Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A fault tolerant architecture for supervisory control of discrete event systems [1]

Andrea Paoli [*,2] **Matteo Sartini** [*] **Stéphane Lafortune** [**]

[*] *Center for Research on Complex Automated Systems (CASY), DEIS, University of Bologna, Viale Pepoli, 3/2 - 40136 Bologna, ITALY,* {andrea.paoli ; matteo.sartini}@unibo.it
[**] *Department of Electrical Engineering and Computer Science, The University of Michigan, 1301 Beal Avenue, Ann Arbor, MI 48109-2122, USA,* stephane@eecs.umich.edu

Abstract: In this paper the problem of Fault Tolerant Control (FTC) in the framework of Discrete Event Systems (DES) modeled as automata is considered. The approach we follow is the so-called active approach in which the supervisor actively reacts to the detection of a malfunctioning component in order to eventually meet degraded control specifications. Starting from an appropriate model of the system, we recall the notion of safe diagnosability as a necessary step in order to achieve fault tolerant supervision of DES. We then introduce two new notions: (i) "safe controllability", which represents the capability, after the occurrence of a fault, of steering the system away from forbidden zones and (ii) "active fault tolerant system", which is the property of safely continuing operation after faults. We show how it is possible to define a general control architecture to deal with the FTC problem by introducing a special kind of automaton, called a "diagnosing-controller". *Copyright ©2008 IFAC.*

Keywords: Fault tolerant control; Fault diagnosis; Discrete event systems; Automata; Supervisory control theory; Safety.

1. INTRODUCTION

Complex technological systems are vulnerable to unpredictable events that can cause undesired reactions and as a consequence damage to technical parts of the plant, to personnel or to the environment. The main objective of the Fault Detection and Isolation (FDI) research area is to study methodologies for identifying and exactly characterizing possible incipient faults arising in predetermined parts of the plant. This is usually achieved by designing a dynamical system which, by processing input/output data, is able to detect the presence of an incipient fault and eventually to precisely isolate it. Automated systems are typically governed by operational rules that can be modeled by Discrete Event Systems (DES), i.e., dynamical systems with discrete state spaces and event-driven transitions. Several methodologies have been developed to solve the FDI problem for systems modeled as DES; see Sampath et al. [1995], Boel and van Schuppen [2002], Hadjicostis [2005], Jiang and Kumar [2006] for a sample of this work.

Once a fault has been detected and isolated, the next natural step is to reconfigure the control law in order to tolerate the fault, namely to guarantee pre-specified (eventually degraded) performance objectives for the faulty system. In this framework, the FDI phase is usually followed by the design of a Fault Tolerant Control (FTC) system, namely, by the design of a reconfiguring unit that, on the basis of the information provided by the FDI filter, adjusts the controller in order to achieve the prescribed performance for the faulty system (see Blanke et al. [2003]). The *passive approach* to the FTC problem deals with the problem of finding a general controller able to satisfy control specifications both in nominal operation and after the occurrence of a fault. In contrast, *active fault tolerance* aims to achieve the control objectives by adapting the control law to the faulty system behavior. The problem of FTC in DES has been recently studied for example in Iordache and Antsaklis [2004] and Dumitrescu et al. [2007]. In Wen et al. [2007b] and Wen et al. [2007a], the authors propose a definition of fault tolerance based on the notions of state equivalence and stability of DES and they provide a necessary and sufficient condition for the existence of a supervisor able to enforce a specification for the nonfaulty plant and another specification for the overall plant.

In this paper we consider the FTC problem in the framework of DES modeled as automata. The approach we follow is the active approach in which the supervisor actively reacts to the detection of a malfunctioning component in order to meet eventually degraded control specifications. To this aim we describe a modeling procedure that results in a structured model of the controlled system containing a nominal part and a set of faulty parts. Starting from this suitable model, we recall the notion of safe diagnosability (see Paoli and Lafortune [2005]) as a necessary step in order to achieve fault tolerant supervision of DES. We

[1] The research of the first and second author is supported by MIUR. The research of the third author is supported in part by NSF grants CCR-03255571 and ECS-0624821.
[2] Corresponding author.

then introduce the new notion of *safe controllability*, which represents the capability, after the occurrence of a fault, of steering the system away from forbidden zones. We also define the new notion of *active fault tolerant system* with respect to post-fault specifications as the property of safely continuing operation after faults. We then present a general control architecture to deal with the FTC problem. This architecture is based on the use of a special kind of diagnoser, called "diagnosing-controller," which is used to safely detect faults and to switch between a nominal control policy and a bank of reconfigured control policies. In this sense, the exploited paradigm is that of switching control in which a high-level logic is used to switch between a bank of different controllers (see Blanke et al. [2003]).

2. SUPERVISORY CONTROL OF DES WITH FAULTS

Following the theory of supervisory control of DES (see, e.g., Chapter 3 of Cassandras and Lafortune [2007]), we consider a model of the uncontrolled system, denoted by G^{nom} and given in the form of an automaton, and a set of specifications on the controlled behavior. In general G^{nom} is expressed as the interconnection, via parallel composition, of a set of interacting components whose models are denoted by $(G_1^{\text{nom}}, \ldots, G_n^{\text{nom}})$. The behavior of G^{nom}, captured by the language $\mathcal{L}(G^{\text{nom}})$, must be restricted by control in order to satisfy the set of specifications. For this purpose, we design a *supervisor*, whose realization as an automaton is denoted by S^{nom}, and connect it with G^{nom} thereby obtaining the controlled system $G_{\text{sup}}^{\text{nom}} := G^{\text{nom}} \parallel S^{\text{nom}}$ with its associated language $\mathcal{L}(G_{\text{sup}}^{\text{nom}})$ satisfying the set of language specifications $\mathscr{K}^{\text{nom}}$.

Potential faults of the system components are usually considered at this point. In this regard, the G_i^{nom} component models are enhanced to include most likely faults and subsequent faulty behavior (see, e.g., Sampath et al. [1995]). Therefore, instead of the nominal model G^{nom}, we now have model $G^{\text{n+f}}$ that embeds the (potential) faulty behavior of the respective components. In the following, for the sake of simplicity, we consider a single fault event f. It follows that $\mathcal{L}(G^{\text{n+f}}) \supset \mathcal{L}(G^{\text{nom}})$ with corresponding event sets $E^{\text{n+f}} = E \cup \{f\}$, where $f \in E_{uo}^{\text{n+f}} \cap E_{uc}^{\text{n+f}}$, i.e., f is unobservable and uncontollable. This means that the actual controlled behavior of the system is described by $G_{\text{sup}}^{\text{n+f}} = G^{\text{n+f}} \parallel S^{\text{nom}}$. Since we are considering persistent faults, after any occurrence of fault f, the supervised system continues evolving according to well-defined *post-fault models* that are completely disjoint from the nominal supervised model.

By construction of S^{nom}, there are no undesired actions in the nominal part. However, undesired sequences of actions can arise in post-fault models due to the effective control actions of the nominal supervisor on faulty components, as captured in $G_{\text{sup}}^{\text{n+f}}$. Consequently, we must avoid that after fault f occurs, the system executes a forbidden substring from a given finite set Φ, where $\Phi \subseteq E^{\star}$. In essence, the elements of the set Φ capture sequences of events that become illegal after the occurrence of fault f. This situation can be formalized by defining the "illegal language" $\mathscr{K}_f$ as in Paoli and Lafortune [2005].

Under the supervision of S^{nom}, the resulting system behavior $\mathcal{L}(G_{\text{sup}}^{\text{n+f}})$ will contain the nominal controlled behavior $\mathcal{L}(G_{\text{sup}}^{\text{nom}})$, which satisfies the nominal specifications $\mathscr{K}^{\text{nom}}$, and in addition post-fault behavior that may include strings that are in the illegal language.

The design objectives of a fault tolerant supervision system can therefore be enumerated as follows:

- **A)** Diagnose the occurrence of event f before the system executes some illegal sequence in the set Φ;
- **B)** Force the system to stop its evolution before the execution of forbidden sequences;
- **C)** Steer the faulty system behavior in order to meet new (eventually degraded) post-fault specifications that are assumed to be expressed in the form of language $\mathscr{K}^{\text{deg}}$.

Note that objective **A** is achieved if the property of *safe diagnosability* described in Paoli and Lafortune [2005] is satisfied by the system. In the following, objective **B** will be studied in terms of a new property called *safe-controllability*, while objective **C** will be linked with the new property of *active fault tolerance*. It is important to emphasize that the post-fault specifications $\mathscr{K}^{\text{deg}}$ are in general incomparable with $\mathcal{L}(G_{\text{sup}}^{\text{n+f}})$; therefore, in order to satisfy them, it is necessary to switch from the nominal supervisor S^{nom} to a new supervisor denoted by S^{deg}, thereby following an active approach to fault tolerance in the sense of Blanke et al. [2003].

3. SAFE CONTROLLABILITY OF DES

This section is concerned with the definition and testing of the property of safe controllability for the purpose of the fault tolerance objectives described in the preceding section. The reader is referred to Sampath et al. [1995] and Paoli and Lafortune [2005] for any undefined notation or terminology. First, we recall the definition of diagnosability, introduced in Sampath et al. [1995], which states that a language L is diagnosable if it is possible to detect within a finite delay occurrences of faults using the record of observed events.

Definition 1. [**Diagnosable DES**] A prefix-closed language L that is live and does not contain loops of unobservable events is said to be diagnosable with bound n with respect to projection P_{o} and fault event f if the following holds: $(\exists n \in \mathbb{N})\ (\forall s \in \Psi(f))\ (\forall t \in L/s)\ (\|t\| \geq n \Rightarrow \mathcal{D})$ where the diagnosability condition $\mathcal{D}$ is: $\omega \in P_{\text{o}}^{-1}[P_{\text{o}}(st)] \cap L \Rightarrow f \in \omega$.

Objective **A** of the preceding section requires that after a fault f occurs, the system should not execute a forbidden substring from a given finite set Φ. This objective is captured by the property of safe diagnosability introduced in Paoli and Lafortune [2005] and now recalled.

Definition 2. [**Safe Diagnosable DES**] A prefix-closed language L that is live and does not contain loops of unobservable events is said to be safe diagnosable with respect to projection P_{o}, fault event f and forbidden language $\mathscr{K}_f$ if (i) L is diagnosable with bound n, with respect to P_{o} and f and (ii) $(\forall s \in \Psi(f))(\forall t \in L/s)$ such that $\|t\| = n$, let t_c, $\|t_c\| = n_{t_c}$, be the shortest prefix of t such that $\mathcal{D}$ holds, then $\overline{st_c} \cap \mathscr{K}_f = \emptyset$.

We make use of the Diagnoser Approach of Sampath et al. [1995] to test diagnosability and safe diagnosability.

The *diagnoser*, denoted by G^{diag}, is an automaton built from the system model $G^{\text{n+f}}_{\text{sup}}$. This automaton is used to perform diagnosis when it observes on-line the behavior of $G^{\text{n+f}}_{\text{sup}}$. The construction procedure of the diagnoser can be found in Sampath et al. [1995]. We recall here a theorem from Sampath et al. [1995] for testing (off-line) the diagnosability of a system using its diagnoser.

Theorem 1. [**Sampath et al. [1995]**] A language L is diagnosable with respect to the projection P_{o} and fault event f if and only if the diagnoser G^{diag} built starting from any generator of L has no F-indeterminate cycles.

By slightly modifying the diagnoser as explained in Paoli and Lafortune [2005], we obtain the so-called *safe-diagnoser*, denoted by $G^{\text{diag,s}}$, in which some states are labeled as *bad states* since they are reachable by executing strings in $\mathscr{K}_f$. As explained in Paoli and Lafortune [2005], the safe-diagnoser can be used to test (off-line) the property of safe diagnosability; we recall the following theorem.

Theorem 2. [**Paoli and Lafortune [2005]**] Consider a diagnosable language L. L is safe diagnosable with respect to projection P_{o}, fault event f, and forbidden language $\mathscr{K}_f$ if and only if in the safe-diagnoser $G^{\text{diag,s}}$ built from any generator of L, (i) there does not exist a state q that is F-uncertain with a component of the form (x, ℓ) such that $f \in \ell$ and x is a bad state and (ii) there does not exist a pair of states q, q' such that: (i) q is F-certain with a component of the form (x, ℓ) such that $f \in \ell$ and x is a bad state; (ii) q' is F-uncertain; and (iii) q is reachable from q' through an event $e \in E_o$.

We have argued previously that safe diagnosability is a first necessary step in order to achieve fault tolerant supervision of DES. If the system is safe diagnosable, reconfiguration actions should be forced upon the detection of faults prior to the execution of unsafe behavior, thereby achieving the objective of fault tolerant supervision. The first step to reconfigure the system is to disable the nominal supervisor and prevent the system from executing a forbidden substring. For this purpose, it is useful to introduce the following property.

Definition 3. [**Safe Controllable DES**] A prefix-closed language L that is live and does not contain loops of unobservable events is said to be safe controllable with respect to the projection P_{o}, fault event f, and forbidden language $\mathscr{K}_f$ if (i) L is safe diagnosable with respect to P_{o}, f, and $\mathscr{K}_f$ and (ii) consider any string $s \in L$ such that $f \in s$ and $s = v\sigma$ with $\sigma \in E_o$. Suppose that $\mathcal{D}$ does not hold for v while it holds for s. Then $(\forall t \in L/s)$ such that $t = u\xi$ with $\xi \in \Phi$, $\exists z \in E_c$ such that $z \in u$.

In words, a language is safe controllable if for any string that contains a fault and a forbidden substring, there exists (i) an observable event that assures the detection of the fault before the system executes the forbidden substring and (ii) a controllable event after the observable event but before the forbidden substring. In this way, after the detection of the fault, it is always possible to disable the controllable event and avoid unsafe behavior.

Consider an automaton G generating language L and assume that L is safe diagnosable with respect to the projection P_{o}, fault event f, and forbidden language $\mathscr{K}_f$. Denote with $\mathcal{FC}$ the set of *first-entered certain states* in the safe-diagnoser $G^{\text{diag,s}}$ built from G; namely, $\mathcal{FC}$ is the set of all safe-diagnoser states q such that q is F-certain and there exists a safe-diagnoser state q' which is F-uncertain and such that q is reachable from q' through an event $\sigma_{\text{o}} \in E_o$. The set $\mathcal{FC}$ contains a finite number of elements:

$$\mathcal{FC} = \{q_{\text{i}}\}\,,\ (i = 1 \dots m)\,. \tag{1}$$

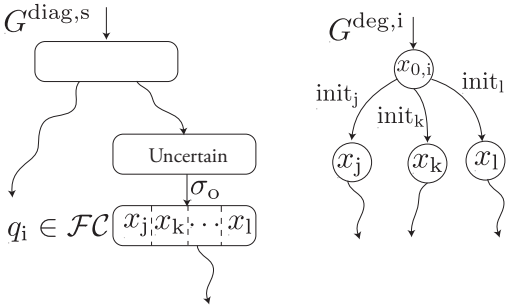

Fig. 1. Post-fault uncontrolled model.

For any $q_{\text{i}} = \{(x_{\text{j}}, F); (x_{\text{k}}, F) \dots, (x_{\text{l}}, F)\} \in \mathcal{FC}$ $(i = 1 \dots m)$ we build a new post-fault uncontrolled model, $G^{\text{deg}}_{\text{i}}$, by taking the accessible part of $G^{\text{n+f}}$ from all the distinct states $x_{\text{j}}, x_{\text{k}} \dots x_{\text{l}}$ of $G^{\text{n+f}}$ that appear in the i-th safe-diagnoser state; see Fig. 1. To make the model deterministic, we add a new initial state $x_{0,\text{i}}$ and connect it with new events called "init_{j}, init_{k}, init_{l}" to the distinct states of $G^{\text{n+f}}$ that appear in the safe-diagnoser state q_{i}. The index of init is used to make these events distinct. Note that init is uncontrollable and unobservable. In practice, this accessible part will be within the faulty part of $G^{\text{n+f}}$, since the occurrence of the fault has forced the system outside its original nominal behavior G^{nom}.

Using the above terminology, we can now present a procedure to test the property of safe controllability.

Proposition 3. Consider automaton G generating language L and assume that L is safe diagnosable with respect to the projection P_{o}, fault event f, and forbidden language $\mathscr{K}_f$. Consider the set $\mathcal{FC}$ of first-entered certain states in the safe-diagnoser $G^{\text{diag,s}}$ built from G. Language L is safe controllable if and only if $\forall q_{\text{i}} \in \mathcal{FC}$, language $\{\varepsilon\}^{\downarrow C}$, computed with respect to the post-fault uncontrolled model $G^{\text{deg}}_{\text{i}}$, does not contain any element of Φ as a substring.

Remark. Standard techniques to remove illegal substrings from a language can be used to test Proposition 3; see, e.g., Section 3.3 in Cassandras and Lafortune [2007].

4. ACTIVE FAULT TOLERANCE OF DES

If language $\mathcal{L}(G^{\text{n+f}}_{\text{sup}})$ is safe controllable then it is always possible to detect any occurrence of event f in a bounded number of observable events and without executing any forbidden action; moreover, in any continuation after the detection of fault f that contains a forbidden action in Φ, there always exists at least one controllable event z that can be disabled to prevent the system from executing unsafe actions. Entering certain state $q_{\text{i}} \in \mathcal{FC}$ should therefore trigger an interrupt signal INT_{i} that disables the controllable event z. Moreover, the same interrupt signal can be used to disable the nominal supervisor S^{nom} and

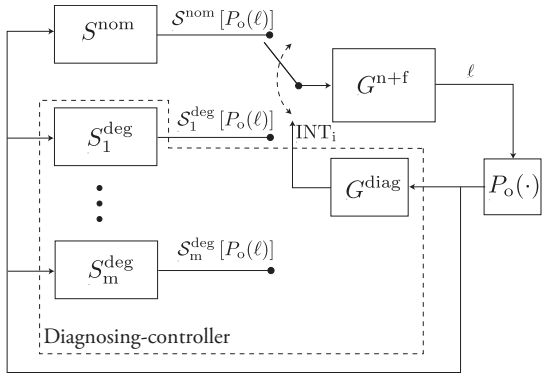

Fig. 2. Fault tolerant supevision architecture for DES.

enable a new supervisor $S_{\text{i}}^{\text{deg}}$ to be designed in order to meet post-fault specifications $\mathscr{K}_{\text{i}}^{\text{deg}}$.

Starting from post-fault uncontrolled model $G_{\text{i}}^{\text{deg}}$, a set of requirements on the controlled behavior can be designed resulting in post-fault degraded specifications $\mathscr{K}_{\text{i}}^{\text{deg}}$, which in general are sublanguages of the language marked by $G_{\text{i}}^{\text{deg}}$. Note that $\mathscr{K}_{\text{i}}^{\text{deg}}$ need not be prefix-closed if the requirements include the ability to reach one of the so-called *recovery states* that are included and marked in $G^{\text{n+f}}$. In simpler cases, $\mathscr{K}_{\text{i}}^{\text{deg}}$ will be a prefix-closed sublanguage of the language generated by $G_{\text{i}}^{\text{deg}}$. In practice $\mathscr{K}_{\text{i}}^{\text{deg}}$ can be designed as the language generated or marked by an automaton $H_{\text{i}}^{\text{deg}}$ built by removing from $G_{\text{i}}^{\text{deg}}$ illegal states in $G^{\text{n+f}}$ and all strings that contain some undesired substring that may be specific to each $i = 1 \dots m$. In some cases, it might be desirable to specify a minimal required behavior $\mathscr{K}_{\text{i}}^{\text{deg,min}}$ to be satisfied after the detection of the fault event f. Considering this set of degraded post-fault specifications $\mathscr{K}_{\text{i}}^{\text{deg}}$ $(i = 1 \dots m)$, we present the following definition.

Definition 4. **[Active Fault Tolerant DES]** Language $\mathcal{L}(G^{\text{n+f}})$ is said to be active fault tolerant if for all $i = 1 \dots m$, there exists a sublanguage of $\mathscr{K}_{\text{i}}^{\text{deg}}$ that is controllable and observable with respect to $\mathcal{L}(G_{\text{i}}^{\text{deg}})$.

Remark. In order to test Definition 4 for prefix-closed specifications $\mathscr{K}_{\text{i}}^{\text{deg}}$, we can compute $\{\varepsilon\}^{\downarrow C}$ with respect to $\mathcal{L}(G_{\text{i}}^{\text{deg}})$ and test if the result is contained within $\mathscr{K}_{\text{i}}^{\text{deg}}$. Of course, this solution is likely to be impractical because too restrictive. Another possibility is to compute the supremal controllable and normal sublanguage of $\mathscr{K}_{\text{i}}^{\text{deg}}$ with respect to $\mathcal{L}(G_{\text{i}}^{\text{deg}})$ (see Cho and Marcus [1989], Inan [1994]). This solution may also be too restrictive since the normality condition is stronger than the required observability condition. In this case, one could use existing algorithms for calculating maximal controllable and observable sublanguages of $\mathscr{K}_{\text{i}}^{\text{deg}}$; for instance, the VLP-PO algorithm presented in Hadj-Alouane et al. [1996] can be used for this purpose. For cases where $\mathscr{K}_{\text{i}}^{\text{deg}}$ is not prefix-closed, the test for active fault tolerance is more complicated, since the $\downarrow C$ operation deals with prefix-closed languages. One could still compute the supremal controllable and normal sublanguage of (marked language) $\mathscr{K}_{\text{i}}^{\text{deg}}$; however, if this approach returns the empty set, we will not know if active fault tolerance is violated, as $\mathscr{K}_{\text{i}}^{\text{deg}}$ could still possess a controllable and observable sublanguage. In this case, the recent results in Yoo and Lafortune [2006] could be used to test the existence or not of such a language. However, a positive test may still yield a solution that is deemed too restrictive. More research is required regarding the development of algorithms for computing controllable and observable sublanguages of non-prefix-closed languages.

In Fig. 2 a possible architecture for active fault tolerant control of DES is presented. During nominal functioning, the partial observation loop is closed on the nominal supervisor S^{nom} which, recording the observation $P_{\text{o}}(\ell)$, issues the control action $\mathcal{S}^{\text{nom}}[P_{\text{o}}(\ell)]$ that encodes the enabled events after the system $G^{\text{n+f}}$ executes the string ℓ. In parallel to the control loop, the diagnoser [3] G^{diag} uses the same observations to detect occurrences of the fault event f. If the system is safe diagnosable, after any occurrence of event f, the diagnoser detects the fault in a bounded number of events and before the system executes any forbidden string in Φ. When the diagnoser becomes F-certain entering state $q_{\text{i}} \in \mathcal{FC}$ (mapped from the safe-diagnoser), the interrupt signal INT_i is issued. If the system is safe controllable, we know that it is possible to stop the evolution of $G^{\text{n+f}}$ before it executes forbidden substrings in Φ. The same interrupt signal is used to switch from the nominal supervisor S^{nom} to the post-fault supervisor $S_{\text{i}}^{\text{deg}}$. The existence of this supervisor is assured if the active fault tolerance property holds; in this case, the behavior of G^{n+f} can be controlled in order to satisfy the specification $\mathscr{K}_{\text{i}}^{\text{deg}}$. As depicted in Fig. 2, it is possible to embed both the diagnoser G^{diag} and the bank of post-fault supervisors $S_{\text{i}}^{\text{deg}}$ in a unique unit called the *diagnosing-controller*, whose structure is shown in Fig. 3.

The diagnosing-controller is an automaton built from the diagnoser G^{diag} and considering the model $G^{\text{n+f}}$. If $G_{\text{sup}}^{\text{n+f}}$ is safe diagnosable, then after any occurrence of fault event f the diagnoser enters, in a bounded number of events, a first-entered certain state $q_{\text{i}} = \{(x_{\text{j}}, F); (x_{\text{k}}, F) \dots, (x_{\text{l}}, F)\} \in \mathcal{FC}$ (again, mapped from the safe-diagnoser) and, after that moment, all the other reachable states in G^{diag} are certain states. When G^{diag} enters q_{i}, the signal INT_i is generated and used to disable the controllable event z to avoid any occurrence of forbidden substrings; moreover the same event is used to disable the nominal supervisor S^{nom}. Considering these facts, when G^{diag} enters q_{i}, we are sure that event f has occurred and the actual state in $G^{\text{n+f}}$ is one of the states in the list of q_{i}, i.e., $x_{\text{j}}; x_{\text{k}} \dots x_{\text{l}}$. Note that if the system is safe diagnosable, none of the states $x_{\text{j}}; x_{\text{k}} \dots x_{\text{l}}$ will be reached by the execution of forbidden substrings (see Paoli and Lafortune [2005]). As previously stated, $x_{\text{j}}; x_{\text{k}} \dots x_{\text{l}}$ can be considered as initial states for the uncontrolled i-th post-fault model of the system. Moreover, the post-fault evolution can be reconstructed considering the connectivity in $G^{\text{n+f}}$ starting from states $x_{\text{j}}; x_{\text{k}} \dots x_{\text{l}}$. At this point, if $\mathcal{L}(G^{\text{n+f}})$ is active fault tolerant with respect to post-fault specification $\mathscr{K}_{\text{i}}^{\text{deg}}$, it is possible to design a post-

[3] The standard diagnoser is used here as it is suffices for the present purpose. Relevant mapping of states is done from the safe-diagnoser to the diagnoser regarding the set $\mathcal{FC}$.

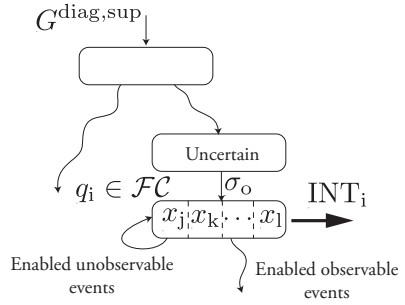

Fig. 3. The diagnosing-controller for the example in Fig. 1.

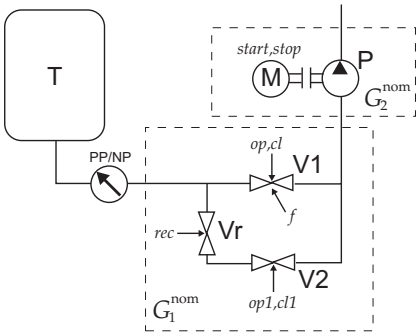

Fig. 4. The hydraulic system.

fault supervisor S_i^{deg}. As depicted in Fig. 3, the realization of the post-fault supervisor S_i^{deg} can be considered as directly connected to first-entered F-certain state q_i in the diagnoser. In the following an algorithmic procedure to build the diagnosing-controller is presented.

Procedure to build the diagnosing-controller.
Step 1: Build the diagnoser G^{diag} from $G_{\text{sup}}^{\text{n+f}}$;
Step 2: For any $q_\text{i} = \{(x_\text{j}, F); (x_\text{k}, F) \ldots, (x_\text{l}, F)\} \in \mathcal{FC}$;
- **Step 2.1:** Stop the evolution of G^{diag} after q_i and enable signal INT_i when entering q_i;
- **Step 2.2:** Build the post-fault model G_i^{deg};
- **Step 2.3:** Compose G_i^{deg} with a realization H_i^{deg} of specification $\mathscr{K}_\text{i}^{\text{deg}}$. Define $R^{\text{deg}} = H_\text{i}^{\text{deg}} \times G_\text{i}^{\text{deg}}$;
- **Step 2.4:** Starting from R^{deg}, build the post-fault supervisor realization S_i^{deg} using techniques from supervisory control theory;
- **Step 2.7:** Overlap the initial state of S_i^{deg} with state q_i of G^{diag};

Step 3: Call the resulting automaton $G^{\text{diag,sup}}$.

5. EXAMPLE

Consider the hydraulic system of Fig. 4; the system is composed of a tank T, a pump P, a set of valves (V1, V2, and Vr), and associated pipes. The pump P is used to move fluid from the tank through the pipe and must be coordinated with the set of redundant valves. The system is equipped with a pressure sensor. Events *op* and *cl* are used to open and close valve V1, events *op*1 and *cl*1 are used to open and close valve V2, event *rec* is used to open safety valve Vr. All these events are observable and controllable. Moreover events *start* and *stop*, observable and controllable, are used to switch on and off the pump, respectively. In Fig. 5 (a) the nominal model of the

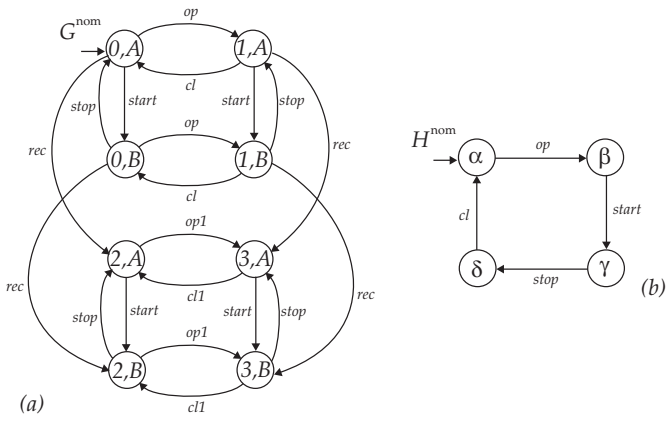

Fig. 5. The hydraulic system example: (a) global nominal model G^{nom}; (b) nominal specification H^{nom}.

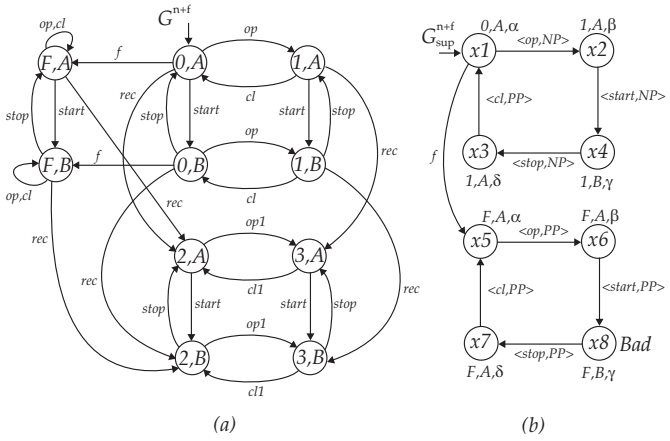

Fig. 6. The hydraulic system example: (a) complete model $G^{\text{n+f}}$; (d) complete supervised model $G_{\text{sup}}^{\text{n+f}}$.

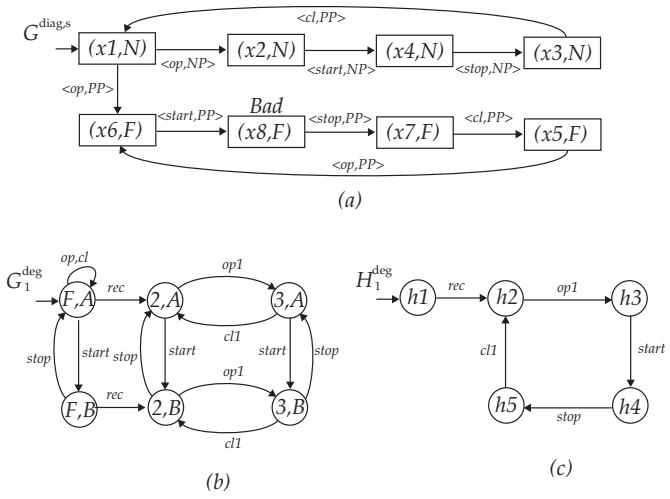

Fig. 7. The hydraulic system example: (a) safe diagnoser $G^{\text{diag,s}}$; (b) post-fault model G_1^{deg}; (c) post-fault specification H_1^{deg}.

system G^{nom} is depicted; this model must be controlled according to the specification defined by automaton H^{nom} shown in Fig. 5 (b). It is easy to see that $\mathcal{L}(H^{\text{nom}})$ is controllable and observable with respect to $\mathcal{L}(G^{\text{nom}})$. Due to malfunctioning, valve V1 may get stuck closed; this fact

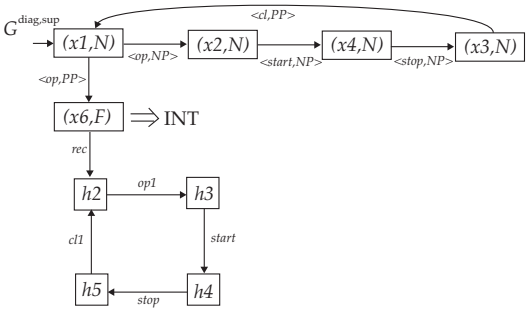

Fig. 8. The hydraulic system example: the diagnosing-controller $G^{\mathrm{diag,sup}}$.

is modeled using unobservable and uncontrollable event f. According to this refined model, the automaton $G^{\mathrm{n+f}}$ modeling the uncontrolled system is depicted in Fig. 6 (a); in Fig. 6 (b) the effect of the nominal supervision policy on $G^{\mathrm{n+f}}$ is denoted by $G^{\mathrm{n+f}}_{\mathrm{sup}}$. Pressure sensor readings are attached to events; note that if valves are closed, the sensor reads an over pressure in the pipe (PP), while, if valves are open, the sensor reads no over-pressure in the pipe (NP). The situation in which the pump is working with closed valves has to be avoided because it is unsafe: $\Phi = \{start\}$. Note that this situation is feasible in $G^{\mathrm{n+f}}_{\mathrm{sup}}$ where state $x8$ is labeled as bad (see Paoli and Lafortune [2005]). In Fig. 7 (a), the safe-diagnoser of $G^{\mathrm{n+f}}_{\mathrm{sup}}$ is shown. Since the bad state $x8$ is not present in any of the uncertain states or in a first-entered certain state, the system is safe diagnosable; in this case the set of first-entered certain states is $\mathcal{FC} = \{(x6, F)\}$. Figure 7 (b) shows the feasible evolution G^{deg}_1 after detection; here, no new initial state and init_j event is needed (see Fig. 1) since G^{deg}_1 is deterministic. It is easy to prove that $\{\varepsilon\}^{\downarrow C}$ computed with respect to $\mathcal{L}(G^{\mathrm{deg}}_1)$ is equal to $\{\varepsilon\}$, therefore $G^{\mathrm{n+f}}_{\mathrm{sup}}$ turns out to be safe controllable. Starting from G^{deg}_1, the post-fault specification generated by automaton H^{deg}_1 in Fig. 7 (c) can be designed. Since this specification turns out to be controllable and observable with respect to G^{deg}_1, the system is active fault tolerant. Finally, the diagnosing-controller $G^{\mathrm{diag,sup}}$ for this example is shown in Fig. 8.

6. CONCLUSIONS

The main contribution of this paper is to present a new supervisory control architecture that deals with the FTC problem in the framework of automata. We follow the "active approach," in which the supervisor actively reacts to the detection of a malfunctioning component in order to eventually meet degraded control specifications. We employ a switching control paradigm in which a high-level logic is used to switch between different controllers. Starting from an appropriate model of the system, we have shown how the notion of safe diagnosability is a necessary step in order to achieve fault tolerant supervision of DES. We have introduced the new notions of "safe controllability" and "active fault tolerant system," as a means to characterize the conditions that must be satisfied when solving the FTC problem using the active approach. Tests for these properties were presented. Finally, we have proposed a procedure to design an automaton called "diagnosing-controller" which is able, if the system satisfies the properties introduced, to solve the fault tolerant supervision problem.

REFERENCES

M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and fault-tolerant control*. Springer-Verlag, 2003.

R. Boel and J. van Schuppen. Decentralized failure diagnosis for discrete-event systems with constrained communication between diagnosers. *Proceedings of the 6th International Workshop on Discrete Event Systems*, 2002.

C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems - Second Edition*. Springer, 2007.

H. Cho and S. I. Marcus. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Math Control, Signals Systems*, 2(1):47 – 69, 1989.

E. Dumitrescu, A. Girault, H. Marchand, and E. Rutten. Optimal discrete controller synthesis for modeling fault-tolerant distributed systems. *Proceedings of the 1st IFAC Workshop on Dependable Control of Discrete Systems*, 2007.

N. Ben Hadj-Alouane, S. Lafortune, and F. Lin. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. *Discrete Event Dynamic Systems: Theory and Applications*, 6(4):379 – 427, 1996.

C. Hadjicostis. Probabilistic fault detection in finite-state machines based on state occupancy measurements. *IEEE Transactions on Automatic Control*, 50(12):2078 – 2083, 2005.

K. Inan. Nondeterministic supervison under partial observation. *11th International Conference on Analysis and Optimization of Systems: Discrete Event Systems*, 1994.

M. V. Iordache and P. J. Antsaklis. Resilience to failure and reconfigurations in the supervision based on place invariants. *Proceedings of the 2004 American Control Conference*, 2004.

S. Jiang and R. Kumar. Diagnosis of repeated failures for discrete event systems with linear-time temporal logic specifications. *IEEE Transactions on Automation Science and Engineering*, 3(1):47 – 59, 2006.

A. Paoli and S. Lafortune. Safe diagnosability for fault tolerant supervision of discrete event systems. *Automatica*, 41(8):1335 – 1347, 2005.

M. Sampath, R. Sangupta, and S. Lafortune. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555 – 1575, 1995.

Q. Wen, R. Kumar, J. Huang, and H. Liu. Weakly fault-tolerant supervisory control of discrete event systems. *Proceedings of the 2007 American Control Conference*, 2007a.

Q. Wen, R. Kumar, J. Huang, and H. Liu. Fault-tolerant supervisory control of discrete event systems: formalisation and existence results. *Proceedings of the 1st IFAC Workshop on Dependable Control of Discrete Systems*, 2007b.

T.-S. Yoo and S. Lafortune. Solvability of centralized supervisory control under partial observation. *Discrete Event Dynamic Systems: Theory and Applications*, 16 (4):527 – 553, 2006.