A Framework for Decentralized Qualitative Model-Based Diagnosis

Luca Console, Claudia Picardi

Università di Torino
Dipartimento di Informatica
Corso Svizzera 185, 10149, Torino, Italy
lconsole,picardi@di.unito.it

Daniele Theseider Dupré

Università del Piemonte Orientale
Dipartimento di Informatica
Via Bellini 25/g, 15100, Alessandria, Italy
dtd@mfn.unimpn.it

Abstract

In this paper we propose a framework for decentralized model-based diagnosis of complex systems. We consider the case where subsystems are developed independently along with their associated (or embedded) software modules – in particular their diagnostic software. This is useful in those situations where subsystems are developed (possibly by different suppliers) without a-priori knowledge of the system in which they will be exploited, or without making assumptions on the role they will play in such system.

We describe a decentralized architecture where subsystems are analyzed by Local Diagnosers, coordinated by a Supervisor. Within the framework, both the Local Diagnosers and the Supervisor can be designed independently of each other, without any advance information on how the subsystems will be connected (provided that they share a common modeling ontology) and allowing also for runtime changes in the overall system structure. Local diagnosers are thus loosely coupled and communicate with the Supervisor via a standard interface, supporting independent implementations.

1 Introduction

The application of model-based diagnosis has often to face the problem of architectural complexity in real physical systems. Decomposition has been recognized as an important leverage to manage this type of complexity. Most of the approaches, however, focused on hierarchical decomposition [Genesereth, 1984; Mozetic, 1991], while decentralization has been explored less frequently (see e.g. [Pencolé and Cordier, 2005]).

In this paper we propose a decentralized supervised approach to diagnosis. This approach generalizes the one in [Ardissono *et al.*, 2005], which is tailored to the specific case of diagnosis of Web Services. We assume that a system is formed by subsystems each one having its Local Diagnoser, while a Supervisor is associated with the overall system, with the aim of coordinating Local Diagnosers and producing global diagnoses.

Although the details of the approach will be discussed in the following sections, it is important to notice here that we consider a general case where:

- The Supervisor needs not have a-priori knowledge about the subsystems and their paths of interaction.
- Each Local Diagnoser can adopt its own diagnostic strategy and must only implement a communication interface with the Supervisor.
- Local Diagnosers don't know each other.
- The Supervisor should behave as a Local Diagnoser in case the system is a subsystem of a more complex system (hierarchical scalability).

Several reasons motivate the adoption of such an approach. First of all, it is suitable for distributed systems where the set of subsystems and their paths of interaction vary across time. The assumptions above guarantee that subsystems can be added/removed/replaced independently of each other at any time, that the paths of interaction may even be nondeterministic, and that no one has to be informed of the change (loosely coupled integration). Such a decoupling is important and interesting for several reasons. First it allows developing Local Diagnosers independently of each other, making it possible to control the design of the diagnostics in a complex system. This is not only a problem of managing complexity. In many application cases the subsystems are designed by different entities (e.g., suppliers of the company assembling the complex system) and thus are black boxes for the designer of the complex system. The problems are common in many application domains. For example, in aerospace applications, it is very common that systems are assembled with subsystems provided by different suppliers. A simple example is a landing gear, including hydraulic, mechatronic and electronic components that are assembled by the assembler of the aircraft. Each subsystem (e.g., power transmission, or an hydraulic actuator) is supplied together with its own FMECA and control/diagnostic software, without any information on its internal details.

Indeed in our approach we assume that Local Diagnosers are designed independently of each other, and that the design of a Supervisor requires only that Local Diagnosers implement a common communication interface (which is a reasonable assumption in an assembler-supplier relationship). In this sense a decentralized approach where local diagnosers

communicate with a Supervisor is more flexible than a fully distributed approach, as we can assume that Local Diagnosers have no information or model about the others and we make no assumption on the diagnosers and the models they use (provided the models share a common ontology).

The paper is organized as follows: in the next section we describe the decentralized architecture we propose. Section 3 explains the assumptions we make on the models used for diagnosis. Then, section 4 discussed how decentralized diagnosis is carried out in our approach. Section 5 concludes with a discussion on related work.

2 Architecture

The architecture we define is a supervised one where:

- A Local Diagnoser is associated with each subsystem. The model of the subsystem is private and it is not visible outside the diagnoser.
- A Supervisor of the diagnostic process is associated with the system. It coordinates the work of the diagnosers optimizing their invocations during a diagnostic session.

The structure of the system (i.e., the subsystems and their connections) may be known a-priori to the Supervisor or may be reconstructed by it during the interaction (without any a-priori information). This increases the flexibility of the approach as it allows us to deal with system whose structure (number of components and their connections) may vary dynamically. The architecture defines a communication interface between the Local Diagnosers and the Supervisor. Local diagnosers are awakened by the subsystem they are in charge of whenever an anomaly (fault) is detected. They may explain the fault locally or may put the blame on an input received from another subsystem. They communicate to the Supervisor their explanations (without disclosing the details of the local failure(s)). The Supervisor may invoke the Local Diagnoser of the blamed subsystem asking it to generate explanations. The Supervisor may also ask a Local Diagnoser to check some predictions. The selection of the Local Diagnosers to invoke is made according to rules that optimize invocations, as we will discuss in section 4.2.

The architecture is scalable (hierarchical) in the sense that the Supervisor can act as a Local Diagnoser when the system is used as a subsystem in a more complex system.

Notice that, in order to communicate, the Supervisor and Local Diagnosers must speak a common language; in other words, we must assume that they share a modeling ontology that allows them to share information about exchanged quantities. This nevertheless allows each Local Diagnoser to use its own modeling language and assumptions, and even to internally use a different ontology if needed, provided that it is able to map it over the common one during communications.

3 Models

The approach we propose focuses on Qualitative Models; in this paper we will in particular deal with deviation models [Malik and Struss, 1996] although we will briefly describe how our proposal can be applied to other qualitative models as well. It is worth noting that deviation models proved to be

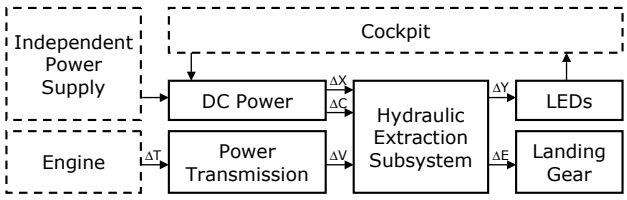

Figure 1: The Landing Gear section of an Aircraft System

successful in several applications to technical domains (e.g., [Sachenbacher *et al.*, 2000; Picardi *et al.*, 2002; 2004]).

As we discussed in the previous section, the Supervisor is not aware of specific subsystem models; however, it is aware of their existence and, in order to coordinate Local Diagnoser and combine the information they provide, it must make some assumptions on the modeling ontology. In this section we will discuss these assumptions.

First of all, as it is common in a large part of model-based diagnosis, we consider component-oriented models, where each component has one correct behavior and one or more faulty behaviors. Each behavior of a component is modeled as a relation over a set of variables. Variables that represent quantities exchanged with other components are either input or output variables. The complete system is specified as a set of assignments of an output variable of one component to an input variables of another one.

By Qualitative Model (of a physical system) we intend a model where all variables have a finite, discrete domain, thus representing either a discretization of the corresponding physical quantity, or an abstraction over some of its properties. This implies that, in a qualitative model, the relation that defines a component or system can be expressed extensionally (although in most cases this is not desirable). In this context, each component C can be represented as a unique relation over a set of variables by introducing a distinguished *mode* variable $C.m$, that ranges over the set of behavior modes $\{ok, ab_1, \ldots, ab_n\}$ of the component itself.

As mentioned above, in this paper we will particularly focus on deviation models, which means that:

- Each physical quantity q represented in the model has a corresponding variable Δq representing a qualitative abstraction of the *deviation* of q with respect to its expected value. Common domains for deviation variables are $\{ok, ab\}$, expressing whether the value of q is normal or not, and $\{-, 0, +\}$, expressing whether q is lower than, equal to, or higher than its expected value. Here we will take this last option; we will therefore be discussing *sign-based* deviation models.
- Each behavior model expresses relations among deviation variables. For example, an electrical model might state that if the resistance in a circuit is higher than expected, than the intensity of the current flowing in the circuit is lower than expected.

We will now introduce a running example that we will use throughout the paper to illustrate the approach.

Figure 1 shows a simplified part of an Aircraft System, namely the part concerned with the Landing Gear. The pic-

ture represents a high-level view of the system: we do not see individual components, but interacting subsystems. Subsystem pictured with a dashed line do not directly take part in the example, but are shown for the sake of completeness.

The Hydraulic Extraction System (**HES**) creates a pressure that mechanically pushes the Landing Gear, thereby extracting it. The **HES** is also connected to some leds on the cockpit, that show whether the subsystem is powered up or not. In order to create the pressure, the **HES** takes power from two sources. The main source is the Power Transmission from the aircraft engine, which actually powers up the main pump of the **HES**. A secondary source (used to transmit the pilot command from the cockpit, and to light up leds) is the Independent Power Supply, which produces a low-amperage DC current.

We will detail parts of this example in the following, as needed to explain the different aspects of the decentralized diagnostic process.

4 Decentralized Diagnostic Process

In this section we will describe how the diagnostic process is carried out. As we have already mentioned, a Supervisor coordinates the activities of several Local Diagnosers $\{LD_1, \ldots, LD_n\}$. In order to obtain a loosely coupled integration, the Supervisor assumes that each LD_i is stateless, that is every invocation of a Local Diagnoser is independent of the others. Moreover, the Supervisor does not make any assumption on the implementation of each Local Diagnoser. The interaction is thus defined by:

- An interface that each Local Diagnoser must implement, discussed in section 4.1. The role of Local Diagnosers consists in generating htpotheses consistent with their model and observations.
- An algorithm for the Supervisor that computes diagnoses by coordinating the Local Diagnoser through the above interface, explained in section 4.2. The role of the Supervisor is to propagate hypotheses to the neighbors of a Local Diagnoser.

Figure 2 shows an example of this architecture for the system in Figure 1.

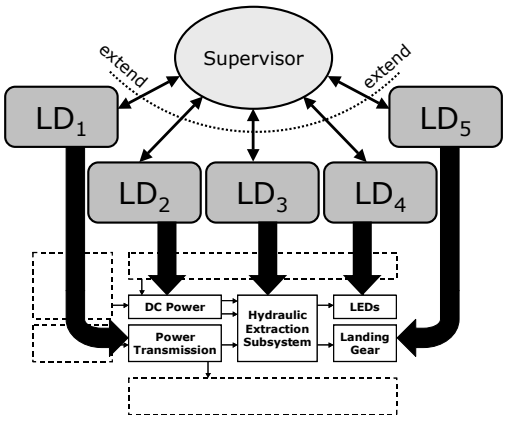

Figure 2: An example of decentralized architecture

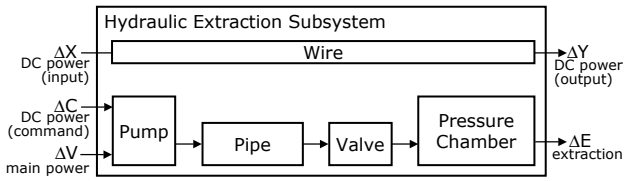

Figure 3: A closer view to the Hydraulic Extraction System

4.1 The Extend interface

We assume that each Local Diagnoser LD_i reasons on a model M_i of its subsystem S_i, according to what we discussed in section 3. From the point of view of the Supervisor and its interactions with Local Diagnosers, each model M_i is a relation over a set of variables where:

- mode variables, denoted by $S_i.m$, express the behavior mode of components in S_i; each of them has thus a different domain;
- input/output variables express deviations of quantities that S_i exchanges with other subsystems.

Of course a model M_i may include additional (internal, and private) variables, and may be not expressed at all as an extensional relation. This is however hidden in the implementation of each Local Diagnoser.

The interface between the Supervisor and the Local Diagnosers is made of a single operation called EXTEND that the Supervisor invokes on the Local Diagnosers. Moreover, upon an abnormal observation, Local Diagnosers autonomously execute EXTEND and send the results to the Supervisor.

The goal of EXTEND is to explain and/or verify the consistency of observations and/or hypotheses made by other Local Diagnosers. Thus, the input to EXTEND when invoked on LD_i is a set of hypotheses on the values of input/output variables of M_i. Such hypotheses are represented as a set of *partial*[1] assignments over the variables of interest.

In the particular case where EXTEND is autonomously executed by a Local Diagnoser upon receiving an abnormal observation from its subsystem, the set of hypotheses contains only the empty assignment (i.e. an assignment with an empty domain).

For each hypothesis α received in input, LD_i must first of all check whether α is consistent with the local diagnostic model M_i and the local observations ω_i. Then LD_i must *extend* α in two directions: backwards to verify whether that hypothesis needs further assumptions to be supported within M_i and ω_i, and forward to see whether the hypothesis has consequences on other subsystems, that may be verified in order to discard or confirm it.

Let us consider as an example the Hydraulic Extension System, a (simplified) close-up view of which is depicted in Figure 3.

Suppose the Local Diagnoser LD_3, associated to the **HES**, receives in input an assignment that assigns the value "$-$" to

[1] A *partial assignment* over a set of variables V is any assignment α whose domain $\mathbf{Dom}(\alpha)$ is a *subset* of V.

its output data variable **HES**.ΔE, representing the mechanical extraction of the Landing Gear. This value means that the Landing Gear did not extract when expected.

LD$_3$ may see that, with respect to the local model, for that piece of data to be less than expected, one of the following must have happened:

- An internal component has failed, for example the pump is stuck or the pipe is leaking.
- One of the two inputs of the pump is wrong, for example the pump is not powered or it has not received the command.

This means that the partial assignment **HES**.$\Delta E = -$ can be extended in four ways: by assigning **HES**.$m =$ **stuck**, or **HES**.$m =$ **leaking**, or **HES**.$\Delta V = -$, or **HES**.$\Delta C = -$.

What is important regarding extensions is that they should include everything that can be said under a given hypothesis, but nothing that could be said also without it. In other words, we are interested in knowing whether a given assignment constrains other variables *more* than the model alone does. The following definition captures and formalizes this notion.

Def. 1 [2] Let M_i be a local model with local observations ω_i, and let γ be an assignment. We say that γ is *admissible* with respect to M_i and ω_i if:

i. γ is an extension of ω_i;

ii. γ is consistent with M_i;

iii. if $M_{i,\gamma}$ is the model obtained by constraining M_i with γ, its restriction to variables *not* assigned in γ is equivalent to the restriction of M_i itself to the same variables:

$$M_{i,\gamma}|_{\text{VAR}(M_i)\setminus\textbf{Dom}(\gamma)} \equiv M_i|_{\text{VAR}(M_i)\setminus\textbf{Dom}(\gamma)}.$$

Thus, for each assignment α received in input, the EXTEND operation of a Local Diagnoser *LD*$_i$ returns a set of assignments E_α that contains **all minimal admissible extensions** of α with respect to M_i and ω_i, restricted to input, output and mode variables of M_i. Notice that, whenever α is inconsistent with the observations and/or the model, E_α is empty.

Minimal admissibility avoids unnecessary committments on values of variables that are not concretely constrained by the current hypothesis.

In order to illustrate these definitions, let us consider a (again, simplified) model M_P of the pump P alone. The model includes four variables: $P.m$ represents the behaviour mode, $P.\Delta V$ the power supply to the pump, $P.\Delta C$ the command that turns on the pump, and $P.\Delta F$ the flow coming out from the pump. The extensional representation of M_P is:

$P.m$	$P.\Delta C$	$P.\Delta V$	$P.\Delta F$	$P.m$	$P.\Delta C$	$P.\Delta V$	$P.\Delta F$
ok	0	0	0	stuck	0	0	$-$
ok	0	$+$	$+$	stuck	0	$+$	$0,+,-$
ok	0	$-$	$-$	stuck	0	$-$	$-$
ok	$+$	0	$+$	stuck	$+$	0	$0,+,-$
ok	$+$	$+$	$+$	stuck	$+$	$+$	$0,+,-$
ok	$+$	$-$	$0,+,-$	stuck	$+$	$-$	$0,+,-$
ok	$-$	0	$-$	stuck	$-$	0	$-$
ok	$-$	$+$	$0,+,-$	stuck	$-$	$+$	$0,+,-$
ok	$-$	$-$	$-$	stuck	$-$	$-$	$-$

[2] The definitions in this section are rephrased from [Ardissono *et al.*, 2005].

Now suppose we execute EXTEND on M_P alone, having in input an assignment α such that **Dom**$(\alpha) = \{P.\Delta F\}$ and $\alpha(P.\Delta F) = -$. Let us first of all verify that α itself is not admissible, by computing $M_P|_{\{P.m,P.\Delta C,P.\Delta V\}}$ and $M_{P,\alpha}|_{\{P.m,P.\Delta C,P.\Delta V\}}$.

$M_P|_{\{P.m,P.\Delta C,P.\Delta V}$

$P.m$	$P.\Delta C$	$P.\Delta V$
ok	0	0
ok	0	$+$
ok	0	$-$
ok	$+$	0
ok	$+$	$+$
ok	$+$	$-$
ok	$-$	0
ok	$-$	$+$
ok	$-$	$-$
stuck	0	0
stuck	0	$+$
stuck	0	$-$
stuck	$+$	0
stuck	$+$	$+$
stuck	$+$	$-$
stuck	$-$	0
stuck	$-$	$+$
stuck	$-$	$-$

$\not\equiv$

$M_{P,\alpha}|_{\{P.m,P.\Delta C,P.\Delta V\}}$

$P.m$	$P.\Delta C$	$P.\Delta V$
ok	0	$-$
ok	$+$	$-$
ok	$-$	0
ok	$-$	$+$
ok	$-$	$-$
stuck	0	0
stuck	0	$+$
stuck	0	$-$
stuck	$+$	0
stuck	$+$	$+$
stuck	$+$	$-$
stuck	$-$	0
stuck	$-$	$+$
stuck	$-$	$-$

It is easy to see that the minimal admissible extensions of α wrt M_P are the following:

	$P.m$	$P.\Delta C$	$P.\Delta V$	$P.\Delta F$
γ_1	stuck			$-$
γ_2		$-$		$-$
γ_3			$-$	$-$

γ_1, γ_2 and γ_3 express the three possible explanations for $\Delta F = -$: either the pump is stuck, or it is not powered, or it did not receive the command. For each of the three assignments, unassigned variables are those whose value is irrelevant to the explanation.

4.2 The Supervisor algorithm

The Supervisor is started by a Local Diagnoser *LD*$_{\text{first}}$ that sends to it the results of an EXTEND operation executed as a consequence of an abnormal observation.

During its computation, the Supervisor records the following information:

- a set $\mathcal{H}$ of assignments, representing current hypotheses;
- for each assignment α and for each variable $x \in$ **Dom**(α), a *modified bit*. By **mdf**$(\alpha(x))$ we will denote the value of this bit for variable x in assignment α.

Moreover, we assume that given an input or output variable x that has been mentioned by one of the Local Diagnosers, the Supervisor is able to determine the variable **conn**(x) connected to it. We do not make any assumption on whether this information is known *a priori* by the Supervisor, or it is retrieved dynamically, or it is provided by the Local Diagnoser itself.

The Supervisor initializes its data structures with the results of the initial EXTEND operation that has awakened it:

- $\mathcal{H}$ contains all the assignments that were sent by *LD*$_{\text{first}}$ as the result of EXTEND;
- for each $\alpha \in \mathcal{H}$, and for each $x \in$ **Dom**(α), the Supervisor extends α to the variable **conn**(x) connected with x by assigning $\alpha($**conn**$(x)) = \alpha(x)$. Then the Supervisor sets **mdf**$(\alpha($**conn**$(x))) = 1$.

Modified bits are used by the Supervisor to understand whether it should invoke a Local Diagnoser or not. The basic rule it uses is the following:

Basic Rule. If a subsystem S_i has a variable x with $\mathbf{mdf}(\alpha(x)) = 1$ for some α, then LD_i is a candidate for invocation and α should be passed on to EXTEND.

There are however two exceptions to this rule, that allow to reduce the number of invocations:

Vertical Exception. Given a variable x belonging to a subsystem S_i, if all assignments give the same value to x, and either the value is 0 or x is an input variable, then x and its modified bits should not count towards deciding whether LD_i should be invoked or not.

This exception avoids that a Local Diagnoser is invoked to verify/discard something that has already been assessed to be true. If however the assesses truth concerns an abnormal value for one of S_i's outputs, then LD_i is still asked to give an explanation.

This exception is referred to as *vertical* because it considers the value of a variable throughout different assignments, that is a *column* in the assignments table representing $\mathcal{H}$.

Horizontal Exception. Given an assignment α and a subsystem S_i, if $mdf(\alpha(x)) = 1$ implies $\alpha(x) = 0$, then α and its modified bits should not count towards deciding whether LD_i should be invoked or not. If LD_i is nevertheless invoked, α should be passed on to extend only if LD_i has never been invoked before.

This exception avoids that a Local Diagnoser is invoked only to verify that "everything ok" is consistent with its model (which is trivially true).

It is called *horizontal* because it considers values of different variables in the same assignment, that is a *row* in the assignments table representing $\mathcal{H}$.

Notice that when deciding whether the **Horizontal Exception** can be applied, only variables with modified bits set to 1 are considered. In order to apply it, we do not need the whole assignment to assign value 0, but only the part of it that has never been examined by the proper Local Diagnoser. Thus , if a Local Diagnoser LD_i has already been invoked, it is never invoked again until one of the variables in S_i is assigned a value different than 0. Thanks to the definition of EXTEND, we do not have to worry that a newly assigned 0 is inconsistent with a previously extended assignment.

After initializing data structures, the Supervisor loops over the following three steps:

Step 1: select the next LD_i to invoke. The Supervisor selects one of the Local Diagnosers LD_i for which there is at least one assignment α meeting the **Basic Rule** with no **Exception**. *If there is none, the loop terminates.*

Step 2: invoke EXTEND**.** If LD_i has never been invoked before in this diagnostic process, then the input to EXTEND is the set of all assignments in H, restricted to variables of M_i. Otherwise the input consists only of those assignments α that meet the **Basic Rule** but not the **Horizontal Exception**.

Step 3: update $\mathcal{H}$. The Supervisor receives the output of EXTEND from LD_i. For each α in input, EXTEND has returned a (possibly empty) set of extensions $E_\alpha = \{\gamma_1, \ldots, \gamma_k\}$. Then α is *replaced* in $\mathcal{H}$ by the set of assignments $\{\beta_1, \ldots, \beta_k\}$ where β_j is obtained by:

- combining α with each $\gamma_j \in E_\alpha$;
- extending the result of this combination to connected variables, so that for each $x \in \mathbf{Dom}(\gamma)$ representing an input/output variable, $\beta_j(\mathbf{conn}(x)) = \beta_j(x)$.

This implies that rejected assignments, having no extensions, are removed from $\mathcal{H}$.

Step 4: update the mdf bits. For each assignment β_j added in Step 3 **mdf** bits are set as follows:

(i) For each variable x *not* belonging to M_i such that $x \in \mathbf{Dom}(\beta_j)$ and $x \notin \mathbf{Dom}(\alpha)$, $\mathbf{mdf}(\beta_j(x))$ is set to 1.

(ii) For each variable x belonging to M_i such that $x \in \mathbf{Dom}(\beta_j)$, $\mathbf{mdf}(\beta_j(x))$ is set to 0.

(iii) For any other variable $x \in \mathbf{Dom}(\beta_j)$, $\mathbf{mdf}(\beta_j(x)) = \mathbf{mdf}(\alpha(x))$.[3]

Notice that the diagnostic process terminates: new requests for EXTEND are generated only if assignments are properly extended, but assignments cannot be extended indefinitely.

At the end of the loop, assignments in $\mathcal{H}$ provide consistency-based diagnoses as follows:

Def. 2 Let α be an assignment in $\mathcal{H}$ at the end of the diagnostic process. The *diagnosis* associated with α is the assignment $\Delta(\alpha)$ such that:

- $\mathbf{Dom}(\Delta(\alpha))$ is the set of all mode variables of all involved models;
- for each $x \in \mathbf{Dom}(\alpha) \cap \mathbf{Dom}(\Delta(\alpha))$, $\Delta(\alpha)(x) = \alpha(x)$;
- for each $x \in \mathbf{Dom}(\Delta(\alpha)) \setminus \mathbf{Dom}(\alpha)$, $\Delta(\alpha(x)) = ok$.

Of these diagnoses, non-minimal ones are discarded.

It can be proved that this algorithm computes all minimal consistency based diagnoses with respect to the model and observations of the part of the system involved in the diagnostic process.

Notice that in the Supervisor algorithm the role of mode variables is rather peculiar. Since these variables are local to a given system, the Supervisor never communicates their value to a Local Diagnoser different than the one originating them. Thus, they are not needed for cross-consistency checks. There are two main reasons why we need the Supervisor to be aware of them:

- They provide the connection between two different invocations on the same Local Diagnoser; by having the Supervisor record them and send them back on subsequent calls, we allow the Local Diagnosers to be stateless.
- The Supervisor needs them to compute globally minimal diagnoses.

[3] Variables in $\mathbf{Dom}(\beta_j)$ but not in $\mathbf{Dom}(\alpha)$ are all covered in the first two cases.

Notice however that, for both of these goals, the value of mode variables needs not be explicit: in other words the Local Diagnosers may associate to each fault mode a coded Id and use it as a value for mode variables. In this way, information on what has happened inside a subsystem can remain private.

Let us conclude this section with an example of execution of the Supervisor algorithm on the system in figure 1. The names of the subsystems will be shortened as follows: **Eng** will denote the Engine; **DCP** the DC Power; **PT** the Power Transmission; **HES** the Hydraulic Extraction System; **LG** the Landing Gear; **LED** the LEDs. Moreover, we will write $S.x$ to denote variable x belonging to subsystem S, and $S.m$ will denote a variable summarizing the behaviour mode of components in S.

Let us assume that **LG** observes a $-$ value for **LG**.ΔE. It autonomously invokes EXTEND, which as an output has only the following assignment:

	LG.m	**LG**.ΔE
α_0		$-$

The Supervisor receives this assignment, and accordingly initializes its data structures. Figure 4 shows how $\mathcal{H}$ changes during the execution of the Supervisor algorithm; in particular, $\mathcal{H}_0$ corresponds to $\mathcal{H}$ immediately after initialization.[4]

At this stage the only candidate for invocation is LD_3, responsible for the **HES**. Its input consists of assignment α_0 restricted to **HES** variables, which (as already discussed in section 4.1) is extended by LD_3 as follows:

	HES.ΔE
α_0	$-$

$\Longrightarrow$

	HES.m	**HES**.ΔE	**HES**.ΔV	**HES**.ΔC
α_{01}	**leak**	$-$		
α_{02}	**stuck**	$-$		
α_{03}		$-$	$-$	
α_{04}		$-$		$-$

The Supervisor receives these result and updates $\mathcal{H}$, obtaining $\mathcal{H}_1$ depicted in Figure 4.

Now there are two candidates for invocation, LD_2 and LD_1, respectively responsible for the **DCP** and for the **PT**. They can be invoked in any order; let us assume that LD_2 is invoked first. Since it is the first invocation, its input are all the assignments in $\mathcal{H}$ restricted to **DCP** variables. For the sake of the example, let us assume that the **DCP** can produce a $-$ on ΔC only by being failed itself, and that as consequence of this failure also ΔX should be $-$ as well. Then, LD_2 extends the input assignments in this way:

	DCP.ΔC
$\alpha_{0[1,2,3]}$	
α_{04}	$-$

$\Longrightarrow$

	DCP.m	**DCP**.ΔC	**DCP**.ΔX
$\alpha_{0[1,2,3]}$			
α_{04}	**fail**	$-$	$-$

The Supervisor therefore updates $\mathcal{H}$; the result is the set $\mathcal{H}_2$.

Now it's LD_1's turn. The input and output of EXTEND are:

	PT.ΔV
$\alpha_{0[1,2,4]}$	
α_{03}	$-$

$\Longrightarrow$

	PT.m	**PT**.ΔV	**PT**.ΔT
$\alpha_{0[1,2,4]}$			0
α_{03}	**fail**	$-$	0

In this case, we assume that the **PT** has observed that its input ΔT from the engine is normal. As a consequence, it can explain the $-$ on ΔV only with an internal failure. Thus, the output sent to the Supervisor is:

[4] In Figure 4 the symbol $\sharp$ indicates that for the corresponding assignment and variable the modified bit is set to 1.

The set of hypotheses is updated accordingly and the Supervisor obtains $\mathcal{H}_3$.

The only candidate for invocation is now LD_3, because although **Eng** has some modified bits set to 1, it falls in both the **Exceptions**. Thus LD_3 is invoked again, only on the modified assignment, that is α_{04}, restricted to **HES** variables. Now, let us assume that from the point of view of the **HES** having $\Delta X = -$ implies necessarily that also $\Delta Y = -$. The result of EXTEND is then:

	HES.m	**HES**.ΔE	**HES**.ΔV	**HES**.ΔC	**HES**.ΔX	**HES**.ΔY
α_{04}		$-$		$-$	$-$	$-$

$\mathcal{H}$ now becomes as $\mathcal{H}_4$ in Figure 4. Now LD_4 is invoked; let us assume that observations on **LED** report that everything is ok with the leds. What happens is that one of the two input assignment is extended, while the other is rejected as inconsistent. Here are the input and output of EXTEND:

	LED.ΔY
$\alpha_{0[1,2,3]}$	
α_{04}	$-$

$\Longrightarrow$

	LED.M	**LED**.ΔY
$\alpha_{0[1,2,3]}$		0

Updating $\mathcal{H}$ the Supervisor obtains $\mathcal{H}_5$.

Notice that due to the **Exception** there is no Local Diagnoser left to invoke, thus the diagnostic process ends here. Minimal diagnoses thus are:

HES.m = **leak:** the pipe of the Hydraulic Extraction System is leaking (from α_{01}).

HES.m = **stuck:** the pump of the Hydraulic Extraction System is stuck (from α_{02}).

PT.m = **fail:** the Power Transmission is broken (from α_{03}).

4.3 Other types of models

We briefly discuss here how the approach can be applied also to models different than the sign-based deviation models used in our explanation.

In general, the algorithm can be applied as is to any type of deviation model where there is only one value associated to the nominal behavior. Models with variable domains such as $\{ok, ab\}$ or $\{--, -, 0, +, ++\}$ belong to this category.

In order to apply the approach to a generic qualitative model, we need to modify the Supervisor algorithm by:

- discarding the **Horizontal Exception**;
- modifying the **Vertical Exception**, so that it applies only to input variables.

It is clear that giving up the **Horizontal Exception** means increasing the number of invocations to Local Diagnosers. What however matters is whether these invocations are useful for the Diagnostic Process; in other words, whether the level of abstraction in the model is suitable for the diagnostic task.

For mixed models, that is models that combine deviation variables with other types of variables, it may neverthelsess be possible to define other exceptions to the **Basic Rule**, more restrictive than the one described here, that allow to reduce the number of invocations. However each type of model should be analyzed separately to discover whether this optimization is possible.

$\mathcal{H}_0$

	LG.m	LG.ΔE	HES.ΔE
α_0		–	–♯

$\mathcal{H}_1$

	LG.m	LG.ΔE	HES.m	HES.ΔE	HES.ΔV	HES.ΔC	DCP.ΔC	PT.ΔV
α_{01}		–	leak	–				
α_{02}		–	stuck	–				
α_{03}		–		–	–			–♯
α_{04}		–		–		–	–♯	

$\mathcal{H}_2$

	LG.m	LG.ΔE	HES.m	HES.ΔE	HES.ΔV	HES.ΔC	HES.ΔX	DCP.m	DCP.ΔC	DCP.ΔX	PT.ΔV
α_{01}		–	leak	–							
α_{02}		–	stuck	–							
α_{03}		–		–	–						–♯
α_{04}		–		–		–	–♯	fail	–	–	

$\mathcal{H}_3$

	LG.m	LG.ΔE	HES.m	HES.ΔE	HES.ΔV	HES.ΔC	HES.ΔX	DCP.m	DCP.ΔC	DCP.ΔX	PT.m	PT.ΔV	PT.ΔT	Eng.ΔT
α_{01}		–	leak	–									0	0♯
α_{02}		–	stuck	–									0	0♯
α_{03}		–		–	–						fail	–	0	0♯
α_{04}		–		–		–	–♯			fail			0	0♯

$\mathcal{H}_4$

	LG.m	LG.ΔE	HES.m	HES.ΔE	HES.ΔV	HES.ΔC	HES.ΔX	HES.ΔY	LED.ΔY	DCP.m	DCP.ΔC	DCP.ΔX	PT.m	PT.ΔV	PT.ΔT	Eng.ΔT
α_{01}		–	leak	–											0	0♯
α_{02}		–	stuck	–											0	0♯
α_{03}		–		–	–								fail	–	0	0♯
α_{04}		–		–		–	–	–	–♯			fail			0	0♯

$\mathcal{H}_5$

	LG.m	LG.ΔE	HES.m	HES.ΔE	HES.ΔV	HES.ΔC	HES.ΔX	HES.ΔY	LED.ΔY	DCP.m	DCP.ΔC	DCP.ΔX	PT.m	PT.ΔV	PT.ΔT	Eng.ΔT
α_{01}		–	leak	–				0♯	0						0	0♯
α_{02}		–	stuck	–				0♯	0						0	0♯
α_{03}		–		–	–			0♯	0				fail	–	0	0♯

Figure 4: The set $\mathcal{H}$ in different stages of the Supervisor algorithm, during a diagnosis for the system in Figure 1.

5 Conclusions

In this paper we discuss a supervised decentralized approach to diagnosis that allows to loosely couple multiple diagnosers, in order to deal with architecturally complex systems.

The goal of the paper is to show that we can effectively perform the diagnostic task and define intelligent strategies for the Supervisor, even if it is not aware of the internal aspects of Local Diagnosers, the Local Diagnosers do not know each other, and their paths of interaction dynamically change.

In this paper we do not face the problem of the implementation of Local Diagnosers. The discussion of efficient algorithms that execute the EXTEND operation is out of the scope of this paper, especially since such algorithms would strongly depend on the modeling language and specific assumptions of each individual Local Diagnoser.

It is worth noting, however, that the decentralized approach can be hierarchically applied to subsystems, possibly having the same program playing both the role of the Supervisor and of the Local Diagnoser. In this case the advantage of the decentralized approach would be, rather than information hiding, to exploit the Supervisor to choose which parts of the model should be analyzed. At the lowest level (i.e., the level of basic components) this would still require to directly execute EXTEND; however for small models the results of this operation could easily be pre-compiled.

Hierarchical scalability is indeed an important feature of our approach which allows to integrate hierarchical decomposition as a further means to deal with the complexity of the systems to be diagnosed.

As we already mentioned, the approach proposed in this paper is a generalization of the one proposed in [Ardissono *et al.*, 2005] to deal with diagnosis of Web Services. In particular, the definition of the EXTEND interface is borrowed from [Ardissono *et al.*, 2005], while the Supervisor algorithm is generalized in order to loosen the assumptions and still produce all minimal consistency-based diagnosis.

If we consider other approaches in the literature, we see that [Pencolé and Cordier, 2005] has some relevant similarity from the point of view of diagnostic architecture: a supervisor is in charge of computing global diagnoses exploiting the work of several local diagnosers. However, this work differs in some significant respects:

- the system to diagnose is modeled as a discrete-event system;
- the main problem is to avoid composing the whole model, because this would produce a state-space explosion;
- as a consequence, the supervisor is aware of the subsystem models, but cannot compose them: it can only compose local diagnoses to obtain a global one;
- due to the nature of the considered systems, reconstructing global diagnoses is a difficult task, and as such it is one of the main focuses of the paper.

Thus [Pencolé and Cordier, 2005] actually focuses on quite

different theoretical and practical problems than those addressed in this paper.

Other papers in the literature deal with completely distributed approaches to diagnosis, where diagnosers communicate with each other and try to reach an agreement, without a supervisor coordinating them. Our choice of a supervised approach was motivated by the need of having loosely coupled diagnosers, while a purely distributed approach requires diagnosers to establish diagnostic sessions and exchange a greater amount of information in order to compute diagnoses that are consistent with each other.

Nevertheless, the distributed approach proposed in [Roos *et al.*, 2003] has some similarity with ours because it is based on the idea of diagnosers explaining blames received from others, or verifying hypotheses made by others. Needless to say, the proposed algorithms are completely different, having to deal with a distributed approach. Moreover, in order to reduce computational complexity, the authors introduce some fairly restrictive assumptions on the models (e.g. requiring that explanations imply normal observations, or that output values are either completely undetermined or completely determined), which are not acceptable in our case.

Another distributed approach on a similar basis is the one in [Kalech and Kaminka, 2004]; however in this case the focus is on diagnosing failures in the communication among agents in a team, a problem the authors refer to as *social diagnosis*. Thus the tackled problem is different from the one we cope with, since it deals with failures arising in the communication between subsystems with different sets of beliefs, rather than with failures happening inside a subsystem and propagating to others.

Finally, an interesting distributed approach is tackled in [Provan, 2002]. Here the scenario is rather similar to ours: each diagnoser has a local model and observations, which are not shared with the others. Each local diagnoser computes local minimal diagnoses, and then engages in a conversation to reach a globally sound diagnosis. However, being the approach purely distributed, solutions are different. In particular, in order to propose a solution with reduced complexity, the author focuses on systems whose connection graph has a tree-like structure. On the contrary, our approach (thanks to the presence of the supervisor) does not need to make any assumption on system structure; in fact, system structure may even dynamically change.

References

[Ardissono *et al.*, 2005] L. Ardissono, L. Console, A. Goy, G. Petrone, C. Picardi, M. Segnan, and D. Theseider Dupré. Cooperative model-based diagnosis of web services. In *Proceedings of the Sixteenth International Workshop on the Principles of Diagnosis (DX 05)*, Pacific Grove, California, 2005.

[Genesereth, 1984] M.R. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24(1-3):411–436, 1984.

[Kalech and Kaminka, 2004] M. Kalech and G.A. Kaminka. Diagnosing a team of agents: scaling-up. In *Proc. 15th Int. Work. on Principles of Diagnosis (DX-04)*, pages 129–134, 2004.

[Malik and Struss, 1996] A. Malik and P. Struss. Diagnosis of dynamic systems does not necessarily require simulation. In *Proc. 7th Int. Work. on Principles of Diagnosis*, pages 147–156, 1996.

[Mozetic, 1991] I. Mozetic. Hierarchical model-based diagnosis. *Int. J. of Man-Machine Studies*, 35(3):329–362, 1991.

[Pencolé and Cordier, 2005] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164(1-2), 2005.

[Picardi *et al.*, 2002] C. Picardi, R. Bray, F. Cascio, L. Console, P. Dague, O. Dressler, D. Millet, B. Rehfus, P. Struss, and C. Vallée. IDD: Integrating Diagnosis in the Design of automotive systems. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI2002)*, pages 628–632, 2002.

[Picardi *et al.*, 2004] C. Picardi, L. Console, F. Berger, J. Breeman, T. Kanakis, J. Moelands, E. Arbaretier, S. Collas, N. De Domenico, E. Girardelli, O. Dressler, P. Struss, and B. Zilbermen. AUTAS: a tool for supporting FMECA generation in aeronautic systems. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI2004)*, 2004.

[Provan, 2002] G. Provan. A model-based diagnostic framework for distributed systems. In *Proc. 13th Int. Work. on Principles of Diagnosis (DX-02)*, pages 16–22, 2002.

[Roos *et al.*, 2003] N. Roos, A. ten Teije, and C. Witteveen. A protocol for multi-agent diagnosis with spatially distributed knowledge. In *2nd Int. Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2003)*, Melbourne, Australia, July 2003.

[Sachenbacher *et al.*, 2000] M. Sachenbacher, P. Struss, and R. Weber. Advances in design and implementation of obd functions for diesel injection systems based on a qualitative approach to diagnosis. In *SAE 2000 World Congress*, 2000.