

# IDD: Integrating Diagnosis in the Design of automotive systems

Authors

**Abstract.** In this paper we overview the achievement of the IDD European projects, which aims at defining a new framework for design of automotive systems. In particular, starting from the weakness of the current design process, especially as regards issues related to diagnosis (diagnosability analysis, generation of the FMEA – Failure Mode Effect Analysis, generation of on-board diagnostic rule), the project aims at defining a new process in which this issues are integrated in the core part of the design of system and of its control strategies. The project also aims at defining and implementing a software toolkit supporting this new design process, toolkit which integrates modules for design and simulation (e.g., Matlab Simulink) and model-based reasoning systems.

## 1 INTRODUZIONE

The importance of diagnosis in onboard automotive systems is constantly growing together with the complexity of the systems. The average dimension of the diagnostic software inside a modern electronic control unit (ECU) is now more than 50% of the whole code. However, if one analyses the design process of any significant mechatronic automotive component, it is very common to see that diagnostic issues are usually taken into account only at the end of the process and are not integrated with the rest of the process. In particular, during the critical phases of the design process, when the actual architecture of the system is conceptualized, the control strategies are defined and models or prototypes of the system are simulated, diagnostic issues are not taken into account. Not only does this mean that the diagnostic software is not developed together with the control software, but, more critically, that issues such as the diagnosability of the system being designed or the analysis of the FMEA (Failure Modes Effect Analysis, which is very useful to discover safety critical faults or failures) are seldom and only partially considered. Usually, these activities are performed by separate teams and most of the time after the design decisions have been made. And in many cases they diagnostic team to face serious problems since the pieces of information that are needed for diagnosing a system (i.e., the sensors in the system) are very different from those that are needed to control a system. However, as the design has been almost completed there is no opportunity to ask for modifications (e.g., addition or replacement of sensors) so that compromises have to be made in the development of the diagnostic software. Let us consider the case of on-board diagnostic software, i.e., software that has to decide about recovery actions whenever there is an evidence of a fault in the system under examination. If the system is not diagnosable, that is, whenever there is a symptom it is not possible to detect the fault (which in practice means that many faults are possible), the recovery action to be taken is the strongest

one, that is the one which guarantees safety in the worst possible case. This is clearly a non-optimal but necessary choice and has a negative impact on aspects such as the availability of the vehicle and the satisfaction of the driver (customer). We experienced this problem directly in the VMDB project, when applying Model-based Diagnosis to the “Common rail injection system” (see [2]). Due to the limited number of sensors available on-board, in many cases it was impossible to discriminate between very critical faults (e.g., a blocked injector; in this case the engine has to be stopped immediately to avoid serious damages) and less critical ones (a slipping belt; in this case it is sufficient to warn the user, possibly limiting performance and suggesting a check at the closest workshop). Being unable to discriminate, the diagnostic software could only perform the strongest action (stopping the engine) to avoid the worst case problems. This is clearly a bad situation, especially for the driver but also for the car manufacturer for which the driver is a customer to be satisfied. The European Fifth Framework project “Integrated Design Process for onboard Diagnosis” (IDD)<sup>1</sup> pursues the goal to formalise and standardise the diagnostic design process, and to enable the introduction of diagnosis early in the chain. This methodological goal has to be combined with another important objective: giving to the designers a set of tools that can help them in evaluating and understanding the effects of each choice on the system being designed.

In particular, we claim that the Model-based approach to diagnosis (see [5]) is suitable for this integration, both from the methodological and the practical point of view. In fact, the basic modeling principles adopted in the process of designing a system and its control have some similarities with those adopted in model-based diagnosis. In the project we exploit these similarities to define a new design process and to develop supporting tools.

The discussion of this process and tools is the main goal of this paper, which is organized as follows: in Section 2 we sketch the current design process, discussing the marginal role played by diagnosis related activities; Section 3 proposes a new design process which integrates these activities; Section 4 discusses modeling issues while Section 5 introduces the prototype architecture we are developing for supporting the new process; Section 6 overviews the guiding applications on which we are working; Section 7 concludes the paper listing some open problems for future research.

---

<sup>1</sup> GRD1-1999-11263, partners: Centro Ricerche Fiat (CRF), DaimlerChrysler, Magneti Marelli, OCC’M Software, PSA Peugeot Citroen, Renault, Technische Universität München, Université de Paris Nord (XIII), Università di Torino.

## 2 THE CURRENT DESIGN PROCESS

A major effort in the first phase of IDD has been devoted to the analysis of the current design process. We analysed the design departments of the industrial partners, considering different types of systems, interviewing several people working in these departments and analyzing documents (guidelines) that describe the process organization. The goals of this analysis can be summarized as follows:

- First of all we aimed at having a general view of the process as a whole, singling out the different phases that lead from the conceptualization of a new system to its implementation and test.
- Second, we aimed at clarifying when different types of decisions are taken; in particular, how requirements are specified, when and how the layout of the system is decided, when and how the system control is defined; in which phases the designers use model for simulating the system; how and when the results of the simulation produce modifications to the design decisions, ..
- Third, we aimed at singling out precisely when diagnosis related issues are taken into account. By diagnosis related issues we mean, for example, when and how are diagnosability issues considered? Do sensor selection and placement take diagnosability into account? when and how is FMEA (Failure Mode Effect Analysis) performed? When and how is the diagnostic software developed? Is this development related with the development of the control software?
- Finally, we aimed at detecting which tools are used in the process and, in particular, when and how models of the system are developed and used.

As a result of the analysis, it turned out that the processes adopted by the industrial partners were very similar; so similar that we could define a sort of “reference current process” to be used in the project for singling out the problems and weaknesses of the current process and to define requirements for the new process.

The results of the analysis can be synthesized as follows.

- The process is divided in three phases: “strategic”, “technology and integration”, “production”; our main focus will be on the second, and specifically on the “technology” phase
  - The structure of the **technology phase** is depicted in figure 1. There are three main steps. Step **A** defines the **specification** of the system; step **B** involves selecting and laying out components; step **C** concerns the definition of a control strategy and the simulation of both the system and the control strategy (possibly on a prototype). These steps can be repeated following one of the three loops that appear in figure 1. The inner loop involves the redesign of control strategies; the middle loop involves also the redesign of components and their layout; the outer loop involves a revision of the whole technology phase.
- As to diagnosis-related issues, it turned out that in most of the cases the generation of the FMEA and the

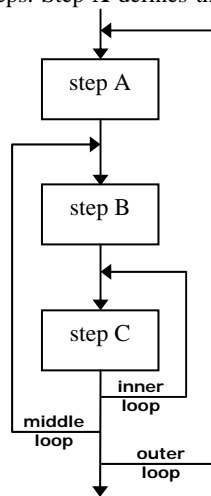


Figure 1.  
the technology phase

diagnosability analysis are performed by different departments and interact in a very loose way with the steps of the design process. Moreover, FMEA and the development of the diagnostic software are sequential activities and the latter is performed at the end of the overall process.

- Various modeling tools are used throughout the process, mainly tools for building control models and engineering models of the system being designed. The tools which is most commonly used in this activity is Matlab/Simulink.

The analysis of the “reference process” evidenced some weaknesses, especially as regards the role of diagnosis related issues. Very roughly, these problems can be summarized by the following three items:

1. FMEA and diagnostic development are sequential activities and they are mainly performed using experience and without supporting tools.
2. Usually development of FMEA and diagnostics is carried out in parallel with control design (step **C** of the technology phase); however the two activities do not interact and if additional requirements or constraints emerge from one of the two tasks they are taken into account in the other one only when (and if!) whole step **C** is revised (that is, when an inner loop is performed), while they could be dealt with immediately due to parallelism.
3. Since FMEA and diagnostics development is carried out **after** component design/layout (step **B** of the technology phase); therefore diagnostic-related issues have an impact on system design only if a middle or outer loop is performed. This does not often happen, and usually diagnostic issues alone are not enough for choosing this (expensive) option.

This analysis led us to the definition of a new design process, in which the various activities are more tightly integrated.

## 3 TOWARDS A NEW DESIGN PROCESS

Based on this analysis of the reference process and the outlined improvements, we propose a framework for a new process which is closely connected with a new tool architecture.

In summary, the framework for a new process has to satisfy the requirement that in the inner design loop of the process, the designers (the different experts involved in the design) should be supported in performing different activities in an interleaved way:

- design of the physical system;
- design of control algorithms, and their simulation (for quantitative analysis);
- generation of the FMEA of the designed system
- analysis of the diagnosability, i.e. investigation of which faults are detectable and discriminable from each other;
- derivation of on-board diagnosis (OBD) software for the system;
- comparative analysis on the current design (physical system and control), i.e., analysis of the consequences of applying changes to the design (e.g., changing the layout or some components) both from the control and diagnosability point of view.
- comparative analysis of different design alternatives

Thus, designers and decision makers are supported in the process of evaluating trade-off between different designs and in making choices about the best design of a system.

With respect to the current process sketched in the previous section, the two internal loops are integrated and the activities in the inner loop are no more separated (between design and

FMEA/diagnosis) or sequential (FMEA and diagnosis), but are completely interleaved.

Figure 2 sketches a conceptual organization for the internal loops of the new process.

A central role in this new process is played by the model of the system being designed, which is the core of the whole process and must support both the types of analysis needed by control design and simulation (bottom part of the figure) and those needed by model-based diagnosis/FMEA generation and diagnosability analysis, On board Diagnostic (OBD) software generation. Indeed, as we shall discuss in more detail in the next section, modeling plays a central role both in design and in diagnosis and some of the modeling principle are similar, although the models are very different.

#### 4 MODELS FOR CONTROL AND DIAGNOSIS

We noticed in the previous section that models play a central role in the definition of the new design process. In this section we analyse the types of models that are currently being used in the two main activities to be integrated: (i) design of the system and of the control and simulation (we shall refer to them as “design models”) (ii) diagnosis related activities (we shall refer to them as “diagnosis models”). The models that are generally used have some similarities but also some very important differences; these

example, a pump may have an hydraulic interface which may be connected to a corresponding hydraulic interface of a pipe.

Each component has an associated model, which is formed by a set of mathematic relations. In particular, the model relates a set of interface variables (associated with the interfaces, e.g., a flow variable associated with an hydraulic interface), plus possibly a set of variables that are internal to the component (some of these variables define a notion of “state” for the component, whose models, in most of the cases have a dynamic nature).

What makes design and diagnosis model completely different is the nature of the variables and of the mathematic relations constituting the models.

While design models are usually quantitative (which means that the variables range over real numbers), diagnosis models are usually qualitative. By qualitative model we mean models in which the variables assume a discrete set of qualitative values (for example, we may consider only three values “positive”, “negative” or “zero”). Consequently, the models can be regarded as a set of relations (or constraints) between these variables. For example, a very simple model of a pipe could say that the derivative of the pressure inside a pipe is positive whenever the input flow is greater than the output flow.

Why does diagnosis adopt qualitative models rather than quantitative ones? There are several reasons that support the use of these simpler models, showing that they are sufficient for performing diagnosis. Let us briefly recall the basic principles of

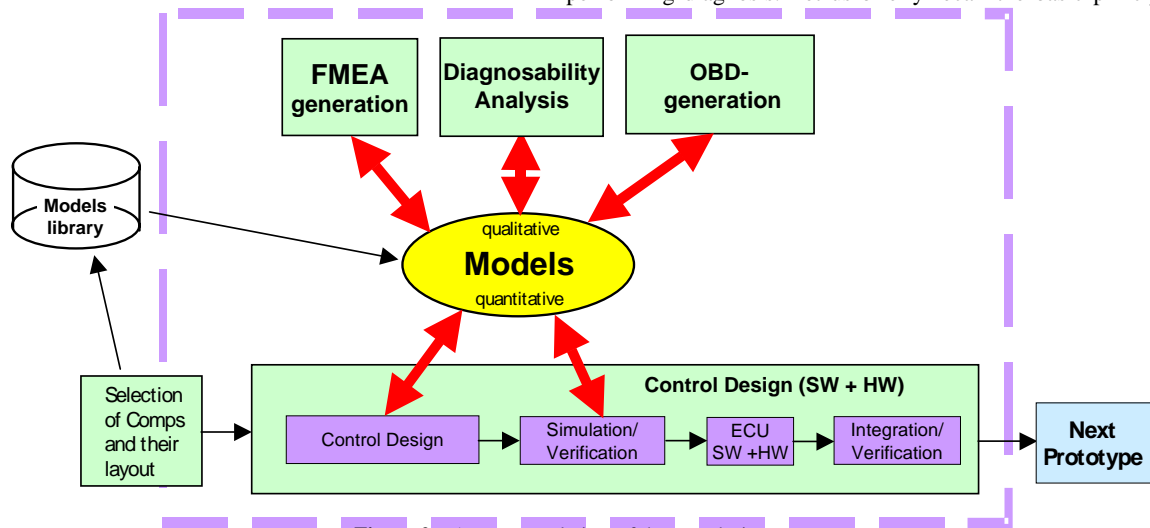


Figure 2. A conceptual view of the new design process

similarities and differences generate at the same time some opportunities and some problems.

Let us start with the similarities.

First of all, both design and diagnosis models are based on a “component centered” philosophy and are compositional. This means that the building blocks are the models of components, which are stored in a database called model library. What counts as a component depends on the goal of the process: in some cases one may be interested in low level components (e.g., a wire or a pipe); in other cases in higher level components (e.g., a pump or an ECU). In both cases modeling can be performed in a hierarchical way, that is one may have model of high level components which can be decomposed by singling out sub-components which can in turn be modeled.

Each component is characterized by a set of interfaces, which can be used for connecting components (each interface is characterized by a physical type, e.g., hydraulic vs. electric). For

Model-based Diagnosis. The idea is that one can use a model to make predictions about the behavior of a system, then diagnoses can be computed by comparing these predictions with the actual observations. In this way one can isolate the faulty components (in case only the correct behavior has been modeled) or can identify the faults of the components (in case some information about the faulty behavior has been modeled). In most of the cases qualitative relations are sufficient to perform fault isolation or identification. Moreover, these models are easier to simulate (and often more efficient). Notice, moreover, that while it is not very complex to produce qualitative fault models, in most cases it would be impossible to build quantitative fault models. Finally, observations are usually known with some error, which would make quantitative modeling and simulation less useful than qualitative one.

On the other hand, qualitative models would not be sufficient for design and, especially for control design and simulation, for which very precise models are needed.

This means that in order to support the different tasks constituting the new design process, two types of models are needed: quantitative ones and qualitative ones. There is a long tradition in engineering and control theory for the development of the former types of models; various methodologies and languages have been proposed in the last decades in the AI community for developing the latter type of models. In particular, these different methodologies rely on different modeling assumptions [4][8] i.e., on different abstractions and thus each one of them is suitable for some applications and problems. Recent research on automotive application showed that simple models, based on signs and possibly involving qualitative deviations can be sufficient for achieving interesting diagnostic capabilities [6][2][3]).

These models can be clearly developed manually, building in such way two separate libraries: the one with quantitative models and the one with qualitative models. Although this approach can be reasonable for demonstrating the advantages of the new design process, in the long run, we believe that it is not feasible to require two separate modeling activities. In particular, we believe that none would develop qualitative models. Consider that the new process and the corresponding tools would be mainly used by designers and engineers, for whom it would be difficult to develop these models and that actually have no interest in these models and, more generally in diagnostic issues; if we want to convince them to adopt the new process we have to guarantee that this will not cause extra work or costs.

For such a reason we are devoting many efforts to the problem of the automatic derivation (or, at least to provide a lot of support in the generation) of qualitative models from quantitative ones. This is not an easy task since the kind of quantitative models used by designers and control people have very different formats; in most of the cases they are even not defined intensionally (i.e., as a set of equations) but are defined extensionally (via tables) or by software code (programs in C language). This means that the only approach that can work in all cases is to perform numerical analysis of the models, trying to determine their qualitative properties (e.g., monotonicity - on a given range -, local maxima or minima, sign of the function).

Currently we are experimenting some alternative methodologies to perform such a derivation and indeed in some simple cases we can obtain qualitative models suitable for the diagnostic task. Still a lot of research has to be carried on to produce a solution that works in general, for any type of quantitative models. We also believe that this will lead to the definition of some guidelines for quantitative modeling, i.e., to a sort of discipline in order to prevent the construction of models that would create problems (e.g., there is no constraint that imposes that quantitative models are only component centered and it would be possible, for example, to introduce integration components, that is state variables and state constraints, external to components; this would clearly be critical for generating component centered qualitative models). The definition of these guidelines is one of the goals of the project.

The automatic derivation of qualitative models is a fundamental opportunity for model-based reasoning and diagnosis and is, in our view, the key for a wider diffusion of these technologies (see also [3]).

A final remark to conclude this section. Another standardization goals of IDD was that of defining some interchange format for representing models and model libraries. In order to achieve these goals we defined an XML format in which we represent the information extracted from quantitative models (we developed some software tools for extracting various types of information, e.g., the structure of the device being designed, the connections

between the components, from Matlab/Simulink internal representations).

## 5 A TOOL FOR SUPPORTING THE NEW PROCESS

In order to support the new design process discussed in Section 3, we developed a set of software modules which implement the functionalities introduced in the previous sections. We conceived these modules not as a new tool, but rather as plug-in to be added to the tools currently used in the design process. In the prototype we are implementing, in particular, the modules are defined as plug-ins that can be activated from the Matlab/Simulink environment. That is, starting from the core of Matlab/Simulink, which can be used for quantitative modeling and simulation, and which is commonly used to model systems and their control and to simulate their behavior, we defined the following modules:

- A tool for the automatic generation of qualitative models; in particular, the tool is composed of two modules:
  - A module that extracts the structural description of a device, i.e. the list of components (with information about the component type of each instance, that is two pipes are recognized as two instances of the same component type) and of their connections.
  - A module that derives, for each component type in the library, a qualitative behavior model starting from the quantitative one. As noticed in the previous section, we only have a partial solution for this task and we are working to extend it.
- A tool for performing diagnosability analysis.
- A tool for supporting the FMEA generation.
- A tool for supporting the generation of onboard diagnostic strategies and software.
- A tool for comparing alternative design options.

The last four tools are based on a common Model-based diagnostic system (core MBD system), which in the prototype is based on the Occ'm Raz'r system [9].

More specifically, the four tools rely on the basic functionalities of the MBD core in the following way:

- The tool for diagnosability analysis allows the user to specify an operational context and a set of sensors and to perform tasks such as:
  - determining if two (a set of) faults can be discriminated;
  - determining if a fault can be detected;
  - determining which additional sensors would allow the system to detect a fault or discriminate between two (a set of) faults.

These tasks can be performed by running the diagnostic core several times and singling out if there are cases where the faults cannot be detected or discriminated.

- The tool for supporting the FMEA generation allows the user to perform qualitative simulations for determining the consequences of faults. These consequences are in fact part of the FMEA tables to be filled in and, presently, are determined by experts, which rely on experience. Thus the tool can provide a significant support, making FMEA generation more reliable and guaranteeing the quality of the process.
- The currently available on-board hardware resources do not guarantee that an MBD diagnostic system can be implemented on a car ECU. VMBD proposed an approach in which the on-board diagnostic code is generated automatically starting from the on-board system and using a machine learning approach. In particular, the approach is based on the generation of

decision (fault)-tree after running several cases with the MBD diagnostic engine and then inducing the tree from these cases [2]. The tool for supporting the generation of on-board diagnostic strategies is based on the same idea and thus relies on the MBD core and on a decision tree learning algorithm.

- The tool for comparing alternative design options and for performing a sort of “what if” analysis during the design process is defined on top of the previous ones. This means, in particular, that one can compare the results of diagnosability analysis or compare the consequences of a fault on two different design options (e.g., after changing some components or replacing some sensors). This tool, therefore, is a sort of interface for using the previous ones on two design schemas.

The modules have very simple graphical interfaces that can be activated from the Matlab/Simulink environment and which make almost completely transparent for the designer the presence of a MBD diagnostic system. We believe that this is important for improving the acceptability of the tools and for inducing designers to use them during the process to evaluate different design decisions.

## 6 GUIDING APPLICATIONS AND PRELIMINARY EVALUATIONS

IDD will use a number of guiding applications with the goal to demonstrate how the diagnostic tasks described can be performed by using the new process and the new tools architecture. Furthermore, we aim to demonstrate how additional advantages of the new method can be achieved, e.g. optimization of sensor placement or deeper diagnostic performance. Thereby, the guiding applications serve, on the one hand, as case studies for the application of the new techniques and, on the other hand, as test cases and demonstrators of the results of the project.

The guiding applications chosen cover on the one hand different mechatronic systems with central ECU-functions, and on the other hand the general application of diagnostic tasks to multiplexed architecture systems.

The choices have been made with respect to significant complexity, relevance to diagnostics, representativeness of typical design process and manifestation of advantages of the new design process and diagnosability analysis.

More specifically, the applications we chose are the following:

- Air delivery system and the Common Rail Injection System.
- Cooling system.
- Air climate system.
- Multiplexed architecture.

For each guiding application, we started from a quantitative model (Matlab model for the first three applications, functional model for the last)., we then showed how qualitative models can be used to perform diagnosis, diagnosability analysis and comparison of different design alternatives. We made these experiments with

hand-made models of some of the systems above while we are currently working at the automatic derivation of the qualitative models from the quantitative ones.

The initial results of the analysis showed that the new process has interesting potentials for improving the design, especially as regards all issues related to diagnosis. In particular, it can overcome some of the problems discussed in the introduction and in Section 2, allowing for a quick exchange of information between the various tasks and allowing the designers to evaluate the effect on diagnosability of different design choices. In this way one can have systems that are diagnosable and not only controllable and the overall time for design could be significantly improved (avoiding loops in the process described in Section 2). A more thorough evaluation will be provided after the complete development of the guiding applications, at the end of the project in the first months of 2003.

## 7 CONCLUSIONS

Bla Bla Bla

## REFERENCES

- [1] P. Bidian, M. Tatar, F. Cascio, D. Theseider-Dupré, M. Sachenbacher, R. Weber, C. Carlén: Powertrain Diagnostics: A Model-Based Approach, *Proceedings of ERA Technology Vehicle Electronic, Systems Conference '99*, Coventry, UK, 1999
- [2] F. Cascio, L. Console, M. Guagliumi, M. Osella, A. Panati, S. Sottano, D. Theseider-Dupré: Strategies for on-board diagnostics of dynamic automotive systems using qualitative models, *AI Communications*, June 1999.
- [3] L. Console, O. Dressler: Model-based diagnosis in the real world: lessons learned and challenges remaining, in *Proc. IJCAI 99*, Stockholm 1999, pp.1393-1400.
- [4] R. Davis: Diagnostic Reasoning based on Structure and behavior, *Artificial Intelligence* 24(1), 1984: pp. 347-410.
- [5] W. Hamscher, L. Console, J. deKleer: *Reading in Model-Based Diagnosis*, Morgan Kaufmann 1992.
- [6] M. Sachenbacher, P. Struss, R. Weber: Advances in Design and Implementation of OBD Functions for Diesel Injection Systems based on a Qualitative Approach to Diagnosis, *SAE 2000 World Congress*, Detroit, USA, 2000.
- [7] M. Sachenbacher, P. Struss: AQUA: A Framework for Automated Qualitative Abstraction. In: *Working Papers of the 15th International Workshop on Qualitative Reasoning (QR-01)*, San Antonio, USA, 2001
- [8] P. Struss: What's in SD, Towards a Theory of Modelling for Diagnosis, in [5]
- [9] Raz'r Version 1.6, Occ'm Software GmbH, see <http://www.occm.de/>