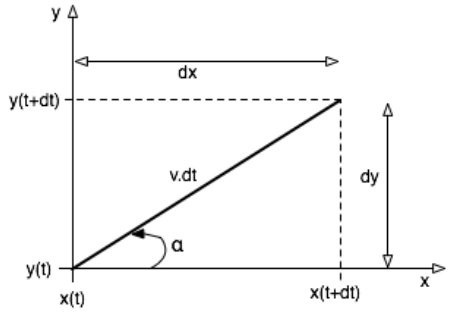

Fig. 4. Angle and speed in Cartesian coordinates

Example : An autonomous electric vehicle
We consider an electric vehicle (Figure 3) which moves without operator, but guiding by variables (white lines on the ground, internal programming of a route, ...). An on-board computer gives instructions to the actuators (engines) according to the sensor values (accelerometer, gyroscope, laser, ...) and to the program it contains.
In this example, the system route (on the right in Figure 3) is made up of five stages and two breaks (50 s). Five sensors (C1 to C5), placed on the route, enable to know if the vehicle is present in the end of a stage (1 if present, 0 else). The vehicle moves at a constant speed (5m/s).
The interesting variables are the speed and the direction angle of the vehicle. In Cartesian coordinates (Figure 4), dynamics correspond to the following equations :

$$\begin{cases} x(t) = \int_0^t \cos(\alpha).v.d\tau \\ y(t) = \int_0^t \sin(\alpha).v.d\tau \end{cases}$$

By simulating the dynamic equations, it is possible to have the characteristic times (Figure 5).

2.2 Functional analysis : using a FAST diagram

The functional analysis is used to determine the main functions, as well as the secondary ones. It allows to know exactly what the system have to do (and how). Among the mass of functional analysis tools, we have chosen to explain the FAST diagram (Figure 6). It consists in breaking a system down into its main functions. Then each function can be broken down into basic functions. For each one, the building solution is specified.
This method is often used to described complex systems as exhaustive as possible.

Example : An autonomous electric vehicle
In the case of the autonomous electric vehicle, the analysis will be made for a quarter of the system. In fact, the vehicle can be decomposed into four identical parts, each made up of a axle-xheel block (wheel, suspension, engine), a steering block (steering jack, steering bar, bracket) and a power block (batteries).
The FAST diagram of the subsystem is shown on the right in the figure 6. For each component, it is now easy to know its role and the associated function.

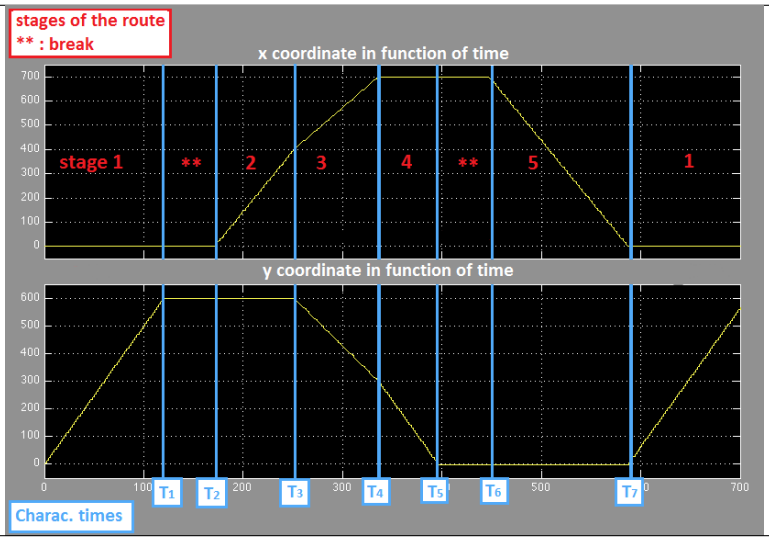

Fig. 5. Characteristic times of the vehicle on the route

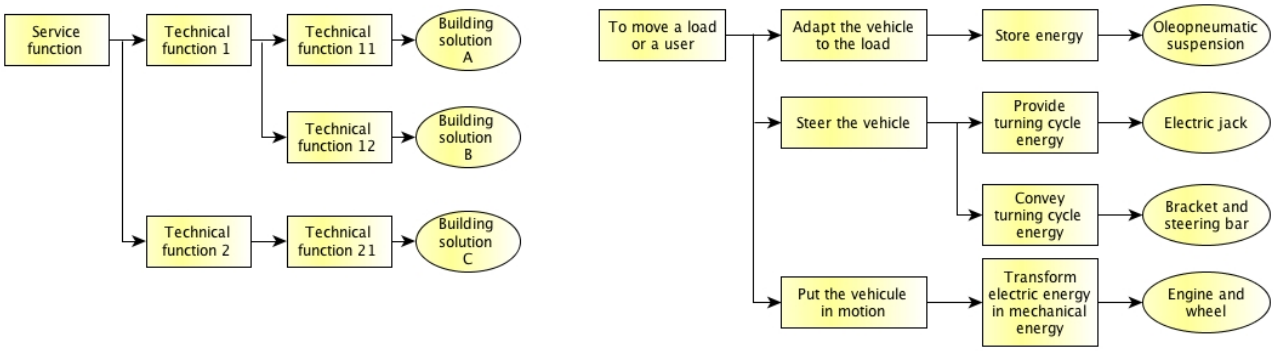

Fig. 6. General FAST diagram (on the left) and for the autonomous electric vehicle (on the right)

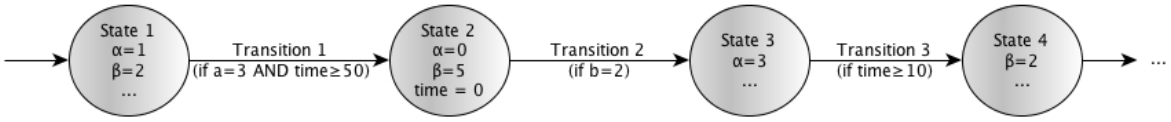

Fig. 7. A simple example of TA

2.3 Modeling of the normal behavior

Thanks to the temporal and the functional analysis, the system is well-known enough to build the model of the normal behavior. We have chosen to model the system as a TA.
A timed-automaton is a finite-state machine extended with a set of clocks. In this way, the automaton states change depending on variable value and on time, that is to say that the transition from a state to another can be a comparison between a sensor and a desired value and/or a comparison between the clock and an integer. Besides, the clocks can be reset during the run of the automaton.
A simple example is given in figure 7 with four states and three transitions and where α, β are control variables and a, b sensors values.

In order to model the system in its normal behavior, we need to represent the operative and control parts (Figure 8).
The operative part, or process, is composed of the dynamics (studied previously in 2.1) and the state of the sensors, depending on the system variables. In the "Sensors" block, the sensors values are changed according to system values.
The control part gives the order to the operative part in function. The order depends on the state of the system. The transition between states depends on the sensor values and the time it occurs. Therefore, this "Control" block has to be a TA.

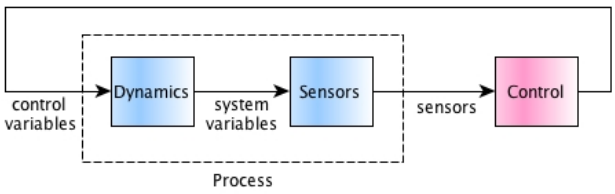

Fig. 8. Modeling of the normal behavior using a TA

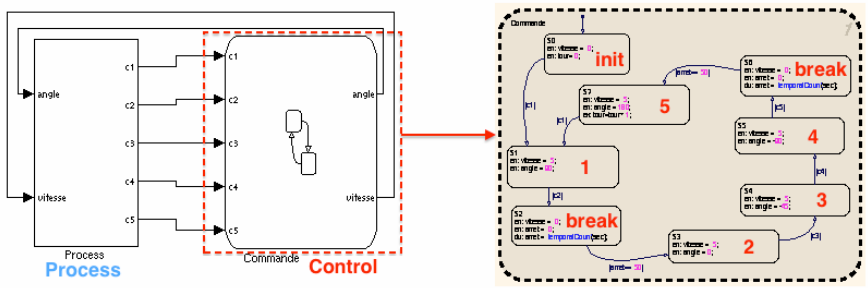

Fig. 9. Modeling of the vehicle in its normal behavior in Matlab *(left : entire model - right : control part)*

We propose in the following example a modeling using Matlab. Matlab is a development environment, often used for numerical calculation, and which has two useful libraries for automation : Simulink and Stateflow. Thanks to them, it is possible to model finite-state machine and TA.

Example : An autonomous electric vehicle
In the case of the autonomous electric vehicle, the control variables are the speed and the angle. From this control values, the dynamics calculates the coordinates of the vehicle (system variables). Then the sensors on the route (C1 to C5) are updated. If the vehicle is present in the same coordinates that a sensor, its value is "1", else it is "0". The process is modeled using Matlab-Simulink (on the left in Figure 9). The "Process" block contains two Matlab programs for dynamics and sensor changes. For more informations, see [6].
As for the control part, a Stateflow chart is build (on the right in Figure 9). It has an initialization state and then the seven states of the route (five movements and two breaks). The transition depends on the value sensors (to finish the first state and get in the second, the sensor C2 has to be "up", ...) and on time (a break is finished if the break lasts 50s). In each state, the control values for speed and angle are defined.
If a simulation of the model is started up, the displayed vehicle behavior is the one in Figure 5.

2.4 Fault analysis : the FMECA approach

Now take an interest in the failures and their modelings. In order to model the behavior of a system in a fault state, the faults have to be listed and their consequences and causes clarified. The FMECA (Failure Mode, Effects and Criticality Analysis) is a powerful tool to describe and analyze the faulty behavior. It appears as a table (Table 1). This is a Component-FMECA : it means that the faults are listed for each component. For each fault, the FMECA gives the faulty component, the altered functions (using the FAST diagram), the effects on the system (locally and on the global behavior) and its probable causes. Then the Criticality level (C) is determined as the product of the occurrence (Occ), the severity (Sev) and the detectability (Det) :

$$C = Occ. \times Sev. \times Det.$$

A grading scale is provided for occurrence, severity and detectability. These ranking scales are ranging from 1 to 5 : for instance, if the occurrence is 1/5, that means that there is a few risk that it occurs ; if severity is 5/5, the consequences on the system are very serious ; if detectability is 1/5, it is easy to detect the failure. Therefore the higher the criticality, the more critical the fault is for the system.
It is possible to add columns to the FMECA in order to specify the maintenance operations, the prevention tasks to avoid having the fault or the tests to identify the failure.

Example : An autonomous electric vehicle
In the case of the autonomous vehicle system, there are 18 failures related to the components of the subsystem and 10 related to the add of the five sensors. The table 2 shows some rows of the FMECA.

Table 1
Structure of a FMECA

Failure Mode Id.	Item	Failure Mode	Function	Causes	Effects	Occ.	Sev.	Det.	Criticality
1	Component block	Failure of the component	Altered function(s)	Causes (extern aggression, intern problem, ...)	Effects (in the block, in the sub-system, in the global system)	*/5	*/5	*/5	(product)

Table 2
Some rows of the FMECA of the autonomous electric vehicle

Failure Mode Id.	Item	Failure Mode	Function	Causes	Effects	Occ.	Sev.	Det.	Criticality
1	Axle-Wheel block	Damaged wheel	Transform electric energy in mechanical energy	nails, pebble, tire wear, ...	-Transmitted mechanical energy does not correspond to request -Altered wheel motion -Altered vehicle direction -Reduced speed -Important energetic consumption	5	3	1	15
6	Axle-Wheel block	Out-of-order suspension	Store energy	Defective suspension jack, dfaillant, broken suspension jack element	No store of the load energy	3	5	2	30
11	Steering block	Crooked steering bar	Convey turning cycle energy	Too high pressure on the bar	Unsufficient turning cycle energy	2	4	4	32
17	Power	Empty battery	System power	Forgetting load, manufacturing fault, wear	No power in the system, no motion	5	3	1	15
18	Power	No power	System power	Short circuit	No power in the system, no motion	4	3	3	36

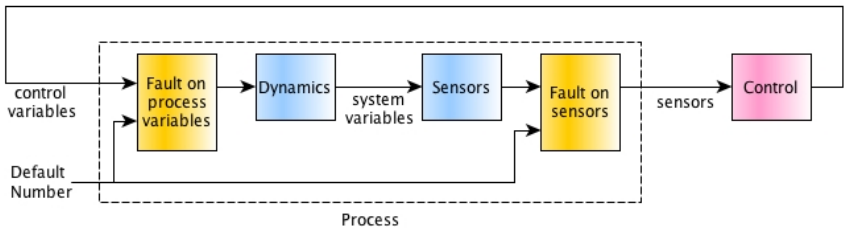

Fig. 10. Modeling of the global system behavior

2.5 Modeling of the global behavior

With the knowledge of the fault behavior (thanks to the FMECA) and the base of the modeling (in the normal behavior), the final stage is to model the global system, that is to say in normal but also in faulty behavior. We are going to consider a single-fault scenario, which means that only one fault can occur on the system.
In order to model the global system, the first step consists in choosing a failure. Then the second step is to consider the consequences on the basic model.

Firstly, the failure has to be chosen. Two methods can be used : a "random-time" method and a "random-failure-and-time" method. The "random-time" method consists in choosing the failure to inject but the time it occurs is random. In the "random-failure-and-time" method, both failure and time are randomly chosen. In this case, the occurrence of a failure depends on its probability (present in component data sheets). For more information, refer to figure 10 ("Fault Choice" block) in the example and [2].

Secondly, the model behavior has to change because of the injection of the fault. When a failure occurs, it can alter sensor values (if it is a fault on a sensor) or control variables. Therefore, the injection of fault in the system alters only the process in the model (Figure 10). If the fault analysis is precise enough about local failure effects, this step of modeling is quite easy.

Example : An autonomous electric vehicle
As said previously, two steps are necessary : failure choice and injection (Figure 11). First at all, for the failure choice, all we have to do is to tell how many failures can happen and their probabilities. Then, it is possible to choose the method for the simulation thanks to the switch.
About the injection of failure effects in the model, the problem is broken down into two parts : effects on sensor values and on control variables. In the Matlab "Fault on sensors" block, each sensor are handled separately (Figure 12). Here, sensors are binary, consequently if a fault occurs on a sensor, it stays either in "down" state or in "up" state ; that is why the programming is made of logical operations.

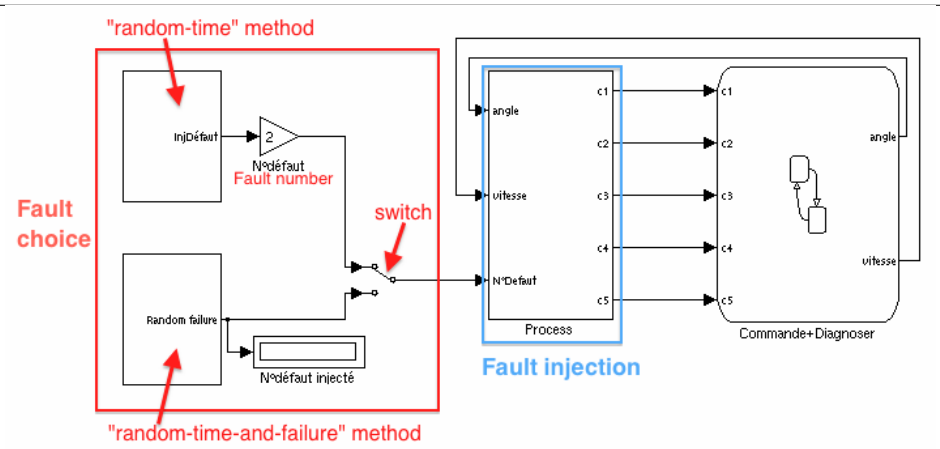

Fig. 11. Fault choice and injection for the autonomous electric vehicle

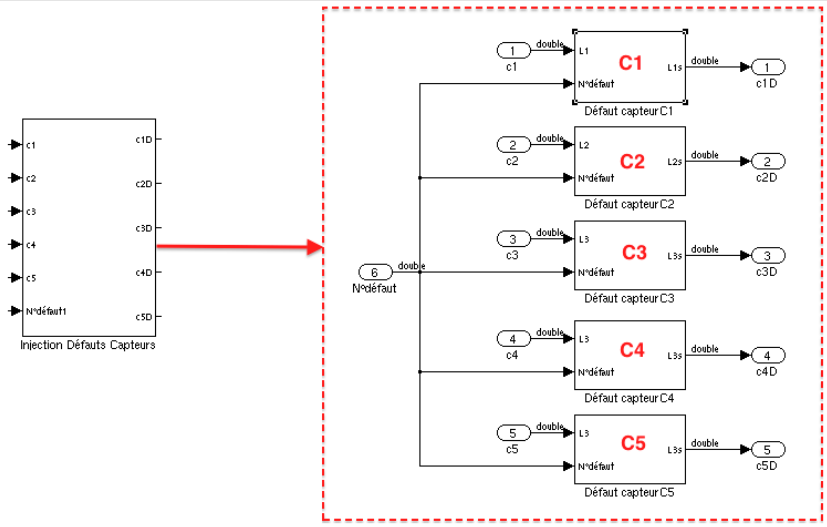

Fig. 12. Failure injection on sensors (the "Fault on sensors" block and its content)

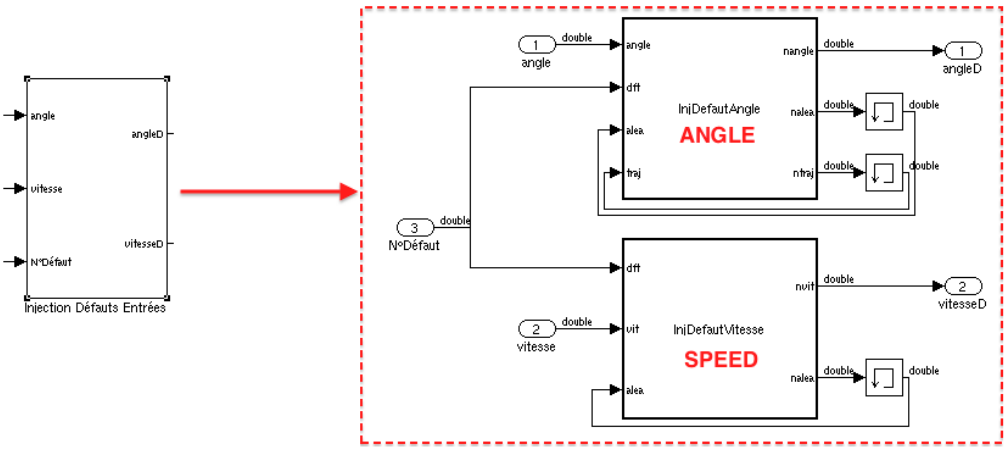

Fig. 13. Failure injection on control variables (the "Fault on process variables" block and its content)

For failures affecting control variables, the "Fault on process variables" block altered separately each variable : here the speed and the angle (Figure 13). We consider that speed have several alteration degrees : low increase, no change, low decrease, high decrease and no speed. In the same manner, the angle can be unchanged (from the normal route), low deflected, high deflected and "straight ahead" (unchanged while it should). In the program, depending on the fault number, the variables have a definite alteration degree ; accordingly they take randomly values on a definite set.

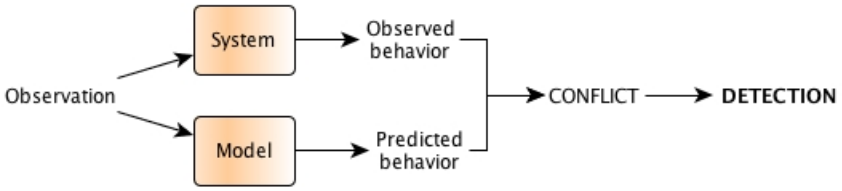

Fig. 14. Failure detection

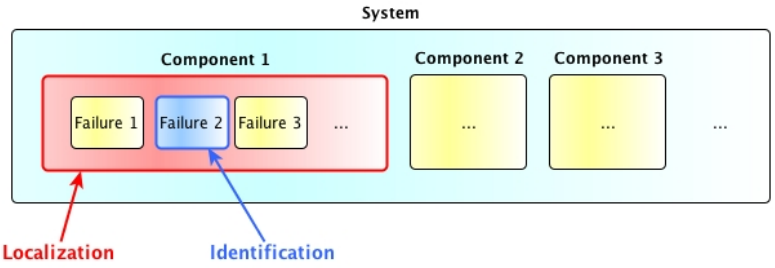

Fig. 15. Failure isolation

Finally, the global model of the system has been built successfully : the system is modeled in its normal behavior but also when a failure occurs. Now we will use this model in order to diagnose failures in the system. With only the system behavior and its characteristic times, the objective is to detect and isolate the fault.

3 Diagnosis by using of signature analysis

In this part, detection and isolation are first explained. Then the notion of fault signature is introduced, which is essential in the diagnostic method in this paper. Finally, the diagnoser building is explained for the example of the autonomous electric vehicle.

3.1 Detection principle

When a failure occurs in the system, the first step is to detect it. It is possible by comparing the system with its model in normal behavior for a same control. If a difference between the two behaviors is noticed, a failure is detected (Figure 14).

3.2 Isolation principle

Once the failure is detected, it need to be isolated (Figure 15). Firstly, the faulty component has to be found : it is the localization. Then a component can be associated with several failures. So all these failures are studied to find the one, which occurs in the system : it is the identification.

3.3 Fault signature analysis

In order to isolate the failure, the method described in this paper uses the notion of fault signature. It considers that a failure can be distinguished from each other by the values of some system variables. The definition of fault signature in this paper is based on [1]. More precisely, a fault affects some variable values in a definite manner. It can be represented as a *diagnostic matrix*.
On figure 16, you can see an example of diagnostic matrices. M_1 is the diagnostic matrix for a variable x_1 which can take three values V_1 to V_3, in a system with five failures d_1 to d_5. If x_1 can take the value V_j in presence of the failure d_i, the matrix element $m_{ij} = 1$ (i^{th} line, j^{th} column), else $m_{ij} = 0$. Then, the signature of the failure d_i corresponds to the i^{th} row of the matrix.

Moreover, it is interesting to use several system variables in order to refined the diagnosis. Indeed, in the example on figure 16, if $x_1 = V_2$, the possible failures are d_1 and d_3. But to distinguish the two failures, it is necessary to add a new variable which does not take the same value for both failures. By adding the variable x_2 and its

$$M_1 = \overbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}}^{V_1\ V_2\ V_3} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix} \qquad M_2 = \overbrace{\begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}}^{W_1\ W_2} \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{matrix}$$

Fig. 16. Example of diagnostic matrices for variables x_1 and x_2

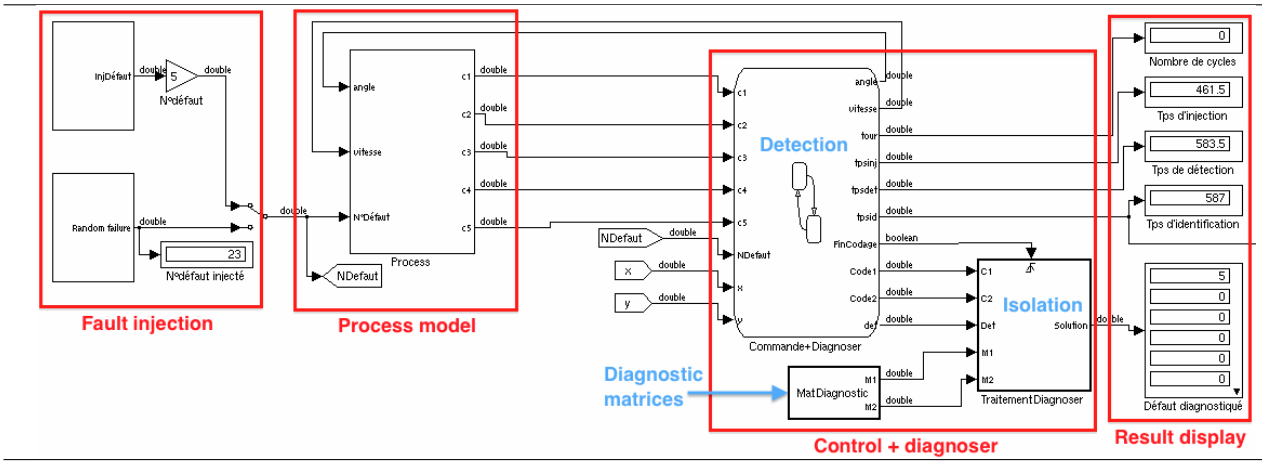

Fig. 17. Matlab interface with the diagnoser

diagnostic matrix, it becomes possible to differentiate between the two failures.
In conclusion, the system variables used for the diagnosis have to be well-chosen in order to isolate correctly all failures.

3.4 Example of diagnoser building

In this subsection, the method to build a diagnoser using fault signatures is introduced. The approach is directly applied to the example of the autonomous electric vehicle, programed with Matlab.
The diagnoser considers time in order to detect failures, consequently the detection has to be a TA (Figure 17). After detection, the failure is isolated thanks to diagnostic matrices and the variable values.

Let's take an interest in the detection (Figure 18). In this figure, the program is broken down into six parts. The first block is the control model of the system. In the second part, the cycle of the system is represented again but if the characteristic times are not respected, the TA leaves the cycle to enter in a detection state. At this moment, the detection occurs.
Once the fault is detected, the system variable values (angle and speed) are calculated (3^{rd} block in figure 18). Indeed, we consider that the system variables used for isolation are unobservable, but can be calculated using observable variables. Once the calculus made, the values are sent out of the TA.
Moreover, the TA contains three blocks to save the injection, detection and isolation times. They will serve to validate the diagnostic method (Section 3).

Now, let's take an interest in the isolation (Figure 17). The isolation block has as entries the diagnostic matrices and the variable values from the TA. For each variable, the column of its diagnostic matrix which corresponds in the calculated value is recovered. By applying a logical "AND" operation between these column vectors, the result of the diagnosis is returned. In the best case, the result column vector contains only one "1" and the failure is isolated. Else, the result is a set of possible failures including the one which occurred.

4 Results and validation

The diagnostic method explained in this paper has been validated. Indeed, for each failure injection, the diagnoser is able to detect and isolate it, as you can see in the table 3.

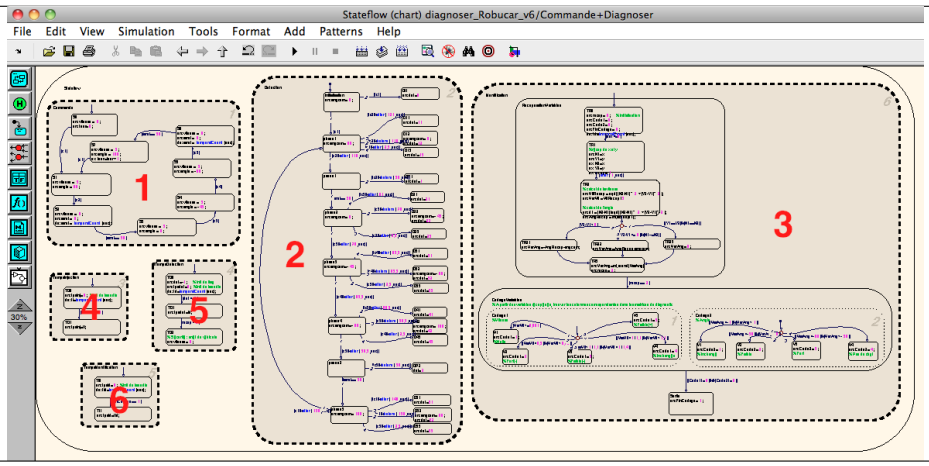

Fig. 18. The detection as a TA
1: control model - 2: detection - 3: variable calculation - 4,5,6: injection, detection and isolation times

Table 3
Results of the diagnosis method : detection and isolation times and diagnoser results for several fault injections

Injected failure number	Injection time	Detection time	Isolation time	Diagnostic time	Diagnoser result (failure number)
1	197.5	250.5	254	56.5	1
15	139.5	250.5	254	114.5	9, 11, 13, 14 or 15

For some failures, the diagnoser is unable to distinguish failures and return a set of failure instead of only one. That means that the variables used for the diagnosis are not enough precise and another one would be necessary to distinguish the failures. However, the set of failures in result contains always the fault which occurred and it targets a precise block of the system : in the case of the 15^{th} failure (i.e. an out-of-order encoder for the steering), all the failures proposed in the diagnoser result belongs to the steering block. The diagnoser allows nevertheless to locate the failure and helps the operate for the maintenance.
About the speed of the diagnostic, in this method, it depends on the injection time. If the injection occurs at the beginning of a new state, the detection can be a bit long : the failure is detected when the state should change in the normal behavior or in the worst case (for the sensor faults) after an entire cycle. However, the isolation is quite fast. It depends essentially on the calculation of variables values.

5 Conclusion

The diagnostic method presented in this paper is interesting, because it can be applied on all timed event-discrete systems easily. Some improvements can nevertheless be brought about : the choice of system variables used for the diagnosis could be studied in order to avoid the uncertainty of the diagnoser. By adding some sensors in precise place on the system, it could be possible to add diagnostic variables and refine the results.
In every case, using the characteristic times of a system is a good idea to diagnose failures. It is an interesting way of work.

References

[1] Sujeevan Aseervatham ASEERVATHAM. Diagnostic à base de modèles intégrant les modèles de fautes. Application : Diagnostic du système de freinage dans un véhicule. MSc dissertation. Technical report, University Paris 13 Villetaneuse, October 2004.

[2] Francis Chalagiraud. Diagnostic des défaillances par automates temporisés. MSc dissertation. Technical report, Polytech Department, Joseph Fourier University, August 2010.

[3] S. Hashtrudi Zad, R.H. Kwong, and W.M. Wonham. Fault Diagnosis in Finite-State Automata and Timed Discrete-Event Systems. In *In 38th IEEE Conference on Decision and Control*, pages 1756–1761, 1999.

[4] I. Hristov, J. Lunze, and P. Supavatanakul. A time discrete-event approach to diagnosis of the damadics actuator benchmark. In *Proceedings of the Fifth DAMADICS Workshop*, 2004.

[5] D. N. Pandalai and L. E. Holloway. Template Languages for Fault Monitoring of Timed Discrete Event Processes. *IEEE Transactions on Automatic Control*, 45(5):868–882, may. 2000.

[6] Bérangère Suiphon. Implémentation d'une méthode de diagnostic des systèmes à événements discrets. MSc dissertation. Technical report, Polytech Department, Joseph Fourier University, August 2011.

[7] P. Supavatanakul, C. Falkenberg, and J. Lunze. Identification of timed discrete-event models for diagnosis. In *14th International Workshop on Principles of Diagnosis*, pages 193–198, 2003.