ON-LINE POSSIBILISTIC DIAGNOSIS BASED ON EXPERT KNOWLEDGE FOR ENGINE DYNO TEST BENCHES

O. de Mouzon,[1] X. Guérandel,[2] D. Dubois,[1] H. Prade,[1] and S. Boverie[3]

[1] *Institut de Recherche en Informatique de Toulouse*
IRIT – UPS, 118 route de Narbonne, 31062 Toulouse Cedex 4, France
{Olivier.deMouzon | Didier.Dubois | Henri.Prade}@irit.fr

[2] *Université Paul sabatier*
DRT ISI, UPS, 118 route de Narbonne, 31062 Toulouse Cedex 4, France
Xavier.Guerandel@free.fr

[3] *Siemens VDO Automotive S.A.S.*
1 Avenue Paul Ourliac, BP 1149, 31036 Toulouse Cedex, France
Serge.Boverie@siemens.com

Abstract Car engine electronic control unit tuning is a tedious and difficult task. Engineers can no more check on-line the measurements validity: Control strategies are becoming more complex and measurements more numerous. A fault diagnosis approach relying on experts' qualitative and uncertain knowledge has been developed to identify whenever a malfunction is present. The theoretical framework - based on possibility theory and on consistency and abductive indices - is first explained. The latest results are then presented. Finally, the whole incremental development process of the prototypes is described, explaining the most difficult points and how they were handled.

Keywords: Diagnosis, engine dynamometer, uncertainty, fuzzy sets, possibility theory.

Introduction

The paper presents a prototype implementation part of a long term research and development project, which aims at improving the calibration of car engine Electronic Control Unit (ECU), on engine dyno test benches, by detecting malfunctions when they occur.

The approach is based on a fault diagnosis system (named *BEST:* Bench Expert System Tool) that makes use of expert's knowledge formalized with fuzzy sets. In fact, since [Sanchez, 1977], much work has been carried out in

this field of fuzzy sets and diagnosis, and applied in different fields: E.g. in medical applications ([Buisson, 1999]) or industrial processes ([Ulieru, 2000]).

The paper is organized as follows: Section 1 describes the engine dyno context, the needs and expected benefits, and the architecture of BEST. The theoretical base on which it relies is presented in section 2. Then, section 3 presents the current prototypes and latest results. The discussion in section 4 points out the problems that were encountered and how they were solved. The conclusion indicates the last steps to be further developed.

1. Context and framework

In oder to better understand the results and the choices made during the project, this section first presents the engine dyno test bench context and then gives the general architecture designed for BEST.

Engine dyno test benches

Within one century, diesel and gasoline cars have gone from the carburettor to the electronic injection (ECU-controlled). More and more complex strategies have been implemented to answer to increasing constraints (pollutant regulations, vehicle behavior, new engine configuration: Direct injection, diesel common rail, variable valve timing, electric controlled throttle...) and an increasing number of variables must be taken into account.

For each new engine or new version, the control strategy parameters have to be calibrated in order to fit the requirements. This is done through a large amount of testing and tuning, starting on engine dyno test benches.

The calibration process. Figure 1 shows the ECU calibration process on an engine dyno, making use of a calibration tool. Basically, the ECU gets measurements from engine sensors (e.g., mass air flow, engine speed...), computes other intermediate variables and finally tells the engine which amount of fuel should be injected and what the spark advance should be. The data used for this are recorded through the calibration tool. They are provided by engine sensors and ECU but also by the engine dyno sensors, as well as additionnal devices. So the *System* to diagnose through BEST is: engine, engine dyno, all sensor sets and additionnal devices.

Data acquisition. Before doing any acquisition, the global conformity of the system has to be checked: Physical verification (sensors, fuel consumption measurements, gas analysis...), but also system configuration (i.e. inhibited system functions). As sensors become more numerous and the strategies more complex, this global check requires more time and so the tuning engineer has

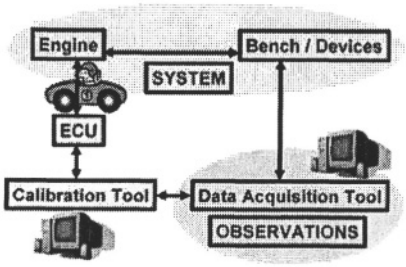


Figure 1 The calibration process

less time for the testing itself and its methodology. Hence test duration is increasing while reliability is degraded.

Then, during the tests, an on-line verification must be done to guarantee the acquisition validity. Again the tuning engineers used to check the validity of measurements manually while running: Single-parameter raw thresholds, coherence of some parameters with standard values, and sometimes coherence between several parameters and previous tests. Today it is nearly impossible for engineers to ensure the global coherence on-line, due to the complexity raise of parameters and strategies. Most of the time, when there is a problem, it is discovered during post processing data treatment. Quite often the test has to be performed again.

Needs and expected benefits. In fact, 10 to 20% of the manual tests must be reworked due to bad acquisitions, bad software configuration... Most of the time malfunctions are due to dyno bench environmental problems (gaz analysis, fuel balance...). So wrong acquisitions should be detected right away and the malfunction that occured should be identified as soon as possible in order to correct it quickly and make sound acquisitions again.

Some common and simple malfunctions are still identified by engineers. Yet others require time-consumming searches for their origin and symptoms whenever they occur. Indeed, engineers cannot keep in mind all the information and past experiences. Moreover they cannot watch in real time all the (numerous) measured channels.

In order to cope with such checkings, an expert system, BEST, has been considered. It has to perform global coherence checking, in the same way as for manual tests. It should also *detect* and *identify* malfunctions *as soon as* they appear: That is *on-line detection.*

General architecture for BEST

BEST represents a huge amount of work and investment. It gathers several functionalities divided into different modules for step by step development and validation (a key point in the success of the implementation, as discussed in

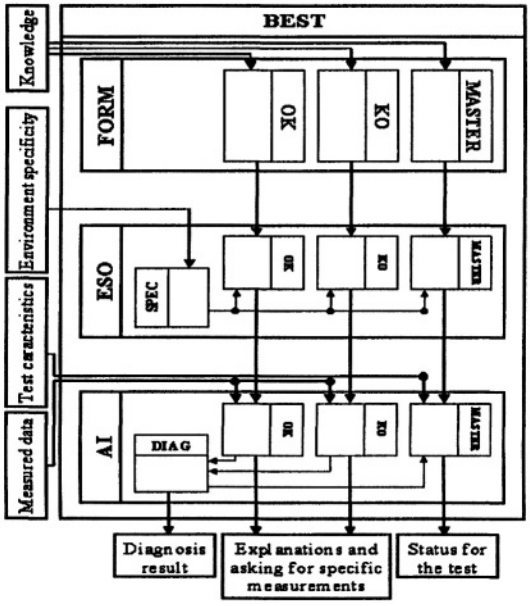


Figure 2 The architecture of BEST

section 4). The general architecture of BEST and the part for which prototypes have been developed are explained below. Some of the ideas underlying this architecture can be found in other approaches to industrial diagnosis (e.g., [Cordier et al., 2000]).

Figure 2 presents the different modules:

- **FORM:** Enables the experts to formalize their knowledge, using fuzzy rules.
- **ESO:** Extracts, Sorts and Organizes the rules w.r.t bench/engine specificities.
- **AI,** which is the Artificial Intelligence part performing the diagnosis: It compares the extracted rules to the measurements made on the engine.

Each module, contains the 3 following submodules: OK (diagnosis based on models of normal behaviour), KO (diagnosis based on models of malfunctions) and MASTER (the supervision rules). In the AI module, OK and KO diagnosis are aggregated in DIAG. BEST_AI_OK gives the models of normal behaviour which are not reached and BEST_AI_KO tells which malfunctions have been identified. The use of two different diagnosis is safer in regard to incompleteness and possible inconsistency of some models. Besides, both diagnosis (OK and KO) can give some feedback explaining why a model of normal behaviour was not reached and why a malfunction was selected. They may also ask for other specific measurements in order to improve their diagnosis. Finally, MASTER decides wheather the test may continue, should use another computation for some variables, or should stop. This decision is taken according to the supervision rules, the test characteristics (necessary variables), the

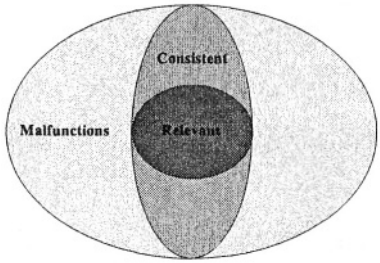


Figure 3 Diagnosis process with μ_{cons} and μ_{rel}

two diagnosis (pointing at the source faulty variables) and a dependency graph (which computes the induced faulty variables).

Prototypes has been developed for the whole KO part of this project, as the main objective was to build a demonstrative application which can detect and also *identify* malfunctions as soon as they appear on engine dyno test benches. This work is further described in the following sections.

2. Diagnosis fuzzy set based approach

This section presents the theoretical basis supporting the whole KO part of BEST. The knowledge relies on expert causal information. Possibility theory is used to capture the uncertainty pertaining to this kind of information. Observations may also be imprecise. Finally, the diagnosis makes use of two indices (see Figure 3). Refinements of these indices and ways to use them for more complex diagnosis (e.g. involving several malfunctions) can be found in [de Mouzon et al., 2001]. Here, only the basis is presented.

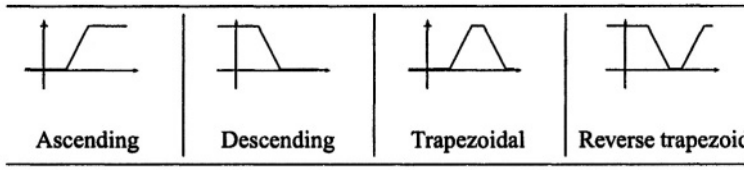
Notations. Let $\mathcal{M}$ be the set of all (known) possible malfunctions and $\mathcal{A}$ be the set of the n observable attributes: $\{X_1, \cdots, X_n\}$. Let $m \in \mathcal{M}$ and $i \in \{1, \cdots, n\}$, then π^i_m denotes the possibility distribution [Zadeh, 1978], that represents the (more or less) plausible values for attribute X_i when malfunction m (alone) is present. Let U_i be the domain of X_i, so $\pi^i_m : U_i \longrightarrow [0, 1]$. $\mathcal{K}^i_m$ will be the fuzzy set corresponding to possibility distribution π^i_m.

The observations may also be imprecise (or uncertain), $\mu_{\mathcal{O}_i} : U_i \longrightarrow [0, 1]$ is the possibility distribution, which represents the (more or less) plausible values for the observed value of attribute X_i. $\mathcal{O}_i$ denotes the fuzzy set corresponding to possibility distribution $\mu_{\mathcal{O}_i}$.

$\mathcal{K}^i_m$ and $\mathcal{O}_i$ both express imprecision, when they contain more than one element. Yet, they model different types of imprecision:

- $\mathcal{O}_i$ imprecision can be "controlled" (in principle): Changing the sensors for more precise (usually more expensive) or less precise observations.
- Imprecision on $\mathcal{K}^i_m$, on the contrary, cannot be reduced (or changed) that easily: It depends on the available knowledge about the *System* only.

Table 1. **Main possibility distribution shapes for knowledge formalization**



When attribute X_i is not yet observed, its value is not known, so it could be any value of U_i: $\forall u \in U_i,\ \mu_{\mathcal{O}_i}(u) = 1$. Similarly, when malfunction m has no known effect on attribute X_i, all values are allowed: $\forall u \in U_i,\ \pi^i_m(u) = 1$.

In fact, for the knowledge representation, the experts are very often more comfortable in expressing their confidence in some values they consider highly possible or, on the contrary, totally impossible. So, for continuous attributes, the experts only need to tell what they know best (values of possibility 0 and 1). Then, π^i_m is computed to follow the given information, be continuous and be piecewise linear. In the formalization process, the possibility distributions mainly used were increasing or decreasing ramp functions or trapezoidal ones, see Table 1. For discrete attributes, different levels of possibility (e.g., from 0 to 1 by step of 0.1 units) can be assessed.

A consistency-based index. A consistency-based index has been defined: $\mu_{cons} : \mathcal{M} \longrightarrow [0, 1]$, which enables to discard observation-inconsistent malfunctions ($\mu_{cons}(m)$ close to 0).

The consistency between observations and knowledge on malfunction m for a given attribute can be computed by $u \longmapsto \min(\mu_{\mathcal{O}_i}(u), \pi^i_m(u))$, which is the possibility distribution attached to $\mathcal{O}_i \cap \mathcal{K}^i_m$. The elements of highest possibility in this intersection give the *consistency degree between* $\mathcal{O}_i$ *and* $\mathcal{K}^i_m$: $\sup_{u \in U_i} \min(\mu_{\mathcal{O}_i}(u), \pi^i_m(u))$. The consistency degree of any malfunction m and the observations is computed from those of $\mathcal{O}_i$ and $\mathcal{K}^i_m$ (for each attribute):

$$\mu_{\mathbf{cons}}(m) = \min_{i=1}^{n} \sup_{u \in U_i} \min(\mu_{\mathcal{O}_i}(u),\ \pi^i_m(u)). \qquad (1)$$

In case of too incomplete information (knowledge and/or observations), μ_{cons} might not be sufficient in order to select a small enough number of malfunctions. So, a second index is required in order to refine μ_{cons} and bring a better conclusion: find, among the undiscarded malfunctions, which one is most relevant to the observations.

An abduction-based index. A malfunction is more relevant to the observations when its effects have been observed for sure. That is: $\mathcal{O}_i \subseteq \mathcal{K}^i_m$ (for

crisp sets). In order to single out most relevant malfunctions, fuzzy inclusion of $\mathcal{O}_i$ in $\mathcal{K}_m^i$ has to be defined.

Inclusion can be defined by implication (for crisp sets, $A \subset B$ is equivalent to $\forall x, x \in A \Rightarrow x \in B$). So, several fuzzy implications $(\rightarrow)$ have been checked for this purpose. Thus a second index is defined, which evaluates the relevance of a malfunction:

$$\mu_{\mathrm{rel}}(m) = \min_{i=1}^{n} \inf_{u \in U} \mu_{\mathcal{O}_i}(u) \rightarrow \pi_m^i(u). \tag{2}$$

The worst implication degree tells the extent to which the malfunction is relevant to the observations. Hence μ_{rel} selects the most relevant malfunctions ($\mu_{rel}(m)$ close to 1).

Dienes' strong implication $(a \rightarrow_D b \doteq \max(1-a, b))$ was chosen because it is the most dicriminating and it keeps the following natural crisp property: If $\mathcal{O}_i \subseteq \mathcal{K}_m^i$, then $\mathcal{O}_i \bigcap \mathcal{K}_m^i \neq \emptyset$. That is: $\mu_{cons} \geq \mu_{rel}$.

This index is abductive in nature: It selects m from the knowledge of the relation between cause m and effects, and from the observation of these effects.

Note that μ_{rel} is useful because observations are imprecise. Indeed, $\mu_{rel} = \mu_{cons}$ in case of precise observations. But a totally precise observation is feasible only for discrete attributes. In case of continuous attributes (e.g., temperature, pressure), the observations given by sensors always have some imprecision (even if it can be made very small with high precision sensors).

Hence, the diagnosis is first based on μ_{cons} in order to discard and rank the malfunctions and then on μ_{rel} in case of ties, as schematically shown on Figure 3. See [de Mouzon et al., 2001] for a more complete discussion on the use of fuzzy sets in this diagnosis process and the extension to multiple-fault diagnosis (not yet implemented).

3. Actual application

This section describes the current complete prototypes which have been (incrementally) developed for the single fault diagnosis on engine dyno test benches, based on the framework presented above. They implement the KO-part (see Figure 2): The first prototype is off-line and integrates FORM and ESO modules. The second is on-line and corresponds to the AI-module.

Off-line prototype: Knowledge formalization and selection

Knowledge is essential to diagnosis. Sometimes it can be computed from a database. None existed here, for reporting normal behaviour data and/or data under each specific malfunction. So, only expert knowledge could be considered. Hence a tool was needed to make both collecting and formalizing knowledge as easy and confortable as possible (BEST_KO_FORM module) for the engine dyno engineers. This creates a centralized knowledge base.

For ease of use, the same tool also enables to select from the knowledge base the pieces to be used, taking into account the state of the ECU calibration, the tests to make, and the test bench specificities. This generates the corresponding formalized knowledge file (BEST_KO_ESO module) directly used by the on-line diagnosis tool.

First, the FORM part contains friendly windows to express the knowledge step by step:

1) Express the malfunction definition: the user has to enter a unique definition with four characteristics: A group, a type, an identification and the nature of default (some choices are proposed for each). E.g., bench equipment [group] temperature sensor [type] 203 (inlet cylinder 3) [identification] dirty [default].

2) Give and structure the malfunction symptoms in an AND/OR tree.

3) Express each symptom through four elements: i) The attribute definition (function of the measurements and time). ii) A possibility distribution (as exemplified in Table 1). iii) Some conditions (engine running, ful load...). iv) An environment (bench and engine specificities).

Then the ESO part enables to easily select (or even modify) the knowledge to give to the diagnosis. When a new ESO file is created, the user has to parameter its environment. Only malfunctions of the knowledge base having compatible symptoms with this environment are automatically added into the file. For the incoherent symptoms, the user can choose to modify their environment in order to make them compatible or to remove them. The generated files are unreadable for the expert, which ensures that they will not be modified by hand. When necessary, modification can be done using this off-line application.

This tool has been a key point of the development of BEST, as knowledge is a root of diagnosis. The FORM part has enabled to express and formalize more than 250 malfunctions (about one third of the estimated total number) in 21 environments (about 75% of all). The ESO part not only supports knowledge selection for each specific environment, but also enables to test and validate knowledge immediately on the on-line diagnosis tool.

On-line prototype: Diagnosis

This prototype is an on-line tool: it reads the data from the engine dyno test bench (through a Dynamic Data Exchange connection) while the calibration is tuning, and warns as soon as possible when a malfunction is detected (in less than 3 minutes today). There are two levels of information:

i) An efficient diagnosis mode (a small icon sits in the system tray of Windows NT taskbar): While performing on engine dyno test, the prototype becomes active and works without human intervention. When an event occurs, i.e. detection of the possible presence of a malfunction, a popup window is

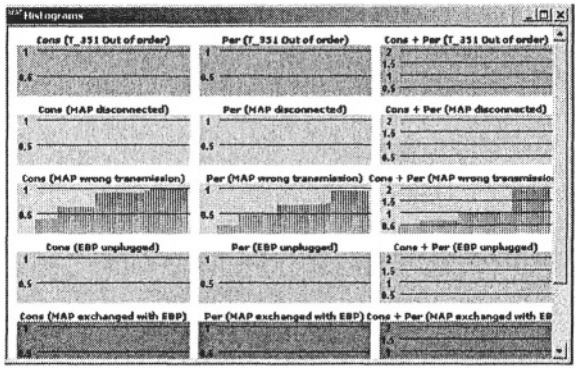


Figure 4 Bargraphs for diagnosis indices (μ_{cons} [Cons] and μ_{rel} [Per]) trace in time for each malfunction ("debug" mode)

automatically opened and gives some information and advice on malfunctions and symptoms involved in the problem. In order to avoid blinking problems (with oscilating values near a single-limit), a malfunction is considered as appearing when μ_{rel} goes over 0.7 and disappearing when μ_{cons} goes under 0.3.

ii) A debug mode, displaying a bargraphs window (see Figure 4) and a console window where each event (attribute calculus, intermediate results, false alarms,...) is logged and can be saved on a text file.

The diagnosis application runs directly on the engine dyno test benches. Experiments have been conducted on-line on the benches in order to validate the method and the application. These experiments have been made on a representative case study involving 35 possible malfunctions (about 5% of all). Figure 4 gives an example for a few malfunctions where the one that occured is just being identified (here, "rel" reads "per").

The results of these experiments are satisfying: All malfunctions have been detected and identified on-line. Very little false alarms occurred: Only for temperature sensors during sharp transition phases. This was then corrected in the knowledge base. Hence, this diagnosis method has shown its usefulness. Some future steps are now given below, towards a complete industrial application.

Future work

Observation validity. Some conditions might be necessary to compute the presence/absence of a symptom (e.g., engine stopped). Thus, at some moment only part of the information has been computed. In order to be able to make a diagnosis despite this problem, the easiest way is to make the hypothesis that former computations are still valid. This often holds and always does in demos (where the conditions to observe the changes are known). Yet, for a real on-line use, this hypothesis might not hold when a fault occurs.

A solution could be to have a validity index (for each piece of knowledge) that would depend on the time elapsed since the latest computation (of that

piece of knowledge). Then a too old information would not be taken into account when it is contradictory to other more recently computed pieces of knowledge. On the contrary, if they still give the same information, this could also be taken into account as giving more support to the overall result.

Observation errors. For the moment, only a general evaluation of the error is computed. In fact, the real error (attached to each sensor) should be used.

This error is usually modelized by a probability distribution (as a gaussian). Yet, a possibility distribution like a fuzzy number (centered triangular distribution, with the maximum error as support) would be very convenient: It covers any symmetric probabilistic distribution on this support, [Dubois et al., 2002].

Besides, this solution is computationnally much more simple when it comes to computing the error of a complex attribute from the channels it combines.

4. Incremental development

The previous section presented the prototype complete version of BEST_KO. The current section explains the problems which had to be tackled and the development process which was adopted: A successful incremental approach. The first step was to analyse the type of knowledge accessible for the diagnosis. The second step was to test off-line the diagnosis method (see section 2). The third step was to test it on-line. The last step was to add a complete and more convenient tool for knowledge formalization, as presented in section 3. All prototypes were developed in Visual C++ V6, under Windows NT4.

First step: Knowledge analysis – Towards formalization

BEST_AI_KO (Figure 2) is based on human beings' experience. Thus, recording and formalizing the available knowledge is a key step of this project, [Kahn et al., 1985]. So a preliminary standard form (Figure 5) had been defined to describe the malfunctions (under the assumption that it is present alone):

- Number (cell 4) – Name (cell 3): Identifying the direct source of observed anomalies). E.g., sensor out of order, badly calibrated sensor...
- Number (cell 6) – Environment (cell 5): For bench or engine environmental specific conditions. In some cases the same malfunction will generate different anomalies depending on the environment: E.g., a sealing defect in the intake system will have strongly different effects, depending on the sensor used on the engine: MAP (mass air pressure) or MAF (mass air flow).
- Description of symptoms of the malfunction may be added:

i) Attribute (cell 7, including observation conditions, if needed). Example for the "exhaust temperature sensor out of order" malfunction: Difference between this temperature and the other cylinder temperatures, when the engine is running... Note: Attributes may be as complex as needed.

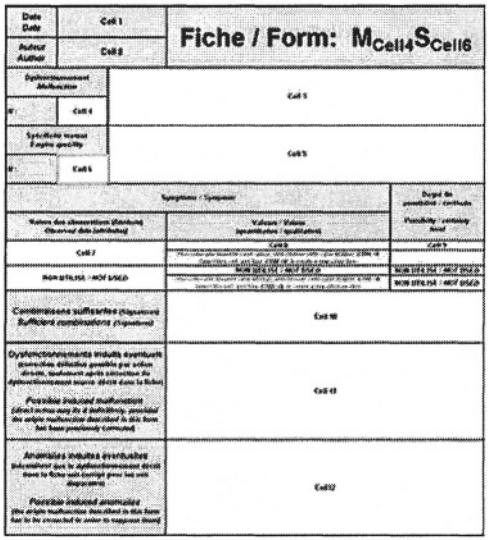


Figure 5 The first form

ii) Finally (cells 8 & 9), a possibility distribution is attached to each attribute, expressing to what extent a value is consistent (possibility 1) or not (possibility 0) with the presence of the malfunction. This makes it easier for the engineers to express which values they know possible, the ones they know impossible and the ones they are not sure about: In practice, engineers are asked for the core and the support of the possibility distribution. For the uncertain values, the possibility is calculated to be linear. The more the value is close to a possible value, the higher its possibility level. The more the value is close to an impossible value, the lower its possibility level (see Table 1).

- Last but not least, the form includes information about induced malfunctions ("in cascade", cell 11).

Of course, not all the knowledge on the malfunctions was asked for at that time, but at least a significant fragment of it was obtained in order to get an explicit structure concerning the type of information BEST would have to deal with. This first form made possible to collect knowledge on about 10% of all the malfunctions and 20% of all the environments. The knowledge cannot be exhibited here. But, the analysis of the filled-in forms led to these conclusions:

- Pieces of knowledge for a complex symptom may use connectors (and, or) between more elementary pieces of information.
- Those elementary pieces of information may also use any shape of function for representing the attribute value.
- Finally, there might be some conditions to be satisfied in order to be able to compute a valid result for presence/absence of a symptom (such as: engine cold, engine stopped, closed throttle, high engine speed...).

So, the knowledge collected from the forms could be formalized in a structured text file. Of course, not all the knowledge on the malfunctions was used at that time: This step is really time consumming and some basic tests about the diagnosis approach were first needed before going further in that direction. Efforts of formalization were first made on a case study, large enough to be representative of the kind of malfunctions to be taken into account in this field of application: The case study considered malfunctions that have an incidence in the engine gaz circulation. And to start on some real basic testing, the formalized knowledge was even restricted to some 15% of the case study.

Second step: Off-line diagnosis

This first diagnosis prototype was implemented for a preliminary evaluation of this approach applied to ECU calibration on engine dyno test benches.

The knowledge was given through the structured text file discussed above. The data used for off-line tests were yet directly measured on the engine dyno test benches. They came from files where the measurements had been stored.

To reduce complexity for the first tests, data in each file had been collected at a stabilized running point. Transition phases were kept for later. Files were created for different running points, representative of the calibration process: engine speed and MAP table, and also richness and spark advance table.

Each file contained at least 131 measured channels during a 5 minute (or more) time period. The time frequency for measurements was every 100 ms. So each file contained at least 400 000 values (differents files for nominal behaviour and for each malfunction). The total number of files used for the tests was 181. For confidential resons, data are not given here (but the numbers given above give some indication on the complexity of the problem) and the results aim only at providing feedback on the implementation process.

Note that the first evaluation could not state whether this approach was good or not: The conclusion was that off-sets and noise (mainly 50 Hz) had first to be filtred. Then, the results of the tests were computed once more. As a general remark, too little knowledge was used for testing this diagnosis approach: 1) Some attributes were too complex to be included in the first tests. 2) Others needed special measurement equipment or conditions not possible at that time. 3) Too little knowledge had been collected. 4) Last but not least some pieces of knowledge turned out to be false. Nonetheless, some tests could be carried out. In fact, this situation is rather logical: Great efforts cannot be asked for with no clue on whether they will be worthwhile. Finally, the very first tests showed that this approach to BEST is all the more efficient as more formalized and validated (e.g. through some experiments) knowledge is provided. For instance, note that no false alarms were raised during normal behaviour and that they were all the less frequent as more specific symptoms were formalized.

Third step: On-line diagnosis

The off-line evaluation not only was good enough to go forward for the on-line version, but also gave even more resons to do so: The necessary knowledge validation is much easier if it is possible to test/change right away pieces of knowledge. Besides, transitions phases had to be taken into account (as they are frequent on-line) and thus BEST would take advantage of observations gathered through out the whole calibration process, going from one running point to another. This is even more needed, as more knowledge is asked for, which means more symptoms that may need specific observations conditions.

In fact, the knowledge used took benefit from the important validation task pointed in the previous section and from the more numerous measurements accessible on-line. Besides, other functions had been implemented to take into account more complex attributes. Finally, transition phases were taken into account so that observation of several stabilized running points was possible, as well as the transition phases themselves. Hence, the previous malfunctions were described both more properly and in greater details.

But the main change in this prototype is that data are now read directly (using a Dynamic Data Exchange link) from the data acquisition tool (see Figure 1). As the acquisition tool has time priority, our prototype only reads data every 500 ms (formerly 100 ms) in order to keep enough time for μ_{cons} and μ_{rel} computations. All those operations are very little time consumming, but the constraints are very high. Time could also be gained with less feedback, but such a feedback (see Figure 4, for instance) is essential during the development process.

Another point to keep in mind is that only the measurements specified for the data acquisition tool are seen. In other words, there may be up to more than 200 measured channels or less than the 131 former ones. Thus, it may be necessary to add some channels during a calibration (that usually does not need them) for the sake of diagnosability.

An example of result is given in Figure 4. This section gives the main issues for a complete 20 minutes demo, testing the prototype on-line and even for some multiple malfunctions.

The results are better with this on-line prototype: Each malfunction is detected when it occurs and there are no false alarms for normal behaviour. This is mainly a benefit from knowledge validation. Besides some multiple (non-dependant) malfunctions were successfully tested. On top of that, all malfunctions are detected within 10 s time after they occur. Yet, this should be reconsidered, as, in the demo, the running points which are important to reach to observe the symptoms are known. It is very likely that a malfunction could be ignored for hours otherwise. This is one more point that asks for more knowledge, so that at least the presence/absence of one symptom may be computed

for each malfunction, whatever the running point. Moreover, the malfunctions that still have too little knowledge provided, give some false alarms. More specific knowledge is needed to avoid them.

5. Conclusion

This paper has presented the prototyping results of a long term project in fault diagnosis applied to engine dyno test benches. Thus, it gives the whole evolution from the very first prototype to the actual. It takes into account not only the diagnosis process, but also all the necessary knowledge considerations for such a tool to be effective. It explains the results obtained at each stage, the problems we encountered and how they were solved.

This incremental process is one of the key of the success of this project. The begining of the project was quite unsure as huge amount of work had to be carried out, including promoting BEST inside the company. Today, a full prototype is implemented, from knowledge formalization to malfunction detection and the database contains about half of the malfunctions.

Some work still need be done, as implementing the real observation validity and errors. The extensions of the approach presented in [de Mouzon et al., 2001] should also be implemented to allow for multiple and cascading malfunction diagnosis.

References

Buisson, J.-C. (1999). *Practical Applications of Fuzzy Technologies,* chapter Approximate reasoning in computer-aided medical decision systems, pages 337–361. The Handbooks of Fuzzy Sets Series. Kluwer Academic, Boston/London/Dordrecht.

Cordier, M.-O., Dague, P., Dumas, M., Lévy, F., Montmain, J., Staroswiecki, M., and Travé-Massuyès, L. (2000). AI and automatic control approches of model-based diagnosis: Links and underlying hypotheses. In *Proc. of the* 4^{th} *IFAC Symp. on Fault Detection Supervision and Safety for Technical Processes (SAFEPROCESS'2000),* volume 1, pages 274–279.

de Mouzon, O., Dubois, D., and Prade, H. (2001). Twofold fuzzy sets in single and multiple fault diagnosis, using information about normal values. In *Proceedings of the* 10^{th} *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2001),* volume 3, Melbourne.

Dubois, D., Foulloy, L., Mauris, G., and Prade, H. (2002). Probability-possibility transformations, triangular fuzzy sets and probabilisitc inequalities. In *Proc.* 9^{th} *Int. Conf. Info. Processing and Management of Uncertainty in Knowledge-based Syst.,* pages 1077–1083.

Kahn, G., Nowlan, S., and Mc Dermott, J. (1985). Strategies for knowledge acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),* 7(5):511–522.

Sanchez, E. (1977). Solutions in composite fuzzy relation equations: application to medical diagnosis in Brouwerian logic. In Gupta, M. M., Saridis, G. N., and Gaines, B. R., editors, *Fuzzy Automata and Decision Processes,* pages 221–234. North-Holland, Amsterdam.

Ulieru, M., editor (2000). *Intelligent Manufacturing and Fault Diagnosis (II). Soft computing approaches to fault diagnosis,* volume 127 of *Information Science.* Elsevier. Special issue.

Zadeh, Lofti Asker (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems,* 1(1):3–28.