A decentralized model-based diagnostic tool for complex systems

Yannick Pencolé, Marie-Odile Cordier, Laurence Rozé
IRISA, 35042 Rennes, FRANCE
{ypencole,cordier,roze}@irisa.fr

Abstract

We address the problem of diagnosing complex discrete-event systems such as telecommunication networks¹. Given a flow of observations from the system, the goal is to explain those observations by identifying and localizing possible faults. Several model-based diagnosis approaches deal with this problem but they need the computation of a global model which is not feasible for complex systems like telecommunication networks. Our contribution is the proposal of a decentralized approach which permits to carry out an on-line diagnosis without computing the global model. This paper describes the implementation of a tool based on this approach. Given a decentralized model of the system and a flow of observations, the program analyzes the flow and computes the diagnosis in a decentralized way. We also present experimental results based on a real system.

1. Introduction

The problem we deal with is the supervision of complex and large discrete-event systems such as telecommunication networks. Given a supervisor continuously receiving observations (alarms) sent by the system components, our purpose is to help operators to identify failures.

Two classical approaches in monitoring such systems are knowledge-based techniques that directly associate a diagnosis to a set of symptoms (for example expert systems [9] or scenario recognition systems [7]), and model-based techniques which rely on a behavioral model of the system ([12]). The main weakness of the first approach is the lack of genericity, as the network changes, a new expertise has to be acquired. We therefore decided to use model-based techniques which are recognized to be more adapted to evolutive systems than expertise-based approaches are and focus in the following on these ones.

¹This work is partially supported by RNRT (French National Research in Telecommunication) through the MAGDA project (Modeling and Learning for Distributed Management of Alarms). For more information see <http://magda.elibel.tm.fr/>.

In the literature, more and more approaches are proposed for diagnosing discrete-event systems, in both AI and control engineering domain. It concerns continuous-variable systems which, after quantization, are represented as discrete systems as well as “discrete by nature” systems such as communicating processes which exchange messages and alarms.

The majority of these approaches are centralized approaches [14, 8, 13]. [15] has proposed the diagnoser approach which consists in the compilation of diagnostic information in a data structure (called *diagnoser*) which maps failures and observations for on-line diagnosis. As telecommunication networks are concerned, [13] proposes an extension of this approach which is well-adapted for on-line diagnosis. Nevertheless, a centralized approach needs to have a global information about the system which is unrealistic for complex and large systems like telecommunication networks. Moreover, such systems are naturally distributed so it is easier to model those systems with decentralized information. In this way, [6] proposes an approach for diagnosing discrete-event systems using decentralized diagnosers. But in this approach, the computation of one decentralized diagnoser needs a global model which is a major drawback of this method.

[2] and [4] propose methods which rely on a decentralized model, but these methods are used *off-line* to solve a diagnosis problem *a posteriori*.

Our contribution is to propose an approach which relies on a decentralized model and provides on-line diagnosis of large discrete-event systems such as telecommunication networks. The idea is to split the flow of observations into temporal windows. For each temporal window, we compute a diagnosis for each component of the system (*local diagnosis*) and then we build a diagnosis of the whole system (*global diagnosis*) by merging these local diagnoses and the diagnosis of previous temporal windows. The challenge is to preserve the efficiency of the global computation which depends mainly on the merging operation of local diagnoses [10][11].

In this paper, we present a tool which implements this approach and some experimental results based on a real ap-

plication: a French packet-switching network.

The paper is organized as follows. We first introduce what kind of systems are considered and the problem of monitoring. Then, we present the formalism based on communicating automata, which is used to represent in a decentralized way the model of the system. In section 3, the diagnostic task is explained by defining observations and diagnoses. Then, the merging operation allowing to build step by step the global diagnosis is presented. We then focus on experimental results by presenting the studied system, and the diagnosis platform which monitors it.

2. Decentralized model of the system

The systems we consider are distributed systems composed of *components* which interact each other (see figure 1). Typical components are switches, multiplexers, control stations... Each component can emit some *observations* (also called *alarms*) when a problem occurs on the network. A *supervision center* is in charge of monitoring the network by receiving the alarms. For most of the faults, there are some automatic recovery procedures so that faults are in general not permanent.

Figure 1. Supervision of a telecommunication network.

The task of the supervision operator is to interpret received alarms in order to determine if an intervention is needed or not. The problem is that the interpretation of the received alarms is difficult because of the important number of the received alarms (thousands of alarms). Thus, automatic diagnostic methods are needed.

As said before, dealing with a global model of such systems is unrealistic. This section explains how the model of the system is described in a decentralized way by means of local models, which describe the behaviors of each compo-

nent of the system and the interactions between them. The formalism chosen for such models is that of communicating automata.

Each component can react to failure events by changing of states and emitting observable events to the supervisor. They interact by exchanging messages (named *internal events*) which occur for instance when failures propagate through the system.

The supervisor receives alarms from each component of the system via a communication channel. Each alarm is associated to the component which sent it. We suppose in the following that the communication channels are FIFO data structures. However, communication channels from different components are not synchronized (their propagation delays can be different). We make the following hypotheses.

Hypothesis 1 *Let o_1 and o_2 be observable events emitted by one component. We assume that o_1 and o_2 are observed by the supervisor in the order of their emission by the component.*

Hypothesis 2 *Let o_1 and o_2 be observable events emitted by two different components. We assume that o_1 and o_2 may not be received by the supervisor in the order of their emission by the components.*

A *component* is faced to two kinds of received events: failure events (Σ_{fail}^i) and internal events ($\Sigma_{intreceived}^i$). A component emits two kinds of events: observable events (Σ_{obs}^i) via its communication channel and internal events ($\Sigma_{intemitted}^i$).

Definition 1 (Model of a component) *A component behavior is described by a communicating finite-state machine $\Gamma_i = (\Sigma_{in}^i, 2^{\Sigma_{out}^i}, Q_i, E_i)$ where*

- Σ_{in}^i *is the set of input events* ($\Sigma_{in}^i = \Sigma_{fail}^i \cup \Sigma_{intreceived}^i$);
- Σ_{out}^i *is the set of output events* ($\Sigma_{out}^i = \Sigma_{obs}^i \cup \Sigma_{intemitted}^i$);
- Q_i *is the set of states of the component;*
- $E_i \subseteq (Q_i \times \Sigma_{in}^i \times 2^{\Sigma_{out}^i} \times Q_i)$ *is the set of transitions.*

A model of component is depicted on the figure 2. Two kinds of failure events can happen on this component: $f1$ and $f2$. For example, if $f1$ occurs on the component at state 1, then the component emits the $a1$ alarm, propagates the failure by the emission of the $i2$ event towards another component and goes to state 2.

The model of the system is described in a decentralized way by the models of its components.

Figure 2. A model of component: $f1$ and $f2$ are failure events, $i1$ and $i2$ are events received from/emitted towards another component, $a1$ and $a2$ are alarms.

Definition 2 (Model of a system) The model Γ of a system is given by the set of models of its components $\{\Gamma_1, \dots, \Gamma_n\}$, a set of failure events (Σ_{fail}), a set of observable events (Σ_{obs}) and a set of internal events (Σ_{int}) such that:

- $\{\Sigma_{obs}^1, \dots, \Sigma_{obs}^n\}$ is a partition of Σ_{obs} ;
- $\{\Sigma_{fail}^1, \dots, \Sigma_{fail}^n\}$ is a partition of Σ_{fail} ;
- $\{\Sigma_{intreceived}^1, \dots, \Sigma_{intreceived}^n\}$ and $\{\Sigma_{intemitted}^1, \dots, \Sigma_{intemitted}^n\}$ are partitions of Σ_{int} ;
- $\forall e \in \Sigma_{int}, \exists! \Gamma_i | e \in \Sigma_{intreceived}^i \wedge \exists! \Gamma_j | e \in \Sigma_{intemitted}^j \wedge i \neq j$.

The global model of the system could be explicitly built by composing the automata of its components (via a classical operation of synchronization on the internal events [1]) but it is exactly what we want to avoid due to the important size of such a model for large systems.

3. Decentralized diagnosis task

The diagnostic task consists of inferring information on the faulty state of the system components and/or on the occurrence of faulty events from the sequence of observations (in our case the sequence of alarms received by the supervisor). As many authors [2, 5, 3], we consider that the diagnostic task consists, for dynamical systems, in finding the trajectories (sequences of states and transitions) explaining the observations. From these trajectories, it is in general easy to extract the diagnosis itself. A simple case is when you are only interested in the set of possible faulty components but in a monitoring context, the operator is often interested in getting a more thorough explanation of what happened, as at least the sequences of exogenous failure events explaining the set of received alarms and the possible current states of the system.

The idea is to use a decentralized diagnosis approach by firstly computing a diagnosis for each component (*local diagnosis*) and then building a diagnosis of the whole system (*global diagnosis*) from these local diagnoses. In the following, we formally define these notions and explain how the global diagnosis is computed by *composing* the local ones.

In the following, we consider the set of observation sequences $\mathcal{O} = \{\mathcal{O}_{\Gamma_1}, \dots, \mathcal{O}_{\Gamma_n}\}$, $\mathcal{O}_{\Gamma_i} \in (\Sigma_{obs}^i)^*$ where each $\mathcal{O}_{\Gamma_i}$ is the sequence of observations received from the component Γ_i during a temporal window Δt^2 . At the beginning of the temporal window, the initial states $X_{init}^{\Gamma_i}$ of Γ_i are known and described by the initial states of each component $\{X_{init}^{\Gamma_1}, \dots, X_{init}^{\Gamma_n}\}$.

3.1. Local diagnosis

Given the model of the component Γ_i and a set of initial states $X_{init}^{\Gamma_i}$, a *local diagnosis* Δ_{Γ_i} describes the subset of trajectories from Γ_i starting from elements of $X_{init}^{\Gamma_i}$ and explaining the sequence of *local* observations $\mathcal{O}_{\Gamma_i}$. In other words, if the diagnosed component follows one of those trajectories then the sequence of emitted observable events according to the model Γ_i is exactly $\mathcal{O}_{\Gamma_i}$.

So we propose to represent a local diagnosis as a communicating finite-state machine $\Delta_{\Gamma_i}(X_{init}^{\Gamma_i}, \mathcal{O}_{\Gamma_i})$, shortly Δ_{Γ_i} . Compared to the automaton Γ_i , the main syntactical difference is that each state Q_{Δ_i} of this automaton is associated to a pair $(s_{\Gamma_i}, \mathcal{E}_i)$ where $s_{\Gamma_i} \in Q_i$ is a state of the component and $\mathcal{E}_i$ is the prefix subsequence of $\mathcal{O}_{\Gamma_i}$ explained in this state (see figure 3).

Figure 3. A local diagnosis of the component presented in figure 2. This diagnosis explains the sequence of alarms $a1a2$ from the state 1. It contains one trajectory.

The initial states of Δ_{Γ_i} are those corresponding to $X_{init}^{\Gamma_i}$ and the final states are those such that $\mathcal{E}_i = \mathcal{O}_{\Gamma_i}$. So the final states represent the current states of the component explaining the whole sequence of local observations. As in Γ_i , the transitions are labeled with failures or internal received events as input and with observed or internal emitted events as output.

²The way successive temporal windows are incrementally taken into account is briefly commented in the conclusion.

Definition 3 (Local diagnosis) The local diagnosis $\Delta_{\Gamma_i}(X_{init}^{\Gamma_i}, \mathcal{O}_{\Gamma_i})$ of Γ_i according to the sequence $\mathcal{O}_{\Gamma_i}$ is a finite-state machine: $(\Sigma_{in}^i, 2^{(\Sigma_{out}^i)}, Q_{\Delta_i}, E_{\Delta_i})$ where

- Σ_{in}^i is the set of input events ($\Sigma_{in}^i = \Sigma_{fail}^i \cup \Sigma_{intreceived}^i$);
- Σ_{out}^i is the set of output events ($\Sigma_{out}^i = \Sigma_{obs}^i \cup \Sigma_{intemitted}^i$);
- Q_{Δ_i} is the set of states ($Q_{\Delta_i} \subseteq Q_i \times (\Sigma_{obs}^i)^*$);
- $E_{\Delta_i} \subseteq (Q_{\Delta_i} \times \Sigma_{in}^i \times 2^{(\Sigma_{out}^i)} \times Q_{\Delta_i})$ is the set of transitions.

Thus, a local diagnosis Δ_{Γ_i} is computed, given the local model Γ_i , a set of initial states X_{init}^i of Γ_i and a sequence of local observations. There exist several algorithms for computing such a diagnosis. Basically, they are based on a transition path search algorithm (depth-first search, breadth-first search...) from the states of X_{init}^i in the model Γ_i . The solution is the set of paths such that the sequence of observable events emitted by one path corresponds exactly to the sequence $\mathcal{O}_{\Gamma_i}$. In our tool, the algorithm of the local diagnosis computation is based on a diagnoser approach [15][10]. A local diagnoser is a data structure which is more efficient to compute local diagnoses: it consists in a compilation of the model of a component in order to efficiently map the local observable events with the local trajectories which explain them.

3.2. Merging local diagnoses to get the global diagnosis

We have defined local diagnoses as finite-state machines which represent the possible trajectories of components according to the local observations. The problem of local diagnoses is that they do not take into account observations from the other components. A local diagnosis supposes the happening of interactions between components (emission or reception of internal events), but those interactions have to be validated.

Given the model of the system Γ and a set of initial states X_{init}^{Γ} , we define the global diagnosis as the set of all the *global trajectories* starting from elements of X_{init}^{Γ} and explaining the observations $\mathcal{O} = \{\mathcal{O}_{\Gamma_1}, \dots, \mathcal{O}_{\Gamma_n}\}$ received by the supervisor. Such a trajectory results from a set of local trajectories (one per component) where the interactions are validated.

Like a local diagnosis, we represent the global diagnosis as a communicating finite-state machine $\Delta_{\Gamma}(X_{init}^{\Gamma}, \mathcal{O})$, shortly Δ_{Γ} . The states Q_{Δ} of this automaton are pairs $(s_{\Gamma}, \mathcal{E})$ where $s_{\Gamma} \in Q_1 \times \dots \times Q_n$ is a state of the system and $\mathcal{E} = \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ where $\mathcal{E}_i$ is the prefix of the

sequence $\mathcal{O}_{\Gamma_i}$ explained in this state. The initial states of Δ_{Γ} are those corresponding to X_{init}^{Γ} and the final states are such that $\mathcal{E} = \mathcal{O}$. So the final states represent the current states of the system explaining the whole sequence of observations. The transitions are labeled with failure events as input ($\Sigma_{in} = \Sigma_{fail}$) and with observed events as output ($\Sigma_{out} = \Sigma_{obs}$).

Definition 4 (Global diagnosis) The global diagnosis $\Delta_{\Gamma}(X_{init}^{\Gamma}, \mathcal{O})$ of Γ according to the set of sequences $\mathcal{O}$ is a finite-state machine: $(\Sigma_{in}, 2^{(\Sigma_{out})}, Q_{\Delta}, E_{\Delta})$ where

- Σ_{in} is the set of input events ($\Sigma_{in} = \Sigma_{fail}$);
- Σ_{out} is the set of output events ($\Sigma_{out} = \Sigma_{obs}$);
- Q_{Δ} is the set of states of the diagnosis;
- $E_{\Delta} \subseteq (Q_{\Delta} \times \Sigma_{in} \times 2^{(\Sigma_{out})} \times Q_{\Delta})$ is the set of transitions.

In a decentralized approach, the model Γ of the whole system is not explicitly given but defined as the set of its component models. The global diagnosis is built by composing all the local diagnoses:

$$\Delta_{\Gamma}(X_{init}^{\Gamma}, \mathcal{O}) = \bigodot_{i=1}^n \Delta_{\Gamma_i}(X_{init}^{\Gamma_i}, \mathcal{O}_{\Gamma_i})$$

where $\odot$ is the classical composition operation between two communicating finite-state machines [1] synchronized on the internal events exchanged between the local diagnoses [10]. The principle of the composition of diagnoses is given on figure 4.

The interesting point in the diagnosis definition given above is that the composition operation of communicating automata is commutative and associative. The global diagnosis can be computed in several ways according to the order in which the diagnoses are progressively merged (composed). The efficiency of the global computation depends directly on this key point. Moreover, intermediate compositions also result in some diagnoses. In the figure 4, the composition $\Delta_{\Gamma_l} \odot \Delta_{\Gamma_m}$ represents the set of all the trajectories of Γ_l and Γ_m explaining the observations from Γ_l and Γ_m under the hypothesis that the interactions with other components are satisfied. In particular, if no interaction with other components are claimed in those trajectories, then $\Delta_{\Gamma_l} \odot \Delta_{\Gamma_m}$ represents all the trajectories of Γ_l and Γ_m which explain the whole set of observations $\mathcal{O}$.

3.3. Strategy for the merging operation

In this section, we consider the problem of efficiently computing the global diagnosis by optimizing the cost of the composition.

Figure 4. Principle of the composition of two diagnoses: synchronization on the internal events. Here, Δ_{Γ_l} claims that $i2$ is exchanged between Γ_l and Γ_m . Δ_{Γ_m} confirms this interaction. $\Delta_{\Gamma_l} \odot \Delta_{\Gamma_m}$ is the diagnosis of the system composed by $\{\Gamma_l, \Gamma_m\}$.

3.3.1 Basic idea

The tool implements a strategy based on interactions between components which are claimed by their local diagnoses [10]. Each local diagnosis Δ_{Γ_i} contains trajectories which claim that the diagnosed component Γ_i has interacted with other components by sending or receiving internal events. Suppose that Δ_{Γ_i} claims that Γ_i emits an event $e_{i \rightarrow j}$ towards Γ_j . If Δ_{Γ_j} does not contain any trajectories which claim the reception of $e_{i \rightarrow j}$ then trajectories from Δ_{Γ_i} claiming the emission of $e_{i \rightarrow j}$ are *inconsistent* - they cannot participate to a global trajectory of the system. The composition of the local diagnoses consists in detecting such inconsistencies.

3.3.2 Principles

The merging operation uses the composition ($\odot$ operation). As said before, this composition operation is commutative and associative, so we can compose diagnoses in many different ways. To improve the efficiency, we will apply the two following principles.

1. *Detecting and eliminating inconsistent trajectories.* Inconsistent trajectories uselessly increase the cost of the composition operation. The tool proposes to detect and eliminate some of them before any composition operation. The idea is to compute the sets of exchanged events $\{E_i\}$ claimed by each local diagnosis. Each event e must belong to a pair of distinct sets (E_i, E_j) corresponding to the emission and the reception of e . If it is not the case, then every trajectory of the local diagnoses which supposes the exchange of e is inconsistent and eliminated.
2. *Composing local diagnoses of interacting components.* The merging of local diagnoses is needed in order

to check the interactions which are claimed by them. Thus, merging two diagnoses which do not claim any interaction between their respective components is not interesting: we want to avoid this kind of computation. Relying on the sets of events $\{E_i\}$, the sets of components which interact each other is computed. Then, the merging operation selects a partition of local diagnoses involving interacting components. We privilege a partition of diagnoses such that:

- a partition element has two diagnoses at the most;
- selected diagnoses are such that the set of exchanged events claimed by those diagnoses is as big as possible.

Once a partition of diagnoses is chosen, the diagnoses of each element of the partition are composed in a parallel way. A new set of diagnoses is obtained where one diagnosis is associated to each element of the partition. The set of possible exchanged events is updated according to the new diagnoses set. Then, a new stage proceeds by choosing a new partition of diagnoses and composing it. The last stage produces a set of diagnoses which claim that there is no possible exchanged events between them. Each diagnosis of the resulting set corresponds to a distinct group of components in the system and implicitly represents the global diagnosis. Because those diagnoses suppose that there are no exchanged events between components of two different groups, the composition is unnecessary. The trajectories of every diagnosis of the resulted set participate to a global trajectory: they are consistent.

4. Experimental results

4.1. Telecommunication network

We have tested the tool on a subpart of a real packet-switching telecommunication network (see figure 5). It is composed of eight switches (*Sw*) which are managed by two different technical centers (*Tc*). Each switch and each technical center can emit some alarms. In this architecture, alarms from switches are emitted via their technical center which means that if the technical center is down, the alarms emitted by the switches are lost: this phenomenon is called *masking phenomenon*. It is expressed in the model of the network.

Figure 5. Supervised telecommunication network.

In this example, we suppose that a technical center may emit two kinds of alarms: *cvhs* and *cves*. A switch may also emit two kinds of alarms: *n003* and *n004*. *cvhs* is emitted when the technical center goes down, due to a rebooting or a connection cut. *cves* is emitted when the technical center begins to work well again. *n003* is due to a blocking of the switch or a booting ordered by its technical center and *n004* is emitted when the switch begins to work well again. The difficulty of this system is that one alarm does not permit to identify a failure (for example, *cvhs* may be explained by a rebooting or a connection cut). Nevertheless, discrimination is possible by looking at the occurrence of other alarms (for example, the rebooting of a technical center provokes the rebooting of switches which emit alarms).

4.2. Diagnosis platform

The diagnosis platform we used for diagnosing such a system is a distributed application (see figure 6).

The tool is based on a CORBA architecture. We developed four kinds of objects :

- **Local diagnoser.** There is one instantiation of a local diagnoser per component of the system. The purpose of this object is to look at the network and to capture the alarms emitted by the associated component. Then,

Figure 6. Distributed diagnosis platform.

the local diagnoser updates the local diagnosis of the component.

- **Coordinator.** The coordinator is unique. Its task is to compute the strategy of merging by taking into account the current interactions claimed by each local diagnoser. Then, it calls the composer objects which apply the strategy.
- **Composer.** The task of a composer is to apply the $\odot$ composition operation. According to the coordinator, it gets the diagnoses from the local diagnosticians or other composers and applies the composition.
- **Interface.** This object is the interface of the tool. It permits to visualize the current diagnoses of the system and to display them to the operator.

There are several advantages of using a distributed architecture. Firstly, we can deploy each local diagnoser on a machine well-adapted for the observation of a given component (machine close to the component...). Secondly, the use of several composers permits to compose the diagnoses in a real parallel way (one composer per processor) and increases the efficiency of the merging operation.

4.3. Results

In this experiment, we have computed the global diagnosis in a temporal window containing 16 alarms (see table 2). The supervision center receives *cvhs* from *Tc1*, the sequence *n003 n004 n003* from *Sw1*...

For this experiment, the diagnosis platform uses two composer objects in order to parallelize some computations. The number of composers to instantiate depends of course

of the complexity of the diagnosed systems and on the number of processors we can use for the diagnosis computation.

As the local diagnoses computation is concerned, dealing with this set of observations is very efficient. In fact, the time of computation is inferior to 400ms for each local diagnosis. This efficiency is due to the diagnoser approach implemented in the tool.

The table 1 presents the strategy the diagnostic system has chosen, following the principles depicted in section 3.3.2. In this experiment, the interactions can only exist between a switch and its technical center. Thus, the strategy forbids the composition of diagnoses which only concern switches. In order to get the global result, 4 stages of compositions are needed. In stage 4, the diagnoses $\Delta_{Tc1,Sw1,Sw2,Sw3,Sw4}$ and $\Delta_{Tc2,Sw5,Sw6,Sw7,Sw8}$ are computed; because there is no interaction between them (they are *independent*), the diagnostic tool does not need to compose them.

Tc1	<i>cvhs</i>
Sw1	<i>n003 n004 n003</i>
Sw2	<i>n003</i>
Sw3	
Sw4	<i>n003 n004</i>
Tc2	<i>cvhs cves</i>
Sw5	
Sw6	<i>n004 n003 n004</i>
Sw7	<i>n004</i>
Sw8	<i>n004 n003 n004</i>

Table 2. The set of alarm sequences observed during one temporal window.

Due to unobservable cycles of faults that can occur on the system, the number of global trajectories is infinite. Nevertheless, because of our representation of diagnosis by a finite-state machine, it is possible to represent all these trajectories. The diagnosis $\Delta_{Tc1,Sw1,Sw2,Sw3,Sw4}$ contains 280 states and 2128 transitions and the diagnosis $\Delta_{Tc2,Sw5,Sw6,Sw7,Sw8}$ contains 833 states and 5326 transitions. The result is obtained in 7 seconds using two computers (UltraSparc 10, 440Mhz). $\Delta_{Tc1,Sw1,Sw2,Sw3,Sw4}$ indicates in particular that all the *n003* alarms emitted by *Sw1,Sw2,Sw4* are due to a blocking of the switches and not a rebooting and $\Delta_{Tc2,Sw5,Sw6,Sw7,Sw8}$ indicates that *Tc2* has rebooted.

4.4. Comparisons with other strategies

In order to show the efficiency of our strategy, we have compared it to other strategies of computations in the same

context of experimentation. Times of computations are presented in the figure 7. For each stage of merging, the figure presents the cumulative time since the first stage. For a better comparison of strategies, the measures do not consider parallelism. It is the reason why 8 stages are needed as opposed to the strategy presented in section 4.3.

Figure 7. Comparisons of the performance of four different strategies of merging.

Strategy 1 is the strategy used by the diagnostic tool (see section 3.3.2) when one composer is available. Strategy 2 does not apply any merging strategy based on interacting diagnoses. Strategy 3 is the same as the strategy 1 except that inconsistent trajectories are not eliminated before any composition. Strategy 4 is a naive strategy: no elimination, no merging strategy. It shows the cumulative effects of strategies 2 and 3.

5. Conclusion

This paper describes a model-based diagnostic tool especially adapted to large and evolutive discrete-event systems such as telecommunication networks. Dealing with global models is clearly impossible with such systems. The idea is to compute local diagnoses from component models and to build the global diagnosis by merging these local results. The challenge is to preserve the efficiency of this global on-line computation. One key point is considering with this respect: the order in which the successive merging operations are performed. Our tool implements a strategy favoring the diagnoses corresponding to interacting components. This strategy can be compared to the reconstruction plan proposed in [2]. The main advantage of our proposal

	Composer 1	Composer 2
Stage 1	$\Delta_{Tc1} \odot \Delta_{Sw1}$ $\rightarrow \Delta_{Tc1,Sw1}$	$\Delta_{Tc2} \odot \Delta_{Sw5}$ $\rightarrow \Delta_{Tc2,Sw5}$
Stage 2	$\Delta_{Tc1,Sw1} \odot \Delta_{Sw2}$ $\rightarrow \Delta_{Tc1,Sw1,Sw2}$	$\Delta_{Tc2,Sw5} \odot \Delta_{Sw6}$ $\rightarrow \Delta_{Tc2,Sw5,Sw6}$
Stage 3	$\Delta_{Tc1,Sw1,Sw2} \odot \Delta_{Sw3}$ $\rightarrow \Delta_{Tc1,Sw1,Sw2,Sw3}$	$\Delta_{Tc2,Sw5,Sw6} \odot \Delta_{Sw7}$ $\rightarrow \Delta_{Tc2,Sw5,Sw6,Sw7}$
Stage 4	$\Delta_{Tc1,Sw1,Sw2,Sw3} \odot \Delta_{Sw4}$ $\rightarrow \Delta_{Tc1,Sw1,Sw2,Sw3,Sw4}$	$\Delta_{Tc2,Sw5,Sw6,Sw7} \odot \Delta_{Sw8}$ $\rightarrow \Delta_{Tc2,Sw5,Sw6,Sw7,Sw8}$

Table 1. Strategy chosen by the diagnostic system.

is that the interactions are looked for by an on-line analysis of diagnoses making this strategy dynamic and adaptable.

Several improvements have to be done in order to use the diagnostic on larger systems. An incremental algorithm able to take into account observations on successive temporal windows is in development [11]. An important point is the choice of the size of the temporal window in which the strategy of merging is applied. The optimal size of the temporal windows has still to be determined, knowing that it clearly depends on the application requirements.

References

- [1] A. Arnold. Transition systems and concurrent processes. In *Mathematical problems in Computation theory*, volume 21. Banach Center, 1987.
- [2] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence*, 110:135–183, 1999.
- [3] C. Barral, S. McIlraith, and T. C. Son. Formulating diagnostic problem solving using an action language with narratives and sensing. In *Proceedings of KR'2000*, pages 311–322, Cambridge, 2000.
- [4] L. Console, C. Picardi, and M. Ribaud. Diagnosis and diagnosability analysis using pepa. In *Proc. of the European Conference on Artificial Intelligence (ECAI'2000)*, pages 131–135, Berlin, 2000.
- [5] M.-O. Cordier and S. Thiébaux. Event-based diagnosis for evolutive systems. In *International Workshop on Principles of Diagnosis (DX-94)*, pages 64–69, 1994.
- [6] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems*, 10(1-2):33–86, 2000.
- [7] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: representation and algorithms. In *the Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'93)*, pages 166–172, Chambéry (Savoie), France, 1993.
- [8] J. Lunze. Discrete-event modelling and diagnosis of quantized dynamical systems. In *Proc. of the International Workshop on Principles of Diagnosis (DX'99)*, pages 147–154, Loch Awe, 1999.
- [9] D. Niebur. Expert systems for power system control in western europe. *Proc. of the IEEE Symposium on Intelligent Control*, pages 112–119, 1990.
- [10] Y. Pencolé. Decentralized diagnoser approach: application to telecommunication networks. In *Proc. of the International Workshop on Principles of Diagnosis (DX'00)*, pages 185–192, Morelia, Mexico, 2000.
- [11] Y. Pencolé, M.-O. Cordier, and L. Rozé. Incremental decentralized diagnosis approach for the supervision of a telecommunication network. In *Proceedings of the International Workshop on Principles of Diagnosis (DX'01)*, pages 151–158, Sansicario, Italy, 2001.
- [12] M. Riese. Diagnosis of extended finite automata as a dynamic constraint satisfaction problem. In *Proceedings of the International Workshop on Principles of Diagnosis (DX'93)*, pages 60–73, Aberystwyth, UK, 1993.
- [13] L. Rozé and M.-O. Cordier. Diagnosing discrete event systems : An experiment in telecommunication networks. In *WODES98, Fourth Workshop on Discrete Event Systems*, pages 130–137, Cagliari, Italy, 1998.
- [14] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event system. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.
- [15] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929, 1998.