

ITERATIVE EXPERT DRIVEN FAULT DIAGNOSIS BASED ON STRUCTURAL MODELING

Stéphane Ploix* Patrick Chazot**

* *Laboratoire d'Automatique de Grenoble, INPG, UJF,
CNRS UMR5528,
BP46, F-38402 Saint Martin d'Hères Cedex, France
stephane.ploix@inpg.fr*

** *2 parc Fernand Texier,
F-38400 Saint Martin d'Hères, FRANCE
patrick.chazot@free.fr*

Abstract: The aim of this paper is to propose a diagnostic tool able to support experts during the localization of faults, which is compatible with the industrial maintenance context. In order to reach this aim, a purely structural approach, which does not require behavioral models, is introduced because it reduces modeling efforts. Then, thanks to this approach, an iterative diagnostic method is proposed. It is driven by experts that select the most relevant tests to be performed. After each iteration, experts only depict the performed tests and a consistency based diagnosis algorithm computes and sorts all the possible faults. It allows experts to improve step by step the structural model from a macroscopic description to a detailed one when it is necessary. *Copyright ©2006 IFAC.*

Keywords: fault diagnosis, structural modeling, maintenance engineering, quality engineering

1. INTRODUCTION

World of maintenance abounds of complex problems to be solved in limited delays. Methodology coming from quality and reliability domains are often adopted. Six Sigma (Doggett 2004) proposes several solving problem tools. Let's mention DMAIC (Define, Measure, Analyse, Improve, Control), the 5 whys, the Cause & Effect Diagram (fishbone or Ishikawa), the method of the 5W2H (Why? What? Where? When? Who? How? How much?), the Root Cause Analysis, the FMEA (Failure Mode and Effects Analysis), HAZOP... Methods like 5W2H and CED deal with testing strategy, in other words, what are the suitable questions to answer to localize faults. Methods like RCA deal with the search of the causes of

faults. As a summary, except for FMEA and HAZOP, all these methods support testing strategy in depth and in breath and identification of the causes of faults but no one tackles the problem of localizing faults. FMEA (Hawkins and Woolons 1998) and HAZOP (Hyatt 2003) aims at identifying risks but are irrelevant to diagnose a fault that has occurred. Therefore, supporting experts in solving complex problems requires additional tools dealing with the localisation of faults. Traditional diagnostic tools require a detailed and complete behavioral description of the systems to be analysed. Unfortunately, systems are generally too complex to be a priori detailed.

Solving problem tools for maintenance companies should be able to: manage complexity, allow high

reactivity, facilitate communication, make possible intuition, be systematic and be as much as possible independent of specific application areas. To satisfy these requirements, the expert knowledge has to be used without requiring lots of formalizations. Expert is indeed able to manage some tasks such as selecting and performing the most relevant tests without any help. But conversely, reasoning when many tests have been performed in order to summarize the current knowledge about a faulty situation is difficult to manage.

The aim of this paper is to propose a diagnostic tool able to support experts during the localization of faults, which is compatible with the industrial maintenance context. In order to reach this aim, it is shown that it is not necessary to model behavior to manage diagnosis: structural modeling is sufficient.

2. FORMALIZATION

This section deals with concepts and formalism used to manage structural diagnosis of industrial systems. The proposed semantics aim at modeling systems in a standardized form for diagnostic analysis. It is shown that complex industrial systems can be modeled by Horn clauses, i.e. with clauses like \neg '*proposition1*' \vee '*proposition2*' i.e. if '*proposition1*' then '*proposition2*', denoted $\textit{proposition2} \leftarrow \textit{proposition1}$, which are very standard in inference engines coming from Artificial Intelligence (Russell and Norvig 2003). Structural modeling has been introduced in (Davis 1984). (Chittaro and Ranon 2004) used semantic theory of abstraction to manage behavior together with different levels of structural abstraction (Nayak and Levy 1995) in diagnosis. Physical system modeling relies on different levels (Chittaro and Kumar 1998):

- structural level, what is in a system and how its components are connected?
- behavioral level, how does each element work?
- functional level, what does each element do?
- teleological level, what is each element for?

An additional level has to be added to this decomposition: the phenomenological level. It is a key level in diagnosis because it represents what can be known about a system to be diagnosed. Behavioral and functional levels may both be modelled by constraints. Therefore, they are merged: it is up to the expert to model components or functions providing that all the model elements are separated. Structural level is essential in diagnosis but teleological level, which is useful to discover the causes of faults, is out of the scope of diagnostic reasoning.

2.1 Phenomenological modeling

A phenomenon is a directly observable element of information about the actual state of the system to be diagnosed. It is usually modeled by physical variables, which have to be distinguished from parameters that are model-dependent.

Even if a phenomenon is an observable element, it is not necessary known. When information is available about a phenomenon v , the observed value is denoted $\textit{val}(v)$ ¹. The information is then represented by the constraint: $\textit{val}(v) = v$. This value may correspond to the phenomenon, but, sometimes, it only corresponds to the phenomenon providing that a component works properly. A Horn clause is then used to model this: $\textit{val}(v) = v \leftarrow \textit{ok}(c)$, which means that the equality is satisfied only if the component named c is in the state \textit{ok} ².

Moreover, the set of all the possible values for a phenomenon v is denoted $\textit{dom}(v)$.

2.2 Behavioral and functional modeling

Behaviors and functions relies on physical variables and possibly on observed values. They may be represented by constraints defined on a space generated by the involved variables. Let $V = \{v_1, \dots, v_n\}$, be a set of variables. Then, κ is a constraint over V if $\kappa \subset \textit{dom}(V(\kappa))$ where $\textit{dom}(V(\kappa)) = \textit{dom}(v_1) \times \dots \times \textit{dom}(v_n)$. The set V is then denoted $V(\kappa)$.

2.3 Structural modeling

Structural modeling represents the links between a set Σ of separated components or functions, and constraints, which rely on physical variables. In order to avoid the difficult task of precise modeling, structures of constraints are used instead of constraints themselves. A constraint κ may be summarized by its structure defined by $s(\kappa) = |V(\kappa)|$. "Structure of constraint" is a concept that has implicitly been used in different areas such as bipartite graphs when analytical redundancy relations are searched. Basically, an element of model can be represented by a Horn clause: $\kappa \leftarrow \textit{ok}(c)$, where c is a component and $\textit{ok}(c)$ stands for "the component c is assumed to be in a normal state". (Struss 1992) has shown that abnormal component state, such as $\textit{leak}(tank)$, may also be handled but this point is not discussed in this paper because diagnostic reasoning for complex

¹ $\textit{val}(v^*)$ may depends on time

² Observations coming from sensors and actuators require usually Horn clause for modeling.

industrial systems requires tools that do not lead to additional complexity.

Horn clauses are used because, generally speaking, it is not possible to prove that $\kappa \leftrightarrow ok(c)$ because the constraint has to be checked for all the tuples of $dom(V(k))$. In practice, a test usually corresponds to only one tuple i.e. to only one situation. Therefore, $\neg\kappa \rightarrow \neg ok(c)$ is the only proposition that can be proved; it corresponds to the Horn clause $\kappa \leftarrow ok(c)$.

Modeling industrial systems requires often to use validity or activating conditions. A validity condition represents the condition under which an element of model is meaningful. For instance, a “mixing process” can be meaningful if an operator has requested a product A , then, validity condition of the “mixing process” corresponds to the constraint “operator request = product A ”. It can be modeled by a new proposition: $\kappa_b \leftarrow ok(c) \wedge \kappa_v$ where κ_b is named behavioral constraint and κ_v , validity constraint. This proposition can be reformulated into a general form that distinguished constraints from state:

$$\kappa_b \vee \neg\kappa_v \leftarrow state(c) \quad (1)$$

With exoneration assumption, the modeling element becomes: $\kappa_b \vee \neg\kappa_v \leftrightarrow state(c)$. This modeling element can be reduced to its structure in order to lead to the following graphical representation:

where $s(\kappa_b) = |\{v_1, v_2, v_3, v_4\}|$ and $s(\kappa_v) = |\{v_5\}|$.

The structure of constraint leads to the representation of how components are connected.

3. PRINCIPLES

Automatic assessment of systems covers two issues: the generation of explanations (David 1998) and the design of detection tests, which can be achieved in connecting constraints. However, all these approaches require formalizations of expert knowledge, which is generally not compatible with industrial context. An iterative expert driven diagnosis method is proposed: it allows a step by step structural modeling, depending of the detection test results, exploiting at maximum the implicit expert knowledge.

3.1 Iterative expert driven approach

The aim of the iterative expert driven diagnostic approach is to avoid useless formalizations or, in other words, to make possible a just-in-time expert knowledge formalization. In order to reach

this purpose, expert knowledge has to be implicitly used. Let’s first reconsider the different modeling levels in order to identify the formalizations that require lots of efforts.

The phenomenological level models connections between components in a system. It is difficult for an expert to name these connections i.e. to formalize the phenomena related to a system. It should be avoided.

The behavioral level contains constraints that require lots of efforts to established: it should be absolutely avoided. As an alternative, only structures of constraints can be considered for formalization.

The structural level is the easiest level for an expert to detail because it corresponds directly to its reality composed of interconnected components. However, depicting all the components in detailed should be avoided when it is not necessary. An iterative description process should be possible starting from a macroscopic description and, then, according to the diagnosis results, detailing just the parts where faults are located.

So the iterative expert driven diagnostic approach assumed that implicitly, an expert is able:

- (1) to perform relevant detection tests to improve fault localisation when possible faults are known
- (2) to formalize the structure of the parts of the system that are tested i.e. the tested components and theirs interconnections modeled by structures of constraints.

It is difficult for an expert to summarize all the detection test results and to analyse them systematically in order to deduce all the possible faults. Usually, when problems are complex, experts use intuition based on experience to localized faults. However, even if this approach is usually successful, it involves a greater number of tests than it should be to discover a fault. Section 4 illustrates how the number of tests can be reduced using an automatic diagnostic reasoning.

Iterative expert driven diagnosis starts with a negative test i.e. a test that reveals a fault somewhere in a system. Expert depicts summarily the components and their interconnections that have been tested with a structural graph. Then, a consistency based diagnosis algorithm interprets the formalized expert knowledge and detection test results to provide all the possible diagnoses sorted according to criteria that are detailed in sub-section 3.3. Then, expert chooses another test that can improve the provided diagnoses i.e. either to confirm some faults or to clear some components. Then, when new tests are performed, some components may be decomposed into sub-

components. Therefore, the diagnosis algorithm reconsiders all the tests where the decomposed components were appearing and asks the expert if the different sub-components were actually involved or not in each test. This is the way the structural modeling is iteratively detailed in a relevant direction.

Nevertheless, modeling tests, especially in presence of validity conditions, has to be well formalized.

3.2 Formalizing tests

Assessing a system to be diagnosed is carried out in checking the consistency between observations modeled by values and behavioral and functional modeling elements. A test comes for a minimal set of behavioral constraints from which all the physical variables can be removed in order to obtain a testing function called analytical redundancy relation in (Blanke *et al.* 2003) and testable subsystem in (Ploix *et al.* 2005). Let $\tau = \{\kappa_{b,i} \leftarrow h_i, \forall i\}$ be a set of modeling elements with $h(\tau) = \{h_i; \forall i\}$ and $K_b(\tau) = \{\kappa_{b,i}; \forall i\}$. Then τ leads to a test if all the constraints $\kappa_{b,i} \in K_b(\tau)$ can be combined to produce a testing function:

$$\rho(\tau) = \bowtie_{\kappa_{b,i} \in K_b(\tau)} \kappa_{b,i} \quad (2)$$

where \bowtie corresponds to the join operator of relational algebra well known in database domain³. The following property can be established directly from the property $((\kappa_{b,i} \leftarrow h_i) \wedge (\kappa_{b,j} \leftarrow h_j) \rightarrow \kappa_{b,i} \bowtie \kappa_{b,j} \leftarrow h_i \wedge h_j)$:

$$\rho(\tau) \leftarrow \bigwedge_{h_i \in h(\tau)} h_i \quad (3)$$

where $h(\tau)$ is called: “support of the detection test”.

Under exoneration assumption, it is assumed that each modeling element can be written: $\kappa_{b,i} \leftrightarrow h_i$. Therefore, the detection test satisfies:

$$\rho(\tau) \leftrightarrow \bigwedge_{h_i \in h(\tau)} h_i \quad (4)$$

3.3 Diagnostic analysis

After each iteration, a diagnostic analysis is performed. It can be decomposed into two steps: the search of minimal diagnoses and the computation of scores for each diagnosis in order to sort them. In crisp logic, after each assessment, two sets of

tests are obtained: the set of negative tests that reveal faults and the set of positive tests.

The negative tests are used to compute diagnoses according to Reiter’s algorithm (Reiter 1987), which formalize logical diagnosis reasoning. Reiter’s algorithm has been chosen because its results are proved: it provides all the possible minimal explanations even if there are simultaneous faults. Indeed, proposition such as (3) cannot exonerate faults, therefore only minimal explanations for available observations can be provided. Consequently, one of the provided explanation is necessary true but additional faults may exist: they will be diagnosed after the repairing of the detected faults.

Nevertheless, Reiter’s algorithm has some drawbacks: the number of possible diagnoses can be great. Therefore, computation of scores used in (Touaf and Ploix 2004a) have been used to sort the diagnoses. Two scores assessing each explanation are used. The contextual score takes into account the positive tests. It corresponds to an Hamming distance between observed test results and the test results that should have been obtained for each diagnosis assuming exoneration i.e. that positive tests exonerate all the tested components. This score is just an indication because it can only be proved when all the situations (tuples in constraints) have been tested. This score is very useful in practice and human usually starts by exonerating components before reconsidering faults when no one diagnosis is found. An additional indicator is used. It is based on the unreliability⁴ of components coming from reliability theory (Gertsbakh 1988) assuming that reliabilities of components are independent. Each component can be characterized by a number between 0 and 1, which represents the probability for a component of being in a normal state. If reliabilities are not known, it is nevertheless useful to fix the same unreliability for each component: it means that two simultaneous faults are less probable than only one. This score is called a priori score.

Moreover, test results may sometimes be dubious. For such situations, a fuzzy logic extension of consistency based reasoning (Touaf and Ploix 2004b) is adopted.

4. ITERATIVE PROCESS

The concepts and principles mentioned in the previous section have been implemented into a command line software called *SMARTlab*. In order to depict the iterative diagnosis process, an example has been chosen. An expert had to face a problem

³ The join operator refers to the variable for which the join operation is achieved. However, this variable has been omitted because, in a testing function, all the variable has to be removed.

⁴ $1 - \text{reliability}$

Fig. 1. Structural graph of the expert

on an computer network with computers, printers, ethernet cables, routers, switches,... Expert has localized the problem without any tool after ten tests. Then, structural diagnosis requirements have been explained to the expert and, on a similar problem, it was asked to localize the faults on the computer network.

The process has been initialized because an user working on a computer called *PC1* cannot print on printer *PRN2*. Expert has drawn up 3 connected boxes *PC1*, *PRN2* and *NE1* in the structural graph (fig. 1) where *NE1* corresponds to a global network element. On *SMARTlab*, the trace was: *test PC1 PRN2 NE1 negative. SMARTlab* replies with the 3 following possible explanations⁵:

- (1) *apriori:10%, contextual:100%: PC1 is not ok*
- (2) *apriori:10%, contextual:100%: PRN2 is not ok*
- (3) *apriori:10%, contextual:100%: NE1 is not ok*

These results are obvious.

Next, expert has decided to print from another computer *PC2* in order to exonerate *PRN2*: a test page appeared on *PRN2*. Expert has decided to improved the structural graph in decomposing *NE1* into *NE11*, *NE12* and *NE13* in order to represent the different branch of the ethernet network. On *SMARTlab*, the trace was: *decompose NE1 into NE11 NE12 NE13. SMARTlab* replies with the following questions⁶ about the initial test:

- *Is NE11 in this test [y/n]? y*
- *Is NE12 in this test [y/n]? n*
- *Is NE13 in this test [y/n]? y*

Then, a new test is added: *test PC2 NE12 NE13 PRN2 positive. SMARTlab* replies with the 3 following possible explanations:

- (1) *apriori:10%, contextual:100%: PC1 is not ok*
- (2) *apriori:10%, contextual:100%: NE11 is not ok*
- (3) *apriori:10%, contextual: 50%: PRN2 is not ok*
- (4) *apriori:10%, contextual: 50%: NE13 is not ok*

So, the problem seems to be on *PC1*.

Expert decided to test *PC1* in sending a "ping" to router named *NE131*: a response has been obtained. Expert has decided to improved the structural graph in decomposing both *PC1* into *PING1* and *PRINT1* services and *NE13* into

NE131 and *NE132*. The trace was: *decompose PC1 into PING1 PRINT1*. Questions about the first test were:

- *Is PING1 in this test [y/n]? n*
- *Is PRINT1 in this test [y/n]? y*

And *decompose NE13 into NE131 NE132*. Questions about the first test were:

- *Is NE131 in this test [y/n]? y*
- *Is NE132 in this test [y/n]? y*

Questions about the second test were:

- *Is NE131 in this test [y/n]? y*
- *Is NE132 in this test [y/n]? y*

Then, the test is added: *test PING1 NE11 NE131 positive. SMARTlab* responds:

- (1) *apriori:10%, contextual:100%: PRINT1 is not ok*
- (2) *apriori:10%, contextual: 67%: PRN2 is not ok*
- (3) *apriori:10%, contextual: 67%: NE11 is not ok*
- (4) *apriori:10%, contextual: 67%: NE132 is not ok*
- (5) *apriori:10%, contextual: 33%: NE131 is not ok*

The actual fault was a problem of print service configuration on *PC1*, which is the most plausible diagnoses after 3 tests instead of 10 with expert intuition. Structural diagnosis helps expert in reasoning. Therefore the number of tests required to localize a fault may decrease. The expert knowledge modelled by structural graph are related to the expert conception. Therefore, diagnoses are also related to expert conception.

5. CONCLUSION

Thanks to a pure structural approach, an iterative diagnostic method has been designed. It is driven by experts that select the most relevant tests to be performed. After each iteration, experts only explain the performed tests and a diagnosis system compute and sort all the possible faults. It allows experts to iteratively enhance a structural model from a macroscopic description to a detailed one, only in the relevant direction.

A software called *SMARTlab* has been designed in order to validate this tool. Relevant graphical user interfaces will also be designed for industrial validation.

REFERENCES

- Blanke, M., M. Kinnaert and M. Staroswiecki (2003). *Diagnosis and fault tolerant control*. Springer.

⁵ unreliability is set to 10% as default

⁶ expert responses also appear

- Chittaro, L. and A.N. Kumar (1998). Reasoning about function and its applications to engineering. *Artificial Intelligence in Engineering* **12**(4), 331–336.
- Chittaro, L. and R. Ranon (2004). Hierarchical model-based diagnosis based on structural abstraction. *Artificial Intelligence* **1-2**, 147–182.
- David, R. (1998). *Random testing of digital circuits: theory and applications*. Marcel Dekker Inc. New-York, U.S.A.
- Davis, R. (1984). Diagnostic reasoning based on structure and behavior. *Artificial Intelligence* **24**, 347–410.
- Doggett, A. M. (2004). A statistical approach of three root cause analysis tools. *Journal of Industrial Technology* **20**(2), 1–9.
- Gertsbakh, I.B. (1988). *Statistical Reliability Theory*. Marcel Dekker company.
- Hawkins, P.G. and D.J. Woolons (1998). Failure modes and effects analysis of complex engineering systems using functional models. *Artificial intelligence in Engineering* **12**(4), 375–397.
- Hyatt, N. (2003). *Guidelines for Process Hazards Analysis (PHA, HAZOP), Hazards Identification, and Risk Analysis*. CRC press.
- Nayak, P. P. and A. Y. Levy (1995). A semantic theory of abstractions. In: *14th International Joint Conference on Artificial Intelligence IJCAI-95*. Montreal, Canada. pp. 196–203.
- Ploix, S., M. Desinde and S. Touaf (2005). Automatic design of detection tests in complex dynamic systems. In: *16th IFAC World Congress*. Prague, Czech republic.
- Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence* **32**, 57–95.
- Russell, S. and P. Norvig (2003). *Artificial Intelligence, a modern approach, 2nd ed.*. Prentice Hall.
- Struss, P. (1992). What’s in sd? towards a theory of modeling for diagnosis. In: *Readings in model-based diagnosis* (W. Hamscher, L. Console and J. De Kleer, Eds.). pp. 419–448. Morgan Kaufman.
- Touaf, S. and S. Ploix (2004a). Diagnosis for large scale distributed industrial plants - application to an hydraulic looper. In: *11th IFAC Symposium on Automation in Mining, Mineral and Metal processing, MMM04*. Nancy, France.
- Touaf, S. and S. Ploix (2004b). Soundly managing uncertain diagnostic analysis. In: *15th International Workshop on Principles of Diagnosis DX’2004*. Carcassone, France.