Decentralized diagnosis and diagnosability by model checking

Philippot Alexandre, Pascale Marangé, François Gellot, Bernard Riera

Universal Journal of Control and Automation 1(2): 28-33, 2013 http://www.hrpub.org
DOI: 10.13189/ujca.2013.010202

Decentralized Diagnosis and Diagnosability by Model Checking

A. Philippot[1,*], P. Marangé[2], F. Gellot[1], R. Riera[1]

[1]Research Centre of STIC (CReSTIC), University of Reims Champagne-Ardenne (URCA), Reims, France
[2]Research Centre in Automation of Nancy (CRAN), University of Lorraine, CNRS, Vandœuvre-lès-Nancy, France
*Corresponding author: alexandre.philippot@univ-reims.fr

Abstract This paper talks about a decentralized approach for diagnosis of discrete event systems based on the plant decomposition. A decentralized structure is used to avoid state space explosion found in centralized structure or decentralized structure with composition step. From plant models, all possible faults are identified to construct abnormal behavior models called diagnosers. Originality of this proposition is also to used model-checking to verify diagnosability of the system. The approach is illustrated using an academic benchmark.

Keywords Discrete Events Systems, Diagnosis, Model Checking, Manufacturing systems

1. Introduction

For a few decades, diagnosing dynamic systems has become an active topic in scientific and industrial research due to complexity increasing but also of costs of maintenance interventions. This complexity and the desire for improved availability, reliability and dependability require the development of systematic approaches of diagnosis to detect and isolate a fault. Various diagnosis approaches have been proposed including fault-trees, expert systems, neural networks, fuzzy logic or Bayesian networks for example. Diagnosing a system means providing candidates of a fault which can explain it by the observations collected during the system operation in a bounded delay. For this, diagnosers are introduced as observers which reconstruct information of the process and help the users in its decisions. One of the major distinctions in diagnosis approaches is whether a Model Based Reasoning is used or not [10]. Modeling the system behavior is often computationally expensive but returns advantages, such as formalization or instantiation of equipment. The models might represent normal or abnormal behavior and can be quantitative (based on equations) or qualitative (based on cause/effect models). Most systems can be viewed or reduced as Discrete-event systems (DESs) [3] where the dynamic is represented thanks to discrete inputs and outputs.

In recent years, literature has been interest in the diagnosability problem for DESs. Several approaches have been developed to solve the Fault Detection and Isolation (FDI) problem. In [15], a centralized approach and a notion of diagnosability have been proposed around a global diagnoser. However, a main disadvantage of centralized approaches for DES is the space states explosion. The decentralized approaches constitute a solution to this drawback. However, designing a decentralized diagnoser requires the existence of centralized model. The decentralized model is designed as a decomposition of a global model which entails again the state explosion problem [16, 17]. Moreover, decentralized approaches often need a coordinator (or supervisor as a high level model) to solve uncertainty or ambiguity of local decisions. In a decentralized diagnosis system, there are multiple local diagnosers, and each of them performs diagnosis based on its own observations without communicating to each other. The global decision is obtained by merging local decisions in order to take into account the dependency relation between the components. In [6], the authors propose an approach to improve the decentralized diagnosis representation based on state and transition independence properties. Distributed diagnosis can be considered as on especial case of decentralized diagnosis. It is composed of a communication protocol between each local diagnoser. A protocol module for each site decides how to share information among various diagnosers. A local diagnoser performs failure diagnosis based on the local observations and the communicated information received from other diagnosers [13].

Most of the diagnosis approaches are modeled by automata or Petri nets. These tools are based on formal languages and permit some manipulations as synchronous compositions or projections (mask). Moreover, this formalization can introduce the various notions of failure diagnosability (according to the structure) in order to prove the ability to detect and diagnose a failure within bounded

delay [13].

In this paper a decentralized diagnosis approach is proposed around components modeling, specifically manufacturing systems with discrete sensors and actuators. The originality of the proposition consists in to avoid global modeling step of the plant G (for example obtained by synchronous composition of local models) and then avoid the space states explosion. Local diagnosers are obtained from local modeling of a class of components, called Part of Plant (PoP), which can be instantiated (Fig. 1). Moreover, we propose an original way to verify the diagnosability by using model checking. Model checking is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for a given state in that model [4]. It is based on Computational Tree logic (CTL) which can be graphically analyzed and be transformed as automata. The verification tools Uppaal is used for this works [2].

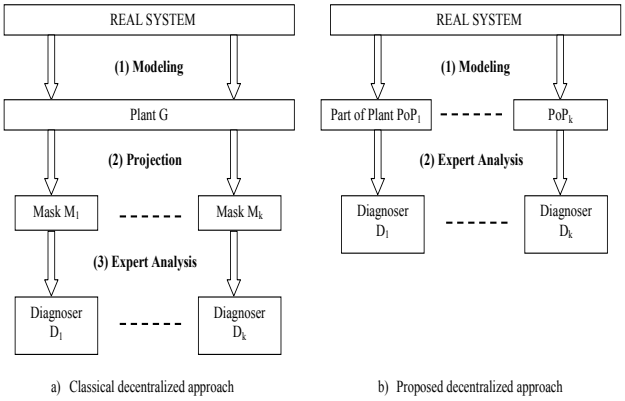

Figure 1. Decentralized approach

The paper is organized as follows. In section 2, the proposed approach is presented through it formalism, the different steps to obtain local diagnosers. The verification of diagnosability by model checking is also introduced. Section 3 illustrates the approach around an academic benchmark. Conclusion and prospects close the paper in section 4.

2. Decentralized Diagnosis

2.1. Plant Modeling

In industrial processes, a manufacturing system is a functional chain composed of a controller that emits signals to a plant and receives sensor values (Fig. 2). Plant represents the mechanical part whereas controller is the logical part which describes the desired behavior. This exchange between controller and plant represents the only observable information available on line. Since a diagnoser is defined as an observer of the system, it is necessary to use this information to rebuild behaviors through models.

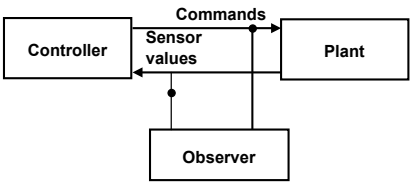

Figure 2. Functional Chain

From the introduction section, a centralized approach appears as unthinkable for large and complex systems. However, the difficulty of a decentralized approach is to determine the level of modular decomposition in a generic way. A manufacturing system is composed of mechanical components (actuators/sensors) which interact or not. Each component can be modeled in a normal behavior as a Part of Plant (PoP). PoP models take into account the technological specifications to obtain realistic models[12]. Consequently, a component i can be modeled by an automaton:

$$PoP_i = (X_i, \Sigma_{oi}, \delta e_i, x_{i0}, I_i) \quad (1)$$

Where X_i is the state space, Σ_{oi} is the observable event set, δe_i is the transition function, x_{i0} is the initial state and $I_i$is the set of interval time where transition functions are expected. The states and the events return only the normal behavior of the component.

A transition function δe_i corresponds to a logical expression composed by expected events from a state. Temporal information centered on the notion of expected event sequencing and timing relationships can be used. This information, instigated from templates of [7], is about events minimal and maximal time occurrence, which represent the actuators minimal and maximal response times. Each interval is constructed for observable correlated events and it describes the next events that should occur and the relative time periods in which they are expected. It is defined by an up and down counter of time.

2.2. Faults Partitions

A local diagnoser is obtained after identification of all possible faults on each component. It is an expert analysis which allows to define the following faults associated to a label of the diagnoser. A subset of faults which can occur on a component is called partition. Then, each fault partition Π_{Fj} is associated with a label Fj indicating the type of failures. Consequently, for N components modeled by N PoPs N diagnosers with a label for each will be defined.

The following two hypotheses are considered:

●Only one failure event responsible of a faulty behavior can occur at the same time ;

●Controller is supposed to be dependable and safety thanks a filter approach [11]. Consequently, the controller cannot be responsible of any fault as the one of sending two opposable control signals.

To determine all possible candidates responsible of an abnormal behavior in a PoP, a knowledge expert is made.

Faults can be modeled as observable or/and unobservable events (Table 1). The case of observable events is a trivial one since faults can be detected as soon as they are observed. In the case of unobservable events, the occurrence of a fault must be inferred from the system model and future observations. A problematic is to identify the real cause of an observation.

Table 1. Possible Partitions Faults

observable sensor fault	Unexpected passage of a sensor value from 0 to 1
	Unexpected passage of a sensor value from 1 to 0
non observable sensor fault	Sensor stuck-off
	Sensor stuck-on
non observable actuator fault	Actuator stuck-off
	Actuator stuck-on

2.3. Faults Partitions

A local diagnoser D_i is based on a PoP_i and is represented by automata:

$$D_i = (X_i \cup XDF_i,\ \Sigma_{io},\ \delta_i,\ x_{i0},\ I_i,\ l_i) \quad (2)$$

With X_i, Σ_{oi}, δe_i, x_{i0} and I_i as defined in eq. (1). XDF_i is the set of abnormal states, δ_i: $X_i \times \Sigma_i^* \rightarrow X_i \cup XDF_i$ is the transition function with the expected (δe_i) and unexpected (δu_i) functions from a state and $l_i$is the set of decision functions of the local diagnoser D_i with $l_i(x)$ the decision function of the state x which can be one or more fault labels {Fj}.

For the local decision, if the label function at a state x is $l_i(x)$ = {N}, then the diagnoser, when it reaches the state x, can decide with certainty the non-presence of faults. If the label function at a state x is $l_i(x)$ = {Fj}, then the diagnoser indicates with certainty the occurrence of a fault of the type F_j. If $l_i(x)$ = {N, Fj}, then the diagnoser cannot decide whether a fault has occurred or not and the system is in ambiguity or indecision case. Labels are defined in partition Π_{Fj}. The knowledge of all faults comes from an expert analysis and/or documentation such as Failure Mode and Effects Analysis (FMEA), a tool used in safety, dependability and quality management [1].

To define the unexpected functions (δu_i) it is possible to define all transition functions by the 2^n possibility (with n: number of events and intervals). However, the mechanical structure of components and the use of control filter make it impossible some combinations. For example, only one interval can be to 1, or thanks to the control filter, opposite orders cannot be sent. Consequently, the complexity depends on the granularity of the local models but also on the performance of the control filter [11].

Concerning the interval time, it is useful to enhance the diagnosability of the system. For each event time occurrence, a tolerance interval corresponding is defined.

2.4. Diagnosability by Model Checker

A formal verification approach is used to ensure the diagnosability of a system using the model-checker UPPAAL. To make this verification, the system and the execution environment must be modeled[14].

In this work, the system is controlled by a (Programmable Logic Controller (PLC) [8]. The model must take into account PLC cyclic and sequential behaviour. Taking into account application domain, and with the intention of minimizing the distance between the model and the real system (PLC + plant), it is necessary to consider the causality delay present in the system model. For that, the system is modeled by:

●Environment model represents the sequential and cyclic behavior of the PLC (Fig. 3). The model of the PLC computing environment is taken into account the evolution of diagnosers.

●Outputs model represents the evolutions which can be sent. This model represents either the most permissive control.

●Diagnosers models with defaults model and intervals models.

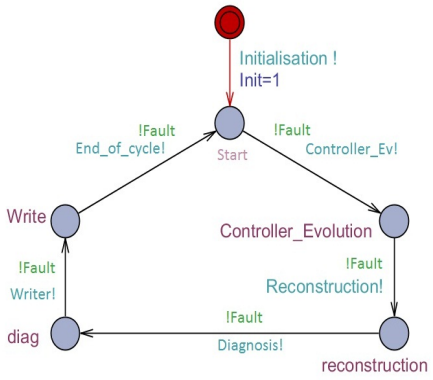

Figure 3. Environment model of a PLC

To verify the diagnosability, it is necessary to check that each faulty state can be attempt with certainty. For that purpose, the logic CTL (Computation Tree Logic) (Clark and Emerson, 1981) is used to define the property. The universal path quantifier A for the All operator ∀, E for the Exist operator ∃, X for neXt operator ○, G for the Globally operator and F for the Finally operator ◊ are use Boolean operators not, && (and), || (or).

Moreover, automata can have infinitely many states or traces of events. Uppaal model-checker cannot visualize all these traces but can show a set of traces. Symbolic traces shown in the simulator of Uppaal can help in the analysis of diagnosability.

3. Illustration

3.1. Presentation of theBenchmark

We propose to illustrate the proposition through this

section and an academic example. This example is the elementary HVAC system from Sampath thesis [15] (Fig. 4).

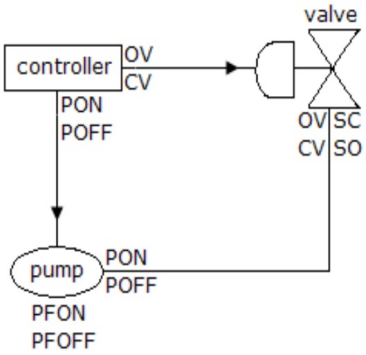

Figure 4. HVAC system from Sampath (1995)

It consists of a pump, a valve and a controller. However, some accommodations and notations are made:

●The pump is not instrumented whereas the valve disposes of two sensorsfor the open position (*fso*) and for the closed position (*fsc*).

●The pump start is activated by *M* Boolean signal.

●The valve closing and opening moves are activated by respectively *C* and *O* Boolean signals.

The valve and sensors fsc and fso constitute 3 components and it is possible to identify each faulty event by a label:

●Sensor *fsc* stuck to 0 (F1) or to 1 (F2)

●Sensor *fso* stuck to 0 (F3) or to 1 (F4)

●Valve stuck to *fsc* (F5) or *fso* (F6) position

●Unexpected *fsc* (F7) or *fso* (F9) from 0 to 1

●Unexpected *fsc* (F8) or *fso* (F10) from 1 to 0

●Unexpected movement from *fsc* to *fso* (F11) or from *fso* to *fsc* (F12)

●Valve blocked between *fsc* and *fso* (F13)

Three fault partitions are defined belong to:

●Sensor *fsc*: $def\_fsc = \Pi_{fsc}$ = {F1, F2, F7, F8}

●Sensor *fso*: $def\_fso = \Pi_{fso}$ = {F3, F4, F9, F10}

●Valve: $def\_V = \Pi_{Va}$ = {F5, F6, F11, F12, F13}

To define the interval time where transition functions are expected, when the Open signal is sent to the valve, a consequence is to receive the deactivation of the sensor fsc in interval I1 or I2 but also the activation of the sensor fso in an interval I4. These pre-defined time periods are determined by experts according to the system dynamic and to the desired behavior (Fig. 5).

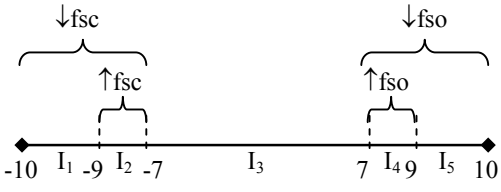

Figure 5. Intervals time of the valve

3.2. Local Diagnosers

The assumption is kept that only one fault can occur at the same time. For each state of a component, an expert must analyze the possibility of fault occurrence and especially the possibility to detect and isolate a fault. It returns a model with faulty states labeled. The diagnoser is initialized from a normal state according to the first sensors' observation. Local diagnoser of the valve sensor *fsc* is presented in Fig. 6. From the initial state, diagnoser looks if the sensor is to 0 (state x_1) or to 1 (state x_2). From x_1, it is possible to attempt x_2 if the closing signal *C* is sent and that sensor *fsc* is to 1 interval I2. Contrary, the opening signal *O* when the sensor is to 0 in interval I1 or I2 allows to go from x_2 to x_1.

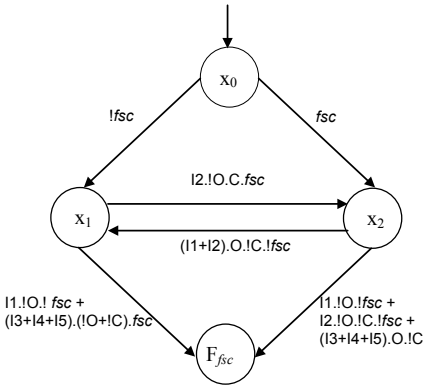

Figure 6. Diagnoser of the valve sensor *fsc*

Analyze of all possible transition functions from x_1 and consequently all forbidden transition functions can be made. After simplification, it enables to give a transition function to attempt the faulty state F_{fsc}. This function defines that it is in interval I1, sensor *fsc* is again to 0 whereas there is none order sent, consequently the sensor is considerate has stuck-off to 0 (F1). Another possibility is to have the unexpected passage of sensor *fsc* from 0 to 1 (F7). The same analysis can be made from x_2 to identify F2 and F8.

Diagnoser of the valve sensor *fso*is similar with reverse parameters. Considering the valve actuator, the local diagnoser is represented in Fig. 7.

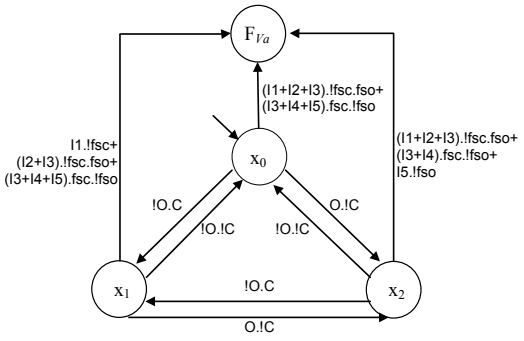

Figure 7. Diagnoser of the valve actuator

3.3. Diagnosability Verification

Environment model of fig. 3 represents the sequential and

cyclic behavior of the PLC and start it synchronization by the evolution of the controller if the guard is with no fault. It is the Outputs model of fig. 8 which represents the most permissive control which can be sent on the valve with the control filter consideration.

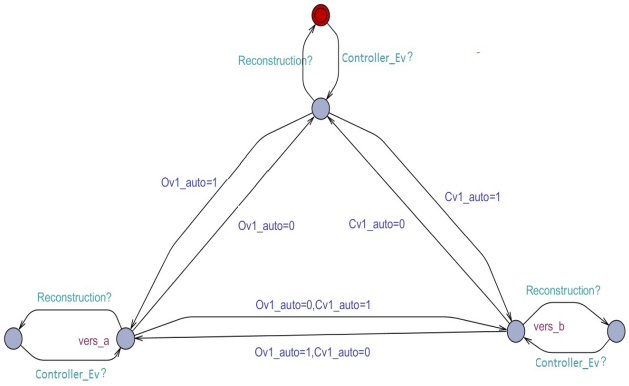

Figure 8. Outputs model of the valve

The Reconstruction information consists in calculating falling or rising edges of signals and also in refresh parameters as the interval (Fig. 9) where the function *cal_inter()* returns the activated interval (I1, I2, I3, I4 or I5) according to the value of an evolving parameter *V1_cpt*. The Reconfiguration information allows also to generate anytime faulty events.

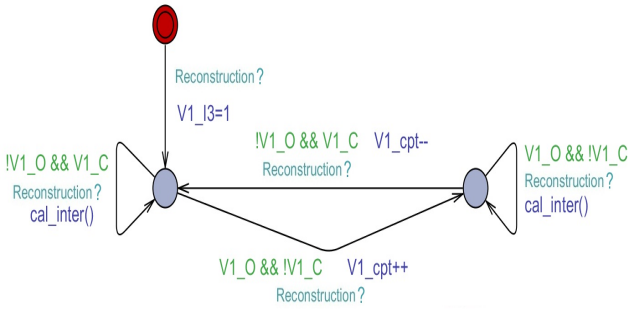

Figure 9. Intervals calculation

Diagnosers models observe the behavior and indicate or not a fault. If the system is considerate as normal, the Environment model authorized the evolution in a writing model of an output signal. Otherwise, all output signals are deactivated. A new cyclic evolution of the PLC can be represented.

For the case study, the satisfied property verified is: "that it is possible to attempt a unique faulty state with certainty". This property is defined in CTL logic for each partition by the equations (1) for the valve, (2) for *fsc* sensor and (3) for *fso* sensor:

$$\text{AG (not (}def\_V \text{\&\& not } def\_fsc \text{\&\& not } def\_fso\text{))} \quad (3)$$

$$\text{AG (not (not } def\_V \text{\&\&} def\_fsc \text{\&\& not } def\_fso\text{))} \quad (4)$$

$$\text{AG (not (not } def\_V \text{\&\& not } def\_fsc \text{\&\&} def\_fso\text{))} \quad (5)$$

These properties are verified with using the Uppaal tool model-checker.

3.3. Discussion

The first analysis has shown that the system is detectable in a bounded delay with certainty for the defined fault partitions. However, it does not satisfy any equations (1), (2), (3) and consequently, the system were defined as non-diagnosable with certainty. Thanks to symbolic traces, we have seen that the only cases where the equations are not verified were when several faults are generated. After extraction of these traces, the properties have shown that it is possible to isolate faults between sensors. However, some cases an ambiguity is given between a sensor and the valve actuator. For example, it is not possible to isolate with certainty states with {F1, F13} (sensor *fsc* stuck to 0 and valve blocked between *fsc* and *fso*) and {F3, F13} (with sensor *fso* stuck to 0). Consequently, another rule must be present to guarantee complete diagnosability notion (Lin, 1994).

For comparison, a second analysis has been made concerning a global diagnoser of the valve with the 2 sensors (Fig. 10). It results the same conclusion as the decentralized proposition but with a diagnoser more complicate to establish and to explain. This global diagnoser is more complex and is composed of 9 normal states and 18 abnormal states (14 diagnosable states with certainty in yellow and 4 non-diagnosable in red).

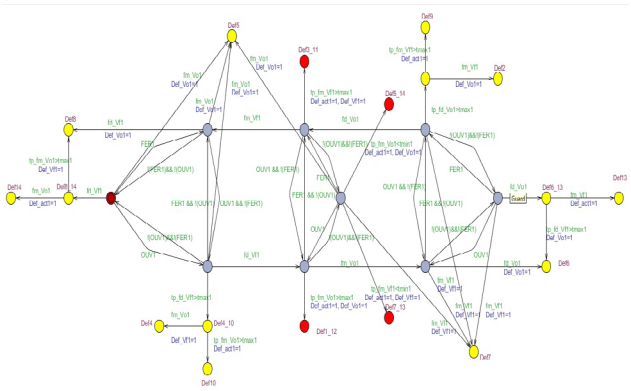

Figure 10. Global Diagnoser of the valve with sensors

The last point of comparison concerns the initial proposition in [15]. In these works, authors have proposed to obtain a global diagnoser from the synchronous composition of the component models for the HVAC system but also with the controller model. Firstly, the composition step makes it impossible for large and complex system (even with classical decentralized approach where the composition step comes before the projection step) (Fig. 11). Secondly, it is dependent of the controller. Consequently, it is not reusable and if your control changes, you must rebuild all your diagnoser. In this proposition, none composition step is made and the most permissive control is considerate.

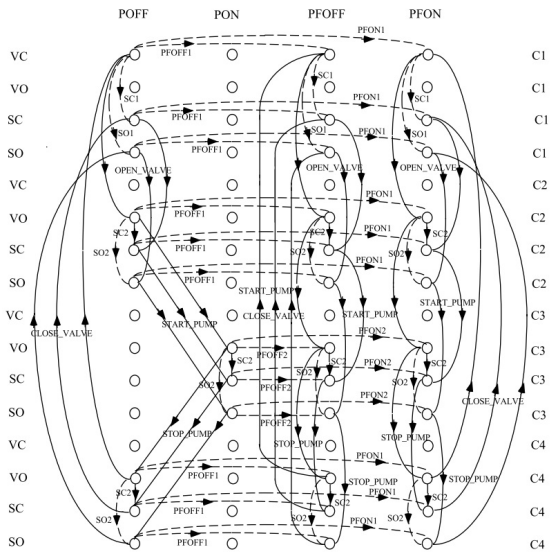

Figure 11. Synchronous composition of Sampath approach

4. Conclusion

The paper presents an approach for diagnosis of discrete manufacturing systems based on the plant decomposition. A decentralized structure is used to diminish combinatory explosion found in centralized structure. From plant models, all possible faults are identified to construct abnormal behavior models called diagnosers. Another originality is to used model-checking approach to verify diagnosability of the system. The approach is illustrated using an academic benchmark.

In literature, most of methods defines specific faults partition and does not considerate all possibility. Moreover, these approaches are not always relevant to real systems. However, in this paper, only the faults related to components (actuators and sensors) are considered. To take into account the product faults, a product model is necessary. This model depends on the product nature and on the production objective. Thus, a future work is to extend this approach to include the faults related to product.

Another prospect is to use diagnosers with filter approach to be dependable regardless the controller and the plant in the aim to reconfigure automatically a system and help the user in it monitoring task.

REFERENCES

[1] Ashley, S.. Failure analysis beats Murphy's law, Mechanical engineering, ISSN 0025-6501, vol. 115, no9, pp. 70-72, 1993.

[2] Behrmann, G., David, A. and Larsen, K. G.. A tutorial on Uppaal. In Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, Vol. 3185 of Lecture Notes in Computer Science, pp. 200–236. Springer-Verlag, 2004.

[3] Cassandra, C.G. and Lafortune, S.. Introduction to Discrete Event Systems. Kluwer Academic Publisher, ISBN 0792386094, 1999.

[4] Christel, B., Joost-Pieter, K.. Principles of Model Checking. The MIT Press, pp.11, 2008.

[5] Clarke E. M., Emerson E. A.. Design and synthesis of synchronization skeletons using branching time temporal logic. In Proc. Logics of Programs Workshop, Vol. 131 of Lect. Notes in Comp. Sci., pp. 52-71. Springer, 1981.

[6] Cordier, M.O., Le Guillou, X. Robin, S., Rozé, L. and Vidal, T.. Distributed Chronicles for On-line Diagnosis of Web Services. 18th International Worshop On Principles of Diagnosis (DX'07), Nashville, USA, 2007.

[7] Holloway, L.E. and Chand, S.. Time templates for discrete event fault monitoring in manufacturing systems. American Control Conference, Baltimore, USA, 1994.

[8] IEC 61131-3. International Electrotechnical Commission, PLCs – Part 3: programming languages. Publication 611131-3, 1993.

[9] Lin, F.. Diagnosability of Discrete Event Systems and its Applications. Discrete Event Dynamic Systems, Kluwer Academic Publishers, USA, 1994.

[10] Magnani, L.; Nersessian, N.J.. Model Based Reasoning. Science, Technology, Values, ISBN 978-0-306-47244-2, 2002.

[11] Marangé P., Gellot F., Riera B.. Remote control of automation systems for D.E.S. course, revue IEEE Transaction on Industrial Electronics Special Section, pp 3103-3111, 2007.

[12] Philippot A., Sayed Mouchaweh M. Carré-Ménétrier. V. "Chapter 16: Component models based approach for failure diagnosis of Discrete Event Systems", Intelligent Industrial Systems: Modelling, Automation and Adaptive Behaviour, IGI, 2010.

[13] Qiu, W.. Decentralized/distributed failure diagnosis and supervisory control of discrete event systems. Thesis, Iowa State University, USA, 2005.

[14] Riera B., Benlorhfar R., Annebicque D., Gellot F. , Vigario B.,. Robust control filter for manufacturing systems : application to PLC training, 18th World Congress of the International Federation of Automatic Control , Milano, Italy, 2011.

[15] Sampath, M.. A Discrete Event Systems Approach to Failure Diagnosis. Thesis, University of Michigan, Michigan, USA, 1995.

[16] Su, R. and Wonham, W.M.. Wonham. Decentralized fault diagnosis for discrete-event systems. CISS, Princeton, New Jersey, USA, 2000.

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