Group-based method for fault identification using the diagnoser approach

Luiza Ocheană*, Dan Popescu*, Luca Ferrarini**

** Faculty of Automatic Control and Computer Science University "Politehnica" of Bucharest Bucharest, Romania (e-mail: luiza.ocheana@yahoo.com, dan_popescu_2002@yahoo.com)*
*** Dipartimento di Elettronica e Informazione Politecnico di Milano Milan, ITALY (e-mail: ferrarin@elet.polimi.it)*

Abstract: The paper presents a new method for fault detection in discrete event systems that was developed based on the classical diagnoser approach. The goal was to reduce the amount of computation that has to be done for isolating a fault as soon as possible. To do this, we have reduced the number of states for the process model, the system model and consequently, the number of states for the diagnoser. In order to demonstrate, the method is used and tested on a compressor station, in a comparative analysis with the classical method.

Keywords: diagnosis, fault detection, fault isolation, industrial process.

1. INTRODUCTION

One of the biggest problems in automation systems is that things can go wrong. Unfortunately, there is no such thing as perfect equipment. Moreover, the behaviour of a system can be modified in time. In industrial systems, if something doesn't work properly or doesn't work at all, this can lead to huge loses, or even worse, to accidents. This is why these faults need to be detected and isolated as soon as accurately as possible.

The increase in requirements for the control system of modern industrial plants also leads to an increase in their vulnerability, mainly because of the big number of sub-processes and their complex interactions. Therefore, it is necessary to develop and implement adequate methods of failure diagnosis, capable of detecting the failures and their location, taking into account all the available details regarding the process and its operation. The knowledge regarding the components of the automation process, its desired behaviour, failure type and its location, are indispensable requirements for the implementation of adequate methods for failure detection. Preventive failure detection can also be used to increase the availability and to reduce the breakdown situations, usually of considerable economic consequences.

Failure detection in dynamic systems is both an old problem (Willsky, 1976) and also a modern one. (Gertler, 1988) identified three layers of failure diagnosis process: failure detection, failure isolation and failure identification. Even since 1995 the detection and isolation of faults in discrete event systems has received a lot of attention (Sampath et al., 1995). Later this work was extended in many directions. For example, (Lunze and Schroder, 2001) showed that the observer of a finite-state stochastic automaton cannot be represented by another finite-state stochastic automaton. (Thorsley and Teneketzis, 2005) introduced the notion of stochastic diagnosability assuming the use of reliable sensors. The authors also presented the method of building stochastic diagnoser. Later, the work was extended to unreliable sensors by (Throsley et al., 2008). (Athanasopoulou et al., 2006) described a method for computing the observation likelihood of a stochastic automaton and use this method to find the most-likely stochastic automaton. (Wen-Chiao et al., 2009) developed sequential window diagnosers (SWDs) utilizing the notions of state probability vector and stochastic diagnoser probability transition matrices. (Ribot et al., 2009) presented a formal characterization of the diagnostic and prognostic problems in order to support the maintenance of a complex system. (Gascard and Simeu-Abazi, 2011) proposed a model-based approach to passive online fault diagnosis for timed systems, describing the system to be diagnosed as a network of communicating timed automata. (Ferrarini et al., 2011) described a whole methodology for the design of a real-time diagnostic system, from the specification to implementation, along with a complete testing on a real industrial automated system. A novel practical algorithm for real-time diagnosis suitable for automated devices (TiDiaM) is presented by (Ferrarini et al., 2011). (Rincon, 2012) presented the study of the detection and diagnosis of multiple faults in a Gas Turbine using principal component analysis and structured residuals method.

A difficult issue in large industrial systems is that there may be multiple faults at one moment of time. And even if the probability of simultaneous failure is low, in these cases, we may find another difficulty: the personnel don't have enough time to fix all faults in time.

Multiple faults can lead to the phenomenon known as "fault masking", meaning that the presence of one fault may hide other faults. This is often seen when speaking about faults which have a major impact on system operation and in case of sensor failure. This is why in theory it is very common to consider the sensors as ideal (do not ever fail).

Because of this, the interest in developing dedicated fault diagnosis methods both for sensor systems and for actuators has grown. For example, (Alexadru and Popescu, 2001) have presented the results of the experiments on an electrical motor in order to perform on-line diagnosis.

In the sensor domain, (Zhu et al., 2010) has presented a method for fault diagnosis in sensor systems based on principle component analysis.

Also trying to solve some errors in measurements, the sensor manufacturers has developed intelligent sensors, smart devices with signal processing functions shared by distributed machine intelligence (Yurish, 2010).

There is also interest in the field of wireless measurements: (Rughinis and Gheorghe, 2013) have developed an attack and fault detection framework for wireless sensor networks - TinyAFD.

Fault diagnosis in large systems requires complex computation. One approach is to break the system and the diagnoser into small and independent ones. Unfortunately, this may lead to diagnostic systems that do not take into account all possible interactions. Another idea is to concentrate on the diagnosis of one type of fault only (one diagnoser per fault). (Pencolé et al., 2006) proposed to analyse the system in order to detect a subsystem that is sufficient for diagnosing this particular type of fault.

The classical diagnoser method can become difficult to implement in large and complex systems due to the large size of the sets of descriptive parameters which are obtained by computing the diagnoser as shown in section II. Large sets lead to the necessity of large computing and storage capabilities. This paper presents a method for fault detection that was born from the need to reduce the amount of computation that has to be done in order to isolate a fault as soon as possible. The method we propose is based on the fact that any automation system is made of several components that have the same characteristics and operate in a specific manner. Based on this, in the first phase, we split the process into disjunctive groups and in the second phase, we detect the fault that occurred in the operation of the process using the diagnoser approach. By this method, we obtain an important reduction in the required number of states used to represent the process and the attached controller, which accelerates the process of tracking and identifying errors. The method is different than any other approach presented above, but it is in line with the trends in fault diagnosis domain.

The rest of the paper is organized as follows: in Section II all the necessary notations and definitions are introduced in order to establish the basis for Section III which introduces the new method for fault detection, based on grouping criteria. Section IV describes an example of implementation on a compressor station. Section V concludes the paper.

2. THE DIAGNOSER AUTOMATON

Because the method presented in the following section is based on the classical diagnoser approach (Ferrarini et al., 2011), we will describe here the basic rules and methods to build the plant model, the control model and finally the diagnoser automaton.

2.1. The plant model

The plant model is described as an automaton which represents the model of physical devices to be controlled (without control):

$$P: = (X_p, \Sigma_p, \delta_p, x_{0p}), \tag{1}$$

where X_p denotes the set of states, Σ_p represents the set of events, δ_p is the transition function between the states and x_{0p} represents the initial (start) state.

The set of events can be partitioned into observable and unobservable events as follows:

$$\Sigma_p = \Sigma_{p-o} \cup \Sigma_{p-uo}. \tag{2}$$

The set of the observable events (Σ_{p-o}) contains all the events from the plant that can be observed. The observable events in a system may be one of the following: commands given by the control system, sensor readings after the execution of the commands or changes of sensor readings. The set of the unobservable events (Σ_{p-uo}) contains events that cannot be observed, failure events or other events that cause changes in the system, without being detected by the sensors.

The plant model is usually represented as parallel composition of each elementary component also measuring and control equipment (sensors, valves, motors, pumps, etc.).

2.2. The control model

The control model represents the desired behaviour of the plant and is also described as an automaton, similar with the plant model:

$$C: = (X_C, \Sigma_o, \delta_C, x_{0C}), \tag{3}$$

In equation (3), X_C represents the set of control states, Σ_o denotes the set of observable events, δ_C is the transition function and x_{0C} represents the initial state.

The set of observable events can be partitioned into subset of measures (Σ_{oe}) and subset of commands (Σ_{ce}) as follows:

$$\Sigma_o = \Sigma_{oe} \cup \Sigma_{ce}. \tag{4}$$

In the control model, we have two cycles that are identified: the nominal cycle and the error handling model (Seungjoo and Dawn, 2005). The first one represents the nominal behaviour, the sequence of the events describing the behaviour of the system, as if nothing goes wrong. The second one describes the behaviour of the system when one or more faults occur. The faulty behaviour is connected to the nominal one and describes the evolution of the system after each specific fault occurrence from each possible nominal behaviour state.

2.3. The diagnoser automaton

The above sub-sections describe how to represent the plant model P and the control model C. Now, the system model can be considered as a parallel structure:

$$\begin{cases} S:=P\|C \\ S:=(X_S,\Sigma_S,\delta_S,x_{0S}) \end{cases}. \tag{5}$$

Having the system model and the set of fault labels:

$$\Delta f:=\{F_1, F_2, \dots, F_n\}, \tag{6}$$

the classical diagnoser automaton can be synthesized as described by Sampath et al. (1995):

$$D:=(X_D,\Sigma_D,\delta_D,x_{0D}). \tag{7}$$

3. THE PROPOSED METHOD

The method we have developed involves, in the first phase, partitioning the process into disjunctive groups and in the second phase, detecting the fault that occur in the operation of the process. By this method, we obtain an important reduction in the required number of states used to represent the process and the attached controller, which accelerates the process of tracking and identifying errors.

3.1. Grouping the components

The efficiency of the method is strongly influenced by the chosen grouping criteria depending on the application we consider.

Within an industrial plant, we can group the automation equipment into several types, according to the following criteria:

Types of components

The components of a facility/plant can be grouped in two main categories, each of them consisting in several types:

- execution components (pneumatic valves, electric valves, pumps, compressors, engines, and so on);
- measurement components (sensors: pressure, temperature, flow, level, weighing, position and so on).

By grouping all or some components of the same type (for example pneumatic valves), the corresponding model could be described by an automaton having the same number of states and events.

Response time

The response time of the components is not a criterion for the method described above. However, if we want to supplement it with time used in TiDiaM, it becomes a very important criterion to consider.

The operation of the components

Depending on the operation of the plant as a whole and the control logic, each type can be divided into several categories. For example, a category can consist of valves that act similar in the operation (meaning that will be either closed or opened at the same time) or work alternatively (will not be all closed or opened at the same time).

This criterion is highly dependent on the particularities of the plant and the control sequences.

3.2. Method

Suppose that the plant has n components:

$$\Pi=\{P_1, P_2, \dots, P_n\}, \tag{8}$$

and the plant model P is a parallel composition of its components:

$$P=P_1\|P_2\|\dots\|P_n, \tag{9}$$

where P_i are automata described by:

$$\begin{cases} P_i:=(X_{pi},\Sigma_{pi},\delta_{pi},x_{0pi}) \\ i\in\{1,2,\dots,n\} \end{cases}. \tag{10}$$

Like in the single automaton model, the event sets for each component can be partitioned into the set of observable (Σ_{pi-o}) and the set of unobservable events (Σ_{pi-uo}), as:

$$\Sigma_{pi}=\Sigma_{pi-o}\cup\Sigma_{pi-uo}. \tag{11}$$

Suppose we group the components in several types, according to one or more criteria presented above. We define the following partition of Π:

$$\begin{cases} \Pi=\Pi_1\cup\Pi_2\cup\dots\cup\Pi_s \\ s<n \end{cases}, \tag{12}$$

where:

$$\begin{cases} \Pi_1=\{P_{11},P_{12},\dots,P_{1i}\}-type1 \\ \Pi_2=\{P_{21},P_{22},\dots,P_{2j}\}-type2 \\ \dots \\ \Pi_s=\{P_{s1},P_{s2},\dots,P_{sk}\}-type3 \end{cases}, \tag{13}$$

and:

$$\begin{cases} \Pi_h\subset\Pi, h\in\{1,2,\dots,s\} \\ \Pi_i\cap\Pi_j=\Phi\,\forall i\neq j \\ i,j\in\{1,2,\dots,s\} \end{cases}. \tag{14}$$

For each group (type h):

$$\begin{cases} \Pi_h=\{P_{h1},P_{h2},\dots,P_{hk}\} \\ h\in\{1,2,\dots,s\} \end{cases}, \tag{15}$$

we consider an automaton:

$$G_h:=(X_{Ghi},\Sigma_{Ghi},\delta_{Ghi},x_{0Gh}), \tag{16}$$

where the elements of the sets X_{Ghi} and Σ_{Ghi} are dependent on the corresponding elements of process components of Π_h.

So, if:

$$\begin{cases} P_{hi}=(X_{hi},\Sigma_{hi},\delta_{hi},x_{0hi}) \\ i\in\{1,2,\dots,k\}, h\in\{1,2,\dots,s\} \end{cases} \tag{17}$$

and:

$$\begin{cases} X_{hi} = \{x_{hi}^1, x_{hi}^2, \dots, x_{hi}^a\} \\ \Sigma_{hi} = \{\sigma_{hi}^1, \sigma_{hi}^2, \dots, \sigma_{hi}^b\} \\ i \in \{1,2,\dots,k\}, h \in \{1,2,\dots,s\} \end{cases}, \qquad (18)$$

then:

$$\begin{cases} X_{Gh} = \{x_{Gh}^1, x_{Gh}^2, \dots, x_{Gh}^a\} \\ \Sigma_{Gh} = \{\sigma_{Gh}^1, \sigma_{Gh}^2, \dots, \sigma_{Gh}^b\} \\ h \in \{1,2,\dots,s\} \end{cases} \qquad (19)$$

are defined as:

$$\begin{cases} x_{Gh}^i = f_{X_{Gh}}(x_{h1}^i, x_{h2}^i, \dots, x_{hk}^i) \\ \sigma_{Gh}^i = f_{\Sigma_{Gh}}(\sigma_{h1}^i, \sigma_{h2}^i, \dots, \sigma_{hk}^i). \\ h \in \{1,2,\dots,s\}, i \in \{1,2,\dots,k\} \end{cases} \qquad (20)$$

The set of group automata is defined as:

$$\Pi_G = \{G_1, G_2, \dots, G_s\}. \qquad (21)$$

For $f_{X_{Gh}}$ and $f_{\Sigma_{Gh}}$ we consider simple functions implemented by the following architecture (Fig. 1):

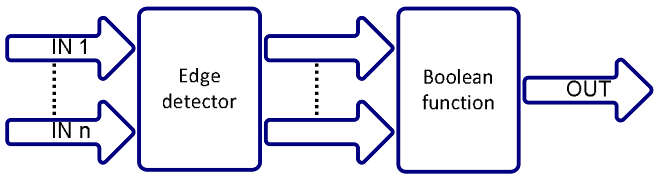

Fig. 1. Architecture of a typical function of group component.

The event set Σ_{Gh} can also be partitioned into observable and unobservable events as follows:

$$\Sigma_{Gh} = \Sigma_{Gh-o} \cup \Sigma_{Gh-uo}. \qquad (22)$$

Based on this grouping, we construct a group-based model of the plant G as a parallel composition of group models:

$$\begin{cases} G = G_1 || G_2 || \dots || G_s \\ G = (X_G, \Sigma_G, \delta_G, x_{0G}) \end{cases}. \qquad (23)$$

It is obvious that the dimension (the number of states and the number of events) of G is lower that the dimension of P:

$$\begin{cases} |X_G| < |X_P| \\ |\Sigma_G| < |\Sigma_P| \end{cases}. \qquad (24)$$

The controller for P is defined as:

$$C_P := (X_{CP}, \Sigma_{CP}, \delta_{CP}, x_{0CP}), \qquad (25)$$

$$\begin{cases} X_{CP} = X_{CP1} \cup X_{CP2} \cup \dots \cup X_{CPn} \\ \Sigma_{CP} = \Sigma_{p1-o} \cup \Sigma_{p2-o} \cup \dots \cup \Sigma_{pn-o}, \\ \delta_{CP} : X_{CP}\, X\Sigma_{CP} \to X_{CP} \end{cases} \qquad (26)$$

where:

X_{CPi} - are the states of the controller corresponding to P_i.

Σ_{pi-o} – is the set of observable events corresponding to P_i.

δ_{CP} - is the transition function between the states.

x_{0CP} - represents the initial (start) state.

We construct the group-based controller model:

$$C_G := (X_{CG}, \Sigma_{CG}, \delta_{CG}, x_{0CG}), \qquad (27)$$

where:

$$\begin{cases} X_{CG} = X_{CG1} \cup X_{CG2} \cup \dots \cup X_{CGs} \\ \Sigma_{CG} = \Sigma_{Gh-o} \\ \delta_{CG} : X_{CG}\, X\Sigma_{CG} \to X_{CG} \end{cases} \qquad (28)$$

and:

X_{CGi} – are the states of the controller corresponding to G_i.

Σ_{CGi} – is the set of observable events corresponding to G_i.

δ_{CG} - is the transition function between the states.

x_{0CG} - represents the initial (start) state.

Because of the way it was build, the group controller has fewer states and events that the initial one.

$$\begin{cases} |X_{CG}| < |X_C| \\ |\Sigma_{CG}| < |\Sigma_C| \end{cases}. \qquad (29)$$

Similar with (5) we construct the system model for P:

$$\begin{cases} S_P = P || C_P \\ S_P = (X_{SP}, \Sigma_{SP}, \delta_{SP}, x_{0SP}) \end{cases} \qquad (30)$$

and the group system model for G:

$$\begin{cases} S_G = G || C_G \\ S_G = (X_{SG}, \Sigma_{SG}, \delta_{SG}, x_{0SG}) \end{cases}, \qquad (31))$$

then:

$$\begin{cases} |X_{SG}| < |X_{SP}| \\ |\Sigma_{SG}| < |\Sigma_{SP}| \end{cases}.. \qquad (32)$$

3.3. Fault detection

Finally, we compute the diagnoser, according to the classical method.

We define the failure partition for P:

$$\Delta f_P = F_1 \cup F_2 \cup \dots \cup F_n, \qquad (33)$$

F_i - the set of failure for the component P_i

We group F_i in accordance with (13):

$$\Delta f_P = F_{\Pi_1} \cup F_{\Pi_2} \cup \dots \cup F_{\Pi_s}, \qquad (34)$$

F_{Π_i} the set of failure for the component Π_i

$$\begin{cases} F_{\Pi_1} = F_{11} \cup F_{12} \cup \ldots \cup F_{1i} \\ F_{\Pi_2} = F_{21} \cup F_{22} \cup \ldots \cup F_{2j} \\ \ldots \\ F_{\Pi s} = F_{s1} \cup F_{s2} \cup \ldots \cup F_{sk} \end{cases}, \quad (35)$$

where:

$$\begin{cases} F_{1a} = \{F_{1a}^1, F_{1a}^2, \ldots, F_{1a}^i\}, a \in \{1,2,\ldots,i\} \\ F_{2b} = \{F_{2b}^1, F_{2b}^2, \ldots, F_{2b}^j\}, b \in \{1,2,\ldots,j\} \\ \ldots \\ F_{sc} = \{F_{sc}^1, F_{sc}^2, \ldots, F_{sc}^k\}, c \in \{1,2,\ldots,k\} \end{cases}. \quad (36)$$

We can now define the failure partition for G as:

$$\Delta f_G = F_{G1} \cup F_{G2} \cup \ldots \cup F_{Gs}, \quad (37)$$

F_{Gi} - set of failure labels for G_i,
where:

$$\begin{cases} F_{G1} = \{F_{G1}^1, F_{G1}^2, \ldots, F_{G1}^i\} \\ F_{G2} = \{F_{G2}^1, F_{G2}^2, \ldots, F_{G2}^j\} \\ \ldots \\ F_{Gs} = \{F_{Gs}^1, F_{Gs}^2, \ldots, F_{Gs}^k\} \end{cases}, \quad (38)$$

$$\begin{cases} F_{G1}^a = f_{G_1}(F_{11}^a, F_{12}^a, \ldots, F_{1i}^a), a \in \{1,2,\ldots,i\} \\ F_{G2}^b = f_{G_2}(F_{21}^b, F_{22}^b, \ldots, F_{2j}^b), b \in \{1,2,\ldots,j\} \\ \ldots \\ F_{Gs}^c = f_{G_s}(F_{s1}^c, F_{s2}^c, \ldots, F_{sk}^c), c \in \{1,2,\ldots,k\} \end{cases}. \quad (39)$$

We can now compute the diagnosers: one for the classical model of the system S and the failure partition Δf_P:

$$D_P = (X_{DP}, \Sigma_{DP}, \delta_{DP}, x_{0DP}) \quad (40)$$

and one using the group system model S_G and the failure partition Δf_G:

$$D_G = (X_{DG}, \Sigma_{DG}, \delta_{DG}, x_{0DG}). . \quad (41)$$

It is now obvious that:

$$|\Delta f_P| < |\Delta f_G|, \quad (42)$$

and taking into consideration also the equation (32) we conclude that:

$$\begin{cases} |X_{DG}| < |X_{DP}| \\ |\Sigma_{DG}| < |\Sigma_{DP}| \end{cases}. \quad (43)$$

Because all the automata involved in the generation of D_G are smaller than the ones used for D_P, the diagoser D_G will also have fewer states and fewer events than the classical one.

When a failure occurs, D_G will isolate the fault and will estimate it belongs to F_{Gi}, meaning a fault at a component from P_i (of type $i \in \{1,2,\ldots,s\}$).

An automation system usually includes an events history component that keeps the record of all the observable events that occurred.

Consider we store the observable events recorded until the fault occurred, in an array (44).

$$L = \{y_t, y_{t-1}, y_{t-2}, \ldots\}, \quad (44)$$

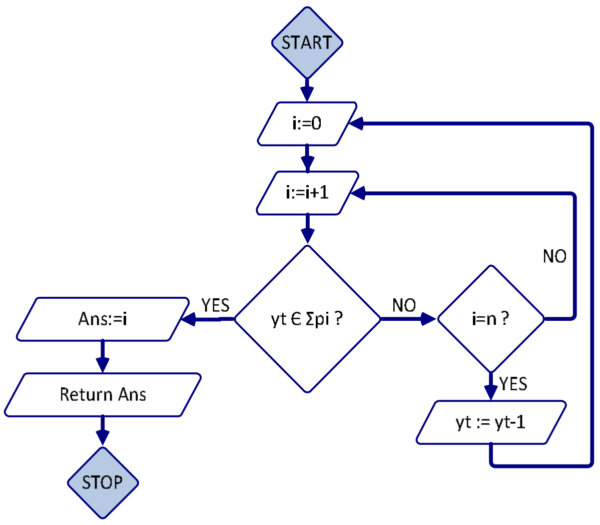

Fig. 2. Identifying the faulty component.

where y_t is an observable event recorded at time t.

We can search for the exact component that has a fault by searching the event from Σ_{pi} that appeared in L (Fig. 2).

The dimension of L depends on the local monitoring and control system and its storage capacity. Most of the systems ensure at least the storage for few days history events.

Having a multiprocessor architecture (w processors) we can identify the defective component using a parallel algorithm (Fig. 3).

Both Fig. 2 and Fig. 3 present algorithms for searching one single fault. They can be easily improved, to search for more than one fault, by replacing the variable Ans with an array.

The method we have presented is highly dependent on the degree of grouping; therefore, on the particularities of the plant and the control sequences.

This is the reason why the most important disadvantage of the method we have presented above is that there may be situations (plants) in which the method will not bring major improvements or will not bring any improvements at all to the classical method.

This can happen especially in small processes with few components, or composed of several sub-processes between which the interaction is limited.

But for small processes the conventional method is not a challenge, since the amount of information and computing is not high.

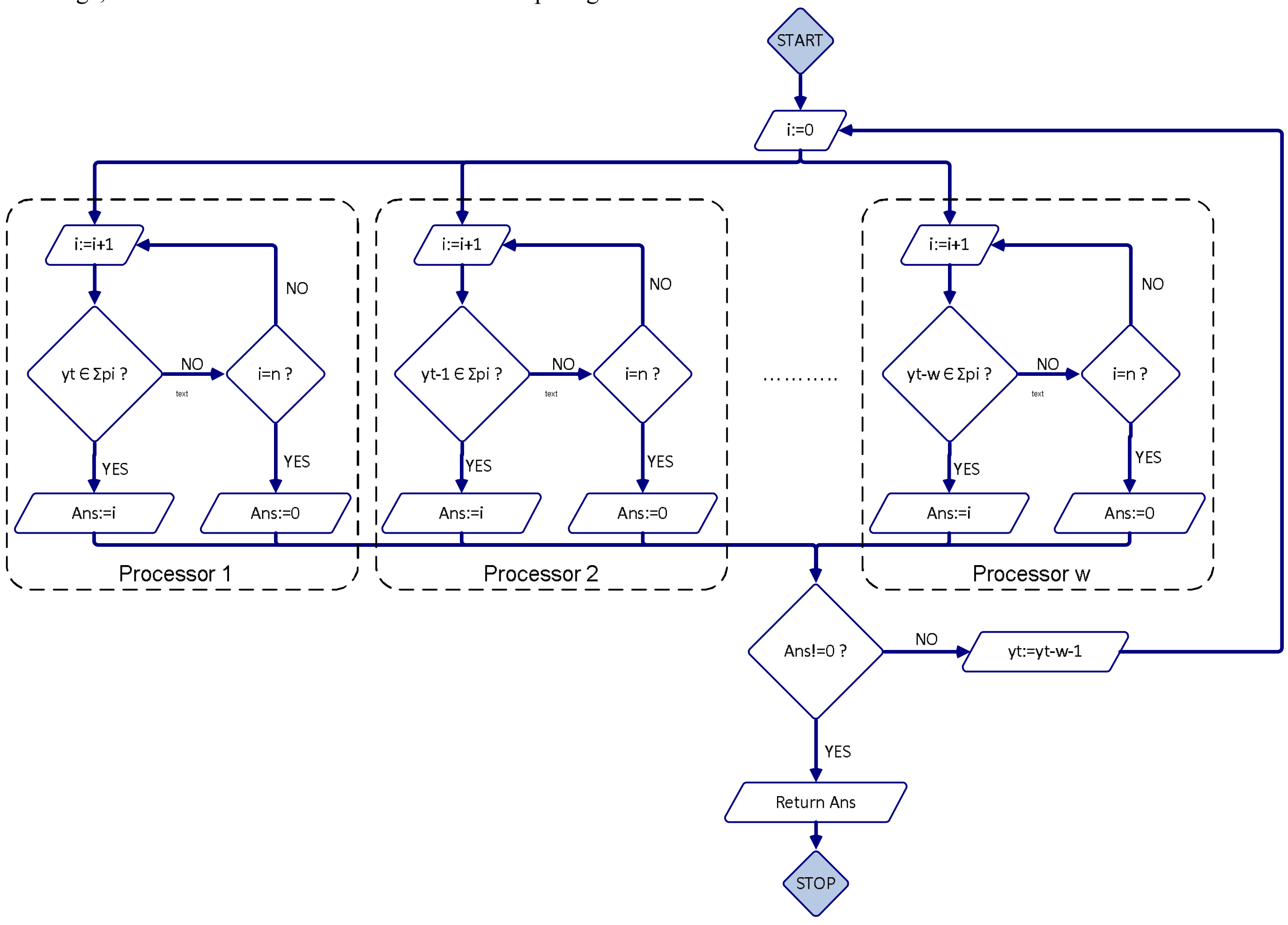

Fig. 3. Parallel algorithm for identifying the faulty component.

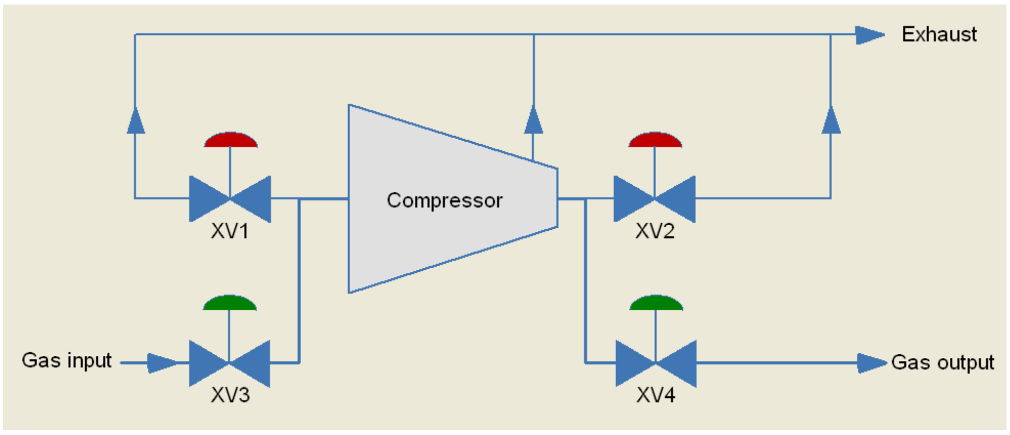

Fig. 4. Compressor station.

For processes that consist in several sub-processes, because of the limited interaction, it is easy to build separated diagnosers for each sub-process, returning in the situation above.

This is why the criteria we have presented do not limit the application of the method to a certain category of processes (plants), but represent the rules for defining the generalized plant model.

4. EXAMPLE OF IMPLEMENTING THE METHOD

In order to demonstrate the effectiveness of the method we presented above, we exemplified it on a small compressor station. The station is composed of one compressor, the gas and ventilation systems (Fig. 4).

The components and the operations of the considered system are the following:
The gas piping system was equipped with two valves:

- XV3 on the inlet;
- XV4 on the outlet.

The ventilation piping system was also fitted with two valves:

- XV1 to vent the inlet gas pipes;
- XV2 for venting the outlet gas pipes.

All valves are of the same type, dimensions and are provided with limit switches for monitoring their position in the control system.

The operation of the compressor station:
The initial state is the same as the safe state of the plant and is described as following:

- Compressor stopped;
- Valves XV1, XV2 – opened;
- Valves XV3, XV4 – closed.

The start-up procedure is composed of the following sequence:

- Close the valves from the ventilation circuit (XV1, XV2);
- Open the valves from the gas pipes (XV3, XV4);
- Start the compressor.

The station will be automatically shut down in the following situations:

- Any alarm (manual, fire alarm, gas leaks);
- Manually by the operator.

The shut down procedure consists in:

- Stop the compressor;
- Close the valves from the gas pipes (XV3, XV4);
- Open the valves from the ventilation circuit (XV1, XV2).

Using the classical diagnoser approach, we would construct the automaton for every component in order to achieve the plant model as parallel composition. Next, we would construct the control model and the system model. For our examples we used DESUMA/UMDES software developed from the University of Michigan, which permits the generation of the automata of the system (plant and control models) and the diagnoser automaton.

For the above example we define the plant Π having 5 components:

$$\Pi = \{P_1, P_2, P_3, P_4, P_5\}, \tag{45}$$

where:

$$\begin{aligned} &P_1 \text{ – is the automaton for XV1 - Fig. 5}\\ &P_2 \text{ – is the automaton for XV2 - Fig. 6}\\ &P_3 \text{ – is the automaton for XV3 - Fig. 7}\\ &P_4 \text{ – is the automaton for XV4 - Fig. 8}\\ &P_5 \text{ – is the automaton for the compressor - Fig. 9} \end{aligned} \tag{46}$$

According to (17) we have:

$$\begin{cases} P_1 = (X_1, \Sigma_1, \delta_1, x_{01}) \\ P_2 = (X_2, \Sigma_2, \delta_2, x_{02}) \\ P_3 = (X_3, \Sigma_3, \delta_3, x_{03}) \\ P_4 = (X_4, \Sigma_4, \delta_4, x_{04}) \\ P_5 = (X_5, \Sigma_5, \delta_5, x_{05}) \end{cases} \tag{47}$$

and:

$$\begin{aligned}
&X_1 = \{XV1\_O, XV1\_C, XV1\_SO, XV1\_SC,\\
&XV1\_C{:}so1\_XV1, XV1\_O{:}sc2\_XV1\}\\
&\Sigma_1 = \{ZS1\_C \to ZS1\_O, ZS1\_O \to ZS1\_C,\\
&(c\_XV1, ZS1\_c), (c\_XV1, ZS1\_o),\\
&(o\_XV1, ZS1\_c), (o\_XV1, ZS1\_o), sc1\_XV1,\\
&sc2\_XV1, so1\_XV1, so2\_XV1\}\\
&X_2 = \{XV2\_O, XV2\_C, XV2\_SO, XV2\_SC,\\
&XV2\_C{:}so1\_XV2, XV2\_O{:}sc2\_XV2\}\\
&\Sigma_2 = \{ZS2\_C \to ZS2\_O, ZS2\_O \to ZS2\_C,\\
&(c\_XV2, ZS2\_c), (c\_XV2, ZS2\_o),\\
&(o\_XV2, ZS2\_c), (o\_XV2, ZS2\_o\ ),\\
&sc1\_XV2, sc2\_XV2, so1\_XV2, so2\_XV2\}\\
&X_3 = \{XV3\_O, XV3\_C, XV3\_SO, XV3\_SC,\\
&XV3\_C{:}so1\_XV3, XV3\_O{:}sc2\_XV3\}\\
&\Sigma_3 = \{ZS3\_C \to ZS3\_O, ZS3\_O \to ZS3\_C,\\
&(c\_XV3, ZS3\_C), (c\_XV3, ZS3\_O),\\
&(o\_XV3, ZS3\_C), (o\_XV3, ZS3\_C),\\
&sc1\_XV3, sc2\_XV3, so1\_XV3, so2\_XV3\}\\
&X_4 = \{XV4\_O, XV4\_C, XV4\_SO, XV4\_SC\\
&, XV4\_C{:}so1\_XV4, XV4\_O{:}sc2\_XV4\}\\
&\Sigma_4 = \{ZS4\_C \to ZS4\_O, ZS4\_O \to ZS4\_C,\\
&(c\_XV4, ZS4\_C), (c\_XV4, ZS4\_O),\\
&(o\_XV4, ZS4\_C), (o\_XV4, ZS4\_C),\\
&sc1\_XV4, sc2\_XV4, so1\_XV4, so2\_XV4\}\\
&X_5 = \{COMP\_STARTED, COMP\_STOPPED\}\\
&\Sigma_5 = \{start\_comp, stop\_comp\}
\end{aligned} \tag{48}$$

The plant model is computed as the parallel composition of its components according to (9):

$$P = P_1 || \ P_2 || \ P_3 || P_4 || P_5. \tag{49}$$

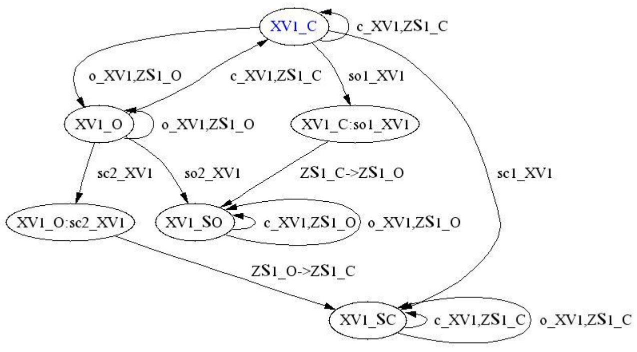

Fig. 5. The automaton for XV1.

We define the control model C_P and we compute the system model for P according to (30).

Now we can summarize: following the classical method we obtain:

- The plant model P having 2592 states;
- The control model C_P having 20 states;
- The system model S_P having 38 880 states.

We define the failure partition:

$$\Delta f_P = \{F_1, F_2, F_3, F_4, F_5\}, \qquad (50)$$

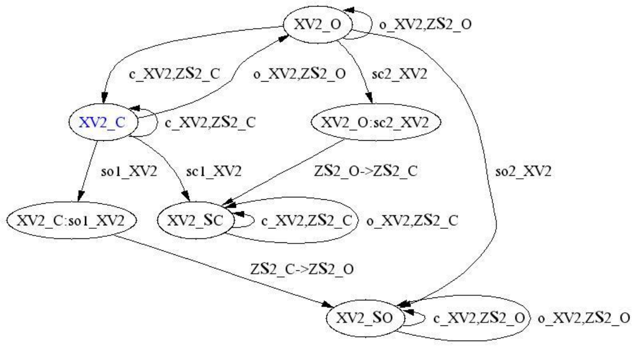

Fig. 6. The automaton for XV2.

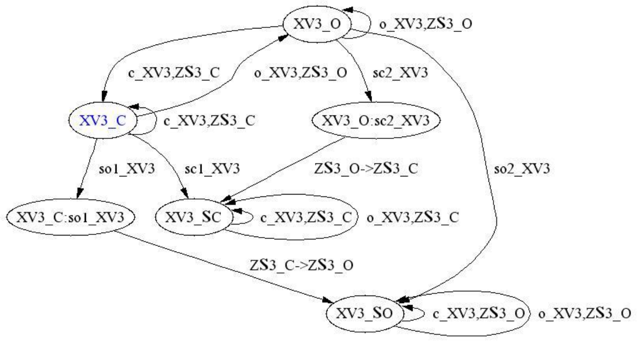

Fig. 7. The automaton for XV3.

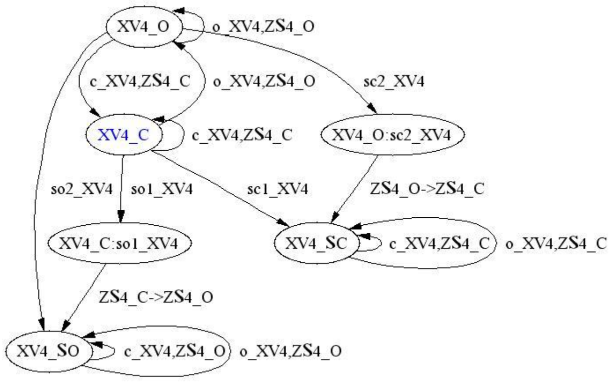

Fig. 8. The automaton for XV4.

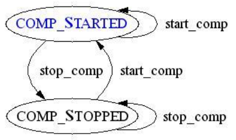

Fig. 9. The automaton for the compressor.

$$\begin{cases} F_1 = \{so1\_xv1, sc1\_xv1, so2\_xv1, sc2\_xv1\} \\ F_2 = \{so1\_xv2, sc1\_xv2, so2\_xv2, sc2\_xv2\} \\ F_3 = \{so1\_xv3, sc1\_xv3, so2\_xv3, sc2\_xv3\} \\ F_4 = \{so1\_xv4, sc1\_xv4, so2\_xv4, sc2\_xv4\} \\ F_5 = \Phi \end{cases} . \qquad (51)$$

We compute the diagnoser D_P with 38 385 states.

Using the method we have described in section III.2, with the following hypothesis:

- Both XV1 and XV2 will be either closed or opened at the same time;
- Both XV3 and XV4 will be either closed or opened at the same time;
- The compressor is ideal, it does not have faults;

It is possible to group the components in 3 types of components using AND function:

- The ventilation valves: XV1 and XV2;
- The valves on the gas circuit: XV3 and XV4;
- The compressor.

We define the following partition of $\prod$:

$$\Pi = \Pi_1 \cup \Pi_2 \cup \Pi_3, \qquad (52)$$

where:

$$\Pi_1 = \{P_1, P_2\} \text{ - type 1 - Fig. 11}$$

$$\Pi_2 = \{P_3, P_4\} \text{ - type 2 - Fig. 12} \qquad (53)$$

$$\Pi_3 = \{P_5\} \text{ - type 3 - Fig. 13}$$

For P_i defined in (47).

We construct the group automata G_1, G_2, G_3:

$$\begin{cases} G_1 = (X_{G1}, \Sigma_{G1}, \delta_{G1}, x_{0G1}) \\ G_2 = (X_{G2}, \Sigma_{G2}, \delta_{G2}, x_{0G2}), \\ G_3 = (X_{G3}, \Sigma_{G3}, \delta_{G3}, x_{0G3}) \end{cases} \qquad (54)$$

where:

$$\begin{aligned}
&X_{G1}=\{XV12\_O, XV12\_C, XV12\_SO, XV12\_SC,\\
&XV12\_C{:}sol\_XV12, XV12\_O{:}sc2\_XV12\}\\
&\Sigma_{G1}=\{ZS12\_C\rightarrow ZS12\_O, ZS12\_O\rightarrow ZS12\_C,\\
&(c\_XV12, ZS12\_C), (c\_XV12, ZS12\_O),\\
&(o\_XV12, ZS12\_C), (o\_XV12, ZS12\_C),\\
&sc1\_XV12, sc2\_XV12, sol\_XV12, so2\_XV12\}\\
&X_{G2}=\{XV34\_O, XV34\_C, XV34\_SO, XV34\_SC,\\
&XV34\_C{:}sol\_XV34, XV34\_O{:}sc2\_XV34\}\\
&\Sigma_{G2}=\{ZS34\_C\rightarrow ZS4\_O, ZS34\_O\rightarrow ZS34\_C,\\
&(c\_XV34, ZS34\_C), (c\_XV34, ZS34\_O),\\
&(o\_XV34, ZS34\_C), (o\_XV34, ZS34\_C),\\
&sc1\_XV34, sc2\_XV34, sol\_XV34, so2\_XV34\}\\
&X_{G3}=\{COMP\_STARTED, COMP\_STOPPED\}\\
&\Sigma_{G3}=\{start\_comp, stop\_comp\}
\end{aligned} \tag{55}$$

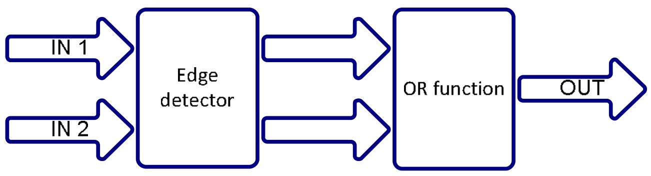

Fig. 10. Architecture of f_{XG1}, $f_{\Sigma G1}$ and f_G.

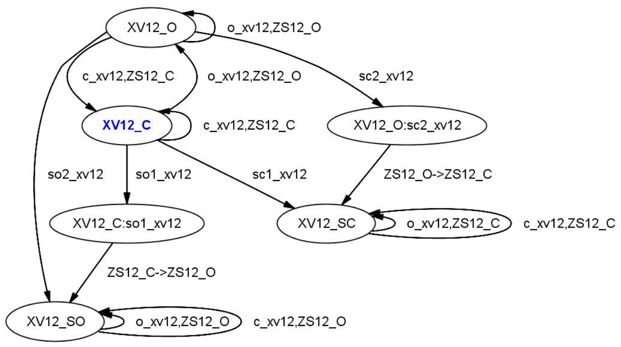

Fig. 11. G_1 automaton.

and:

$$\begin{aligned}
&XV12\_O=f_{XG1}(XV1\_O, XV2\_O)\\
&XV12\_C=f_{XG1}(XV1\_C, XV2\_C)\\
&XV12\_SO=f_{XG1}(XV1\_SO, XV2\_SO)\\
&XV12\_SC=f_{XG1}(XV1\_SC, XV2\_SC)\\
&XV12\_C{:}sol\_XV12=f_{XG1}(XV1\_C{:}sol\_XV1,\\
&XV2\_C{:}sol\_XV2)\\
&XV12\_O{:}sc2\_XV12=f_{XG1}(XV1\_O{:}sc2\_XV1,\\
&XV2\_O{:}sc2\_XV2)
\end{aligned} \tag{56}$$

$$\begin{aligned}
&ZS12\_C\rightarrow ZS12\_O=f_{\Sigma G1}(ZS1\_C\rightarrow ZS1\_O,\\
&ZS2\_C\rightarrow ZS2\_O)\\
&ZS12\_O\rightarrow ZS12\_C=f_{\Sigma G1}(ZS1\_O\rightarrow ZS1\_C,\\
&ZS2\_O\rightarrow ZS2\_C)\\
&(c\_XV12, ZS12\_C)=f_{\Sigma G1}((c\_XV1, ZS1\_C),\\
&(c\_XV2, ZS2\_C))\\
&(c\_XV12, ZS12\_O)=f_{\Sigma G1}((c\_XV1, ZS1\_O),\\
&(c\_XV2, ZS2\_O))\\
&(o\_XV12, ZS12\_C)=f_{\Sigma G1}((o\_XV1, ZS1\_C),\\
&(o\_XV2, ZS2\_C))\\
&(o\_XV12, ZS12\_C)=f_{\Sigma G1}((o\_XV1, ZS1\_C),\\
&(o\_XV2, ZS2\_C))\\
&sc1\_XV12=f_{\Sigma G1}(sc1\_XV1, sc1\_XV2)\\
&sc2\_XV12=f_{\Sigma G1}(sc2\_XV1, sc2\_XV2)\\
&sol\_XV12=f_{\Sigma G1}(sol\_XV1, sol\_XV2)\\
&so2\_XV12=f_{\Sigma G1}(so2\_XV1, so2\_XV2)
\end{aligned} \tag{57}$$

For f_{XG1} and $f_{\Sigma G1}$ we consider the following architecture (Fig. 10):

Similar we define X_{G2} and Σ_{G2}.

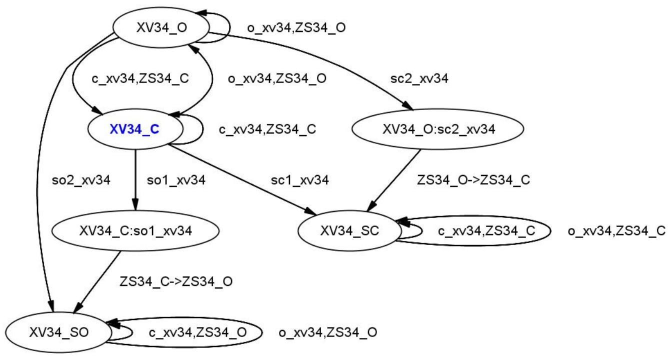

Fig. 12. G_2 automaton.

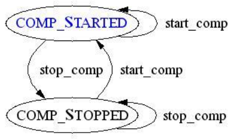

Fig. 13. G_3 automaton.

It is now possible to compute the group-based model of the plant G as a parallel composition of group models according to (23):

$$G=G_1||G_2||G_3. \tag{58}$$

We construct the group-based controller model (Fig. 14) and system model based on (27) and (31).

In this case:

- The plant model has 72 states;
- The control model has 12 states;
- The system model has 648 states.

We group F_i defined in (50):

$$\Delta f_P = F_{\Pi_1} \cup F_{\Pi_2} \cup F_{\Pi_3}, \tag{59}$$

where:

$$\begin{cases} F_{\Pi_1} = F_1 \cup F_2 \\ F_{\Pi_2} = F_3 \cup F_4. \\ F_{\Pi 3} = F_5 \end{cases} \tag{60}$$

We define Δf_G:

$$\Delta f_G = F_{G1} \cup F_{G2} \cup F_{G3}, \tag{61}$$

where:

$$\begin{cases} F_{G1} = \{so1\_xv12, so2\_xv12, sc1\_xv12, sc2\_xv12\} \\ F_{G2} = \{so1\_xv34, so2\_xv34, sc1\_xv34, sc2\_xv34\} \\ F_{G3} = \Phi \end{cases} \tag{62}$$

and:

$$\begin{aligned} & so1\_xv12 = f_G\,(so1\_xv1, so1\_xv2) \\ & so2\_xv12 = f_G\,(so2\_xv1, so2\_xv2) \\ & sc1\_xv12 = f_G\,(sc1\_xv1, sc1\_xv2) \\ & sc2\_xv12 = f_G\,(sc2\_xv1, sc2\_xv2) \\ & so1\_xv34 = f_G\,(so1\_xv3, so1\_xv4) \\ & so2\_xv34 = f_G\,(so2\_xv3, so2\_xv4) \\ & sc1\_xv34 = f_G\,(sc1\_xv3, sc1\_xv4) \\ & sc2\_xv34 = f_G\,(sc2\_xv3, sc2\_xv4) \end{aligned} . \tag{63}$$

We compute the diagnoser D_G for the group-based system model and the failure partition Δf_G having 861 states, unlike the initial one with 38 385 states.

The diagnoser D_G will act as a classical diagnoser but for the group-based system model, meaning it will isolate the group of faults (F_{Gi}). After isolating the group we use the algorithm presented in Fig. 2 to find the component within the group that has a fault.

For example, if the fault $so1\_xv1$ occurs, D_G will isolate F_{G1}, meaning we have a fault either in XV1 or XV2. Next, we compare the observable events that occurred (recorded by the monitoring and control system of the plant) with the sets of events Σ_1 and Σ_2. We will find that one of the latest observable event belongs to Σ_1 meaning we have a failure at XV1.

5. CONCLUSIONS

In this paper we have presented a new method based on grouping components for detecting and isolating faults in industrial automated systems. The method is based on the classical diagnoser approach, but it significantly reduces the number of states in the plant model, the dimension of the set of failure labels and consequently the number of states of the diagnoser. This is very important in large applications where the computing and storage capacity should be high for the classical method.

The applicability and efficiency of the method are process dependent, meaning that it depends on the structure and functioning of the automated process how much the number of states is reduced and how effective the method can be.

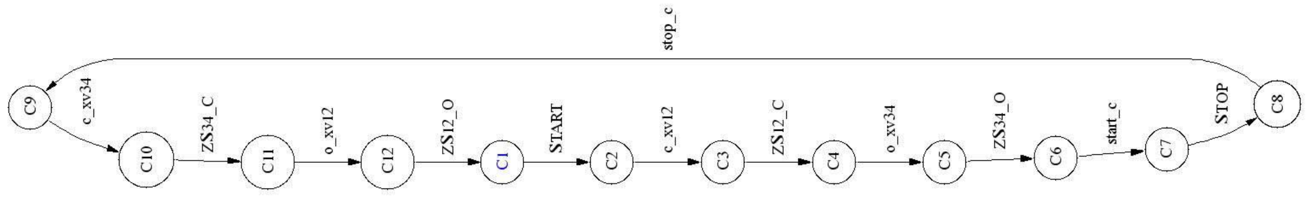

Fig. 14. The group-based controller model.

REFERENCES

Alexandru M. and Popescu D. (2001). A Robust Fault Detection and Isolation of A DC Motor, *Journal of Control Engineering and Applied Informatics*, Vol.3, No.2 pp. 28-436;

Athanasopoulou E., Lingxi L., and Hadjicostis C.N. (2006). Probabilistic failure diagnosis in finite state machines under unreliable observations, In *Proc. of 2006 8th International Workshop on Discrete Event Systems*, pp. 301 – 306;

Ferrarini L., Allevi M., and Dede A. (2011). A Methodology for Fault Isolation and Identification in Automated Equipments, *9th IEEE International Conference on Industrial Informatics*, pp. 157 – 162;

Ferrarini L., Allevi M., and Dede A. (2011). Implementation and testing of an online fault isolation methodology in a real industrial scenario, *3rd International Workshop on Dependable Control of Discrete Systems* (DCDS), pp. 13 – 18;

Ferrarini L., Allevi M., and Dede A. (2011). A real-time algorithm for fault identification in machining centres. *IFAC2011*, pp. 5201-5206;

Gascard E. and Simeu-Abazi Z. (2011). Automatic Construction of Diagnoser for Complex Discrete Event Systems, *3rd International workshop on Dependable Control of Discrete systems*, pp. 90 - 95;

Gertler J. (1988). Survey of Model-Based Failure Detection and Isolation in Complex Plants, *IEEE Control Systems Magazine*, 8(6), pp. 3-11;

Lunze J. and Schroder J. (2001). State observation and diagnosis of discreteevent systems described by stochastic automata, Discrete Event Dyna. *Syst.: Theory Appl.,* pp. 319–369;

Pencolé Y., Kamenetsky D. and Schumann A. (2006). Towards Low-Cost Fault Diagnosis in Large Component-Based Systems, *6th IFAC Symposium on Fault Detection*, Supervision and Safety of Technical Processes, pp. 1473-1478;

Ribot P., Pencol´e Y. and Combacau M. (2009). Diagnosis and prognosis for the maintenance of complex systems, *IEEE International Conference on Systems*, Man and Cybernetics, pp. 4146 – 4151;

Rincon A. (2012). Multiple fault detection and diagnosis in a Gas Turbine using principal component analysis and structured residuals. *20th Mediterranean Conference on Control & Automation* (MED), pp. 91 - 97;

Rughinis R. and Gheorghe L. (2013). TinyAFD: Attack and Fault Detection Framework for Wireless Sensor Networks, *Journal of Control Engineering and Applied Informatics,* Vol.15, No.1 pp. 33-44;

Sampath M., Sengupta R., Lafortune S., Sinaamohideen K. and Teneketzis D. (1995). Diagnosability of discrete event systems, *IEEE Transactions on Automatic Control, 40(9)*, pp. 1555–1575;

Seungjoo L., and Dawn T. (2005). A modular control design method for a flexible manufacturing cell including error handling, *44th IEEE Conference on Decision and Control*, 2005 and 2005 European Control Conference. CDC-ECC '05., pp. 8355 – 8360;

Thorsley D. and Teneketzis D. (2005). Diagnosability of stochastic discreteevent systems, *IEEE Trans. Autom. Control,* pp. 476– 492;

Throsley D., Yoo T., and Garcia H. E. (2008). Diagnosability of stochastic discrete event systems under unreliable observations, In *Proceedings of the American Control Conference*, pp. 1158 – 1165;

University of Michigan (2008). DESUMA/UMDES, http://www.eecs.umich.edu/umdes/toolboxes.html.

Wen-Chiao L., Garcia H. E., Thorsley D., and Yoo T. (2009). Sequential Window Diagnoser for Discrete - Event Systems Under Unreliable Observations, Allerton'09 *Proceedings of the 47th annual Allerton conference on Communication, control, and computing*, pp. 668-675;

Willsky A. S. (1976). A Survey of Design Methods for failure Detection in Dynamic Systems (vol. 12) *Automatica, 12*, pp. 601-611;

Yurish S. Y. (2010). Sensors: Smart vs. Intelligent, *Sensors & Transducers Journal* Vol.114, pp. 4;

Zhu D., Bai J. and Yang S. X. (2010). A Multi-Fault Diagnosis Method for Sensor Systems Based on Principle Component Analysis, *Sensors*, pp. 241-253;