IDD: Integrating Diagnosis in the Design of automotive systems

Claudia Picardi[1], **Rosanna Bray**[2], **Fulvio Cascio**[3], **Luca Console**[4], **Philippe Dague**[5], **Oskar Dressler**[6], **David Millet**[7], **Berndt Rehfus**[8], **Peter Struss**[9], **Christian Vallée**[10]

Abstract. In this paper we overview the achievements of the IDD European project, which aims at defining a new framework for the design of automotive systems. In particular, starting from the weaknesses 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 software), the project aims at defining a new process in which these issues are integrated within the design of a system and of its control strategies. The project also aims at defining and implementing a software toolkit supporting the new process. The toolkit integrates applications for design and simulation (e.g., Matlab Simulink) and model-based reasoning systems for diagnosis-related tasks.

1 INTRODUCTION

The importance of diagnosis in onboard automotive systems is constantly growing together with the complexity of the systems themselves. 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 it. 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 Mode Effects Analysis, which is very useful to discover safety critical faults or failures) are only partially considered in this phase. Usually, these activities are performed by separate teams and most of the times after the design decisions have been made. In many cases the diagnostic team has 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, there are cases in which it is not possible to single out 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). This problem has been previously experienced in the VMBD project, when applying Model-based Diagnosis to the “Common rail injection system” (see [1]). In that application few sensors were available on-board, thus 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 whom the driver is a customer to be satisfied.

The European V Framework project “Integrated Design process for onboard Diagnosis” (IDD) 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 the designers a set of tools that can help them in evaluating 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 software 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: in Sect. 2 we sketch the current design process, discussing the marginal role played by diagnosis related activities; Sect. 3 pro-

[1] Dipartimento di Informatica, Università di Torino (Italy); email: picardi@di.unito.it

[2] Centro Ricerche Fiat (Italy); email: rosariaanna.bray@crf.it

[3] Centro Ricerche Fiat (Italy); email: fulvio.cascio@crf.it

[4] Dipartimento di Informatica, Università di Torino (Italy); email: luca.console@di.unito.it

[5] LIPN, Université Paris Nord XIII (France); email: dague@lipn.univ-paris13.fr

[6] OCC’M Software GmbH (Germany); email: dressler@occm.de

[7] PSA - Peugeot Citroen (France); email: david.millet@mpsa.com

[8] Daimler Chrysler (Germany); email: bernd.rehfus@daimlerchrysler.com

[9] Technische Univ. München and OCC’M Software GmbH (Germany); email: struss@in.tum.de

[10] Adersa (France); email: Christian.Vallee@adersa.com

poses a new design process which integrates these activities; Sect. 4 discusses modeling issues while Sect. 5 introduces the prototype architecture we are developing for supporting the new process; Sect. 6 overviews the guiding applications on which we are working; Sect. 7 concludes the paper mentioning some open problems for future research.

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 taking into account 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 testing.
- 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 models for simulating the system being designed; how and when the results of the simulation produce modifications to design choices, ...
- 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 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 and models are used in the different phases of the process

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 problems and weaknesses of the current process and for defining requirements for the new process.

The results of the analysis can be synthesized as follows.

- The process is divided in four phases: "strategic", "technology", "integration" and "production"; our main focus is on the second, whose structure is depicted in figure 1. There are three main steps: Step A defines the specification of the system being designed; 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 cases the generation of FMEA and On-Board Diagnostics 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 (and simulating) control models and engineering models of the system being designed. The tool 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. These problems can be summarized by the following three items:

- FMEA and diagnostic development are sequential activities and they are mainly performed using experience and without model-based supporting tools.
- Usually the 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.
- Since FMEA and diagnostics development is carried out after component design/layout (step B of the technology phase); 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 deemed to be a sufficient reason 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 tightly connected with a new architecture for the tools to be used.

The proposal is centered around the idea that the designers (that is, the different experts involved in the design process) should perform a series of activities in an interleaved way (instead of performing them in a sequence). These activities are:

- design of the physical system;
- design and simulation of control algorithms;
- generation of the FMEA;
- development of on-board diagnostic (OBD) software.

In other words, the activities in step B, those in step C and those performed in parallel with step C (FMEA, OBD) should be carried on in parallel, and with a strong feedback from one to the other.

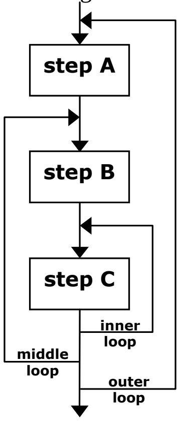

Figure 1. The technology phase

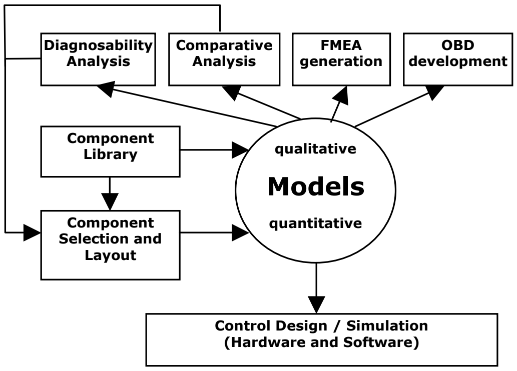

Figure 2. Different activities within the new process

Moreover, new activities should be introduced, that help in optimizing system design with respect to diagnostic software; these activities are:

- diagnosability analysis, i.e. investigation of which faults are detectable and discriminable from each other;
- 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.

The redefinition of the design process can be made possible by providing designers with a set of tools that (i) automate some of the tasks, in particular the new ones, thus softening the transition to the new design process; (ii) simplify the integration of different tasks, by offering a uniform interface and re-using as much information as possible among different tools; (iii) support designers in exploiting the output of the different tasks for the improvement of the design.

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 models of the system being designed, which are 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 principles 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 we want to integrate: (i) design of the system, design of its control software 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 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 might be interested in lower 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 models of high level components which can be decomposed by singling out subcomponents 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 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 mathematical 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 model has often a dynamic nature).

What makes design and diagnosis models 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 a model in which variables range over a discrete set of qualitative values (for example, we may consider only thr three values "positive", "negative" or "zero"). Consequently, models can be regarded as a set of relations (or constraints) among 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 Model-based diagnosis [5]. 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 cases qualitative relations are sufficient to perform fault isolation or identification. Moreover, these models are easier to simulate (and often more efficient). Besides, 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 are not sufficient for design and especially for control design and simulation, which require very precise models.

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, while various methodologies and languages have been proposed in the last decades in the AI community for developing the latter type. In particular, these different methodologies rely on different modeling assumptions [3, 7] i.e., on different abstractions, and thus each one of them is suitable for some applications and problems. Recent research on automotive applications showed that simple models, based on signs and possibly involving qualitative deviations can be sufficient for achieving interesting diagnostic capabilities [6, 1, 2]).

These two types of models can be clearly developed manually, thus building two separate libraries: one with quantitative models and one with qualitative models. Although this approach can be reasonable for demonstrating the advantages of the new design process, we believe that in the long run 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 who have no actual interest in these models or 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 kinds of quantitative models used by designers and control people have very different formats; in many cases they are not even 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 the discussion in [2]).

A final remark to conclude this section. Another standardization goal of IDD is 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 information extracted from quantitative models.

5 TOOLS 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 application program, but rather as plug-ins 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 on-board 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 [4].

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 (sets of) faults can be discriminated;
  - determining if a fault (or set of) can be detected;
  - determining which additional sensors would allow the system to detect a fault or discriminate between two (sets 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, reducing the time needed for producing the FMEA of a system and making FMEA generation more reliable and guaranteeing the quality of the process.
- Currently available on-board hardware resources do not guarantee that an MBD diagnostic system can be implemented on a car ECU. The European project VMBD proposed an approach in which the on-board diagnostic code is generated automatically starting from the system model 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 [1]. 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) or even in two completely different design schemas (i.e., schemas containing different sets of components or with different component layouts). 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 make almost completely transparent to the designer the presence of a MBD diagnostic system. We believe that this is important for improving the acceptability of the tools and for bringing designers to use them during the process to evaluate different design decisions.

6 GUIDING APPLICATIONS AND PRELIMINARY EVALUATIONS

IDD is using 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 at demonstrating 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 the typical design process and evidence of advantages of the new design process and diagnosability analysis.

More specifically, the applications we chose are the following:

- Air and fuel delivery system and the Common Rail Injection System (by Centro Ricerche Fiat and Magneti Marelli).
- Cooling system (by Daimler Chrysler).
- Air climate system (by PSA and Adersa).
- Multiplexed architecture (by Renault).

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 effects of different design choices on diagnosability. 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

In the paper we presented an overview of the goals and achievements of the IDD European project. The aim of the project is to introduce a new process for the design of complex automotive systems. In the new process different activities can be performed in an interleaved way: selection of components and of their layout, definition and simulation of control, diagnosability analysis, FMEA generation, comparative analysis of different design choices. This can lead to a significant transformation of the design cycle, leading to a reduction in the design time (fewer cycles, due to the possibility of immediate exchange of information and to the interleaving between the various activities) and to the design of systems which are not only easily controllable but also easily diagnosable. Furthermore, other tasks can be supported, such as the generation of the FMEA and the generation of on-board diagnostic software. In order to support the new process we also defined a toolkit which integrates software platforms for design and simulation (namely Matlab/Simulink) and model- based reasoning systems for performing the activities related to diagnosis.

The project showed that MBR can play a fundamental role in the new design process and that, at the same time, this direction is an important one for the growth of MBR. However, it also pointed out new open problems to be faced by research in the next years. The most important (and interesting) one is the derivation (or construction with intelligent support) of qualitative models from quantitative ones. Finding a general solution for this problem is challenging but it is also very critical because, as pointed out in the paper, the problem itself is the bottleneck for a wide diffusions of MBR technologies.

ACKNOWLEDGEMENTS

The project was supported by the European Commission, V Framework projects under grant GRD1-1999-11263, partners: Centro Ricerche Fiat (CRF), DaimlerChrysler, Magneti Marelli, OCC’M Software, PSA Peugeot Citroen, Renault, Technische Universitat München, Universitè de Paris Nord (XIII), Università di Torino.

REFERENCES

[1] F. Cascio, L. Console, M. Guagliumi, M. Osella, A. Panati, S. Sottano, and D. Theseider Dupré, ‘Generating on-board diagnostics of dynamic automotive systems based on qualitative deviations’, *AI Communications*, **12**(1), 33–44, (1999).

[2] L. Console and O. Dressler, ‘Model-based diagnosis in the real world: lessons learned and challenges remaining’, in *Proc. 16th IJCAI*, pp. 1393–1400, Stockholm, (1999).

[3] R. Davis, ‘Diagnostic reasoning based on structure and behavior’, *Artificial Intelligence*, **24**(1-3), 347–410, (1984).

[4] Occ’m Software GmbH, ‘Raz’r version 1.6, see http://www.occm.de’, Technical report, (2001).

[5] *Readings in Model-Based Diagnosis*, eds., W. Hamscher, L. Console, and J. de Kleer, Morgan Kaufmann, 1992.

[6] 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).

[7] P. Struss, ‘What’s in SD? Towards a theory of modeling for diagnosis’, In Hamscher et al. [5], 419–449.