Model-based Tools for the Integration of Design and Diagnosis into a Common Process - A Project Report*

P. Struss[1,8], B. Rehfus[2], R. Brignolo[3], F. Cascio[4], L. Console[5], P. Dague[6], P. Dubois[7], O. Dressler[8], and D. Millet[9]

[1]Technical Univ. of Munich, [2]DaimlerChrysler AG, Stuttgart, [3]Magneti-Marelli Spa, [4]Centro Ricerche Fiat; Torino, [5]Universita di Torino, [6]Université de Paris Nord, [7]Renault, Paris, [8]OCC'M Software, Munich, [9]PSA Peugeot Citroën, Paris
struss@in.tum.de, struss@occm.de

Abstract

The growing importance of on-board diagnosis for automobiles demands for a close integration of diagnostic tasks in the entire design process. This report describes work carried out to date within the European project „Integrated Design Process for onboard Diagnosis„ (IDD). It presents an analysis of the current design process and the model of a new process which allows for a better integration of diagnosis related tasks, such as diagnosability analysis, failure-modes-and-effects analysis (FMEA), on-board diagnosis design, in the overall design process of mechatronic subsystems. We then discuss in what way model-based technology can provide tools to support the actual integration and, in particular, present an approach to model-based diagnosability analysis..

Introduction

The importance of diagnosis in onboard automotive systems is constantly growing together with the complexity of the systems. The average dimension of the diagnostic code inside a modern electronic control unit (ECU) is now more than 50% of the whole code. At present, there is no correspondence between such an important role of diagnosis in onboard systems and a similar role that diagnosis could play in the design process chain.

The correct way of dealing with this situation is **to re-organize the design and development chain so that the diagnosis is no longer the last task in the design chain.** This goal provides an opportunity and challenge to model-based systems technology for several reasons. First, in early design stages, when physical prototypes of the designed system are not existing, diagnostic reasoning can only be based on a model. Second, since the design is subject to revisions, the adaptation of diagnostics and fault analysis to such revisions has to happen automatically or, at least, without major efforts. Finally, the existence and use of (simulation) models for the development and validation of control design can provide a basis for the application model-based diagnosis technology.

The European Fifth Framework project „Integrated Design Process for onboard Diagnosis" (IDD) pursues the goal to formalize and standardize 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 model-based tools that can help them in evaluating and understanding the effects of each choice on the system being designed.* The IDD project was started February 2000 with a duration of three years and involves both industrial and academic partners: Fiat CRF (Torino), Magneti-Marelli SpA (Torino), PSA, Peugeot Citroen (Paris), Renault (Paris), DaimlerChrysler AG (Stuttgart), OCC'M Software GmbH (München), Universita di Torino, Université de Paris Nord, XIII, and Technische Universität München.

Except for the approach to diagnosability analysis, this paper does not aim at presenting new model-based theories or techniques, but rather focuses on describing the work and intermediate results of this project in order to increase the awareness of this challenge in the field of model-based reasoning. Therefore, we start with a description of the current design process and its deficiencies. Based on this, a new design process is proposed in section 3 that introduces the exchange of models as the major medium for a closer interaction between control design on the one hand and failure-modes-and-effects analysis (FMEA) and diagnostic design on the other hand. Section 4 outlines the technological and software basis chosen by IDD to develop the tools that are required to realize this integrated process. We then present our approach to model-based diagnosability analysis. Finally, we outline the remaining work in the project and list the guiding applications which will be used in the project for validation of the tools.

* This work is supported by the Commission of the European Union (Project no. G3RD - CT199-00058)

Analysis of the Current Process of Design and Generation of Diagnostics

The current processes of each industrial partners have been investigated with a focus on the integration of the diagnostic process and diagnosis-related processes into the whole design process of mechatronic subsystems.
Starting from these results a „merged process“ has been developed that is based on the similarities recognized, ignoring details and small differences. The abstraction of this process will be used as a comprehensive reference for the current design processes. This analysis and its consequences are presented in more detail in [Brignolo et. al. 01].
In the framework presented here we consider especially processes related to **mechatronic** *subsystems*, such as air conditioning or engine control systems. These subsystems involve ECUs as centers of control and diagnostic functions and the physical system, comprising mechanic, hydraulic, electric components. Following [Bortolazzi-Steinhauer 00], Fig. 1 summarizes the overall design, isolating the different phases and showing in which way the process for a subsystem, which is the most interesting one in this project, is related to the entire process.

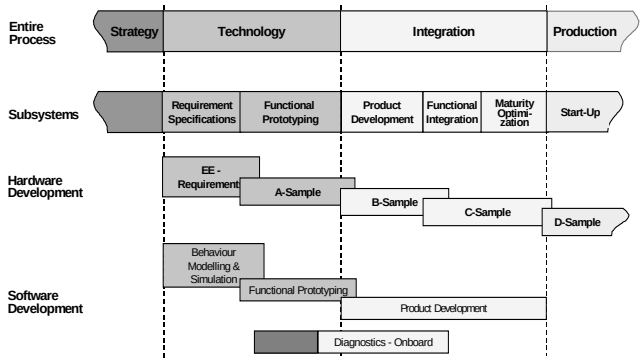

Figure 1 Entire Process and subsystem process, overview

During the ‚strategy phase‘ a first conceptual framework for the new product is worked out, the ‚technology phase‘ targets the concept approval, the ‚integration phase‘ focuses on the realization of the new product by taking into consideration technical feasibility and manufacturing aspects, and, finally, the ‚production phase‘ ensures the industrial mass production with the correct requirements of quality.
The IDD approach focuses primarily on the Technology phase which leads to the first almost complete prototype, but takes into account that a good amount of diagnostic development is performed at present in the Integration phase, as illustrated in Figure 1.

From an abstract point of view, the reference process, which is focussed on the functional prototyping within the technology phase, can be modeled as a set of nested loops:

- **Specifications loop**: Definition of requirements, specifications and implementation of the validated result. In this phase also feedback from after-sales and customers may be involved. Further requirements may be added depending on mock-up observations.
- **Outer design loop**: Design of the whole system prototype, involving the definition of the overall structure of the system, i.e. the selection of the physical (mechanic, hydraulic, electric) components and decisions about the overall layout of the system. This loop terminates when the prototype meets all the requirements and specifications. The core activities are design of the system including its control and diagnosis, comprising a series of inner design loops, and the hardware development of the physical system, which runs in parallel.
- **Inner design loop**: Design of the ECU-based control system and components. Each iteration involves the design of the control algorithms, FMEA, diagnostic development, implementation of the ECU (HW and SW) and verification of the algorithms, as shown in Figure 2. The verification step at the end of the first iterations is performed using models (software/ hardware in the loop), whereas, later, the physical system is used. Depending on the achieved results, there are several iterations, each one of them producing an advanced prototype.

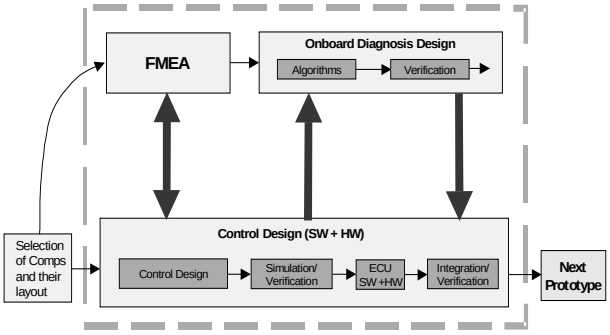

Figure 2 The reference process, one iteration of the inner design loop

Three problem areas in the reference design process have been identified as the essential ones with respect to a better integration of the diagnostic tasks, mainly in the inner and the outer design loops.
The first problem concerns the interaction between the diagnosis design process and the FMEA generation (cf. upper part of Figure 2).

- FMEA and generation of onboard diagnosis are separated and sequential tasks.
- Only few tools support the information extraction process needed for the FMEA, e.g. simulating the consequences of faults or studying interactions between faults. Thus, a lot of work is left to the experience and sensibility of the people that perform FMEA.

The second problem area concerns the interaction between FMEA and the development of diagnostics, and the development and design of control algorithms of the system (cf. Figure 2).

Currently, these are two substantially separate tasks, despite the fact that there are important interdependencies. Examples for possible interactions are:

- a change of the control algorithm may turn a physical component, that was not very essential before, into a critical one and, hence require additional diagnostics,
- a change of the control algorithm promotes the masking of certain faults that were detectable more easily before. Again, additional diagnostics have to take this into account,
- a change of the diagnostics aiming at enhancing diagnosability may exploit additional signals, which may possibly improve control, as well.

As a consequence, requirements and constraints arising from one of these tasks can be dealt with by the other ones only in the next inner design loop, i.e. changes in the design of control algorithms can have impact on FMEA/ diagnosis only during the next inner design loop and vice versa, thus causing additional iterations and time delay.

The third problem area concerns the relation between the design of diagnosis and component selection and layout definition (cf. left-hand part of Figure 2).

The problem here is, that currently the component selection task is external to the inner design loop. As a consequence, for instance the choice or placement of sensor is often not optimized with respect to diagnosis purposes, or, if later changes are made, additional (outer and inner) design loops are needed that cause delays.

An improvement could be reached by performing a comparative analysis (‚what-if-analysis') inside the inner design step and the integration in the early phases of control and diagnostic development. Thus, part of the component selection task is moved inside the inner design process, and, in particular in the early phases of the inner design loop, it is possible and cheap to modify component choices, e.g. sensors, regarding type, sensitivity or placement and to immediately explore the impact on control generation, FMEA, diagnosability analysis, and diagnosis generation.

The New Process

Based on this analysis of the reference process and the outlined improvements, we propose a frame for a new process which is closely connected to 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 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 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 different designs and in making choices about the best design of a system.

- Such a tight integration of different activities and the aim to perform them concurrently require the fast and reliable exchange of information about any changes in the design introduced by any of the activities. This is why we propose that the **model of the system being designed must play a central role in the new process**, as indicated by Figure 3.
- The aims to update FMEA, diagnosability analysis and OBD generation quickly after a change and to consider different design alternatives in parallel establishes the requirement that these tasks can be effectively supported or automated by computer tools based on the model, i.e. they have to be **model-based tools**.

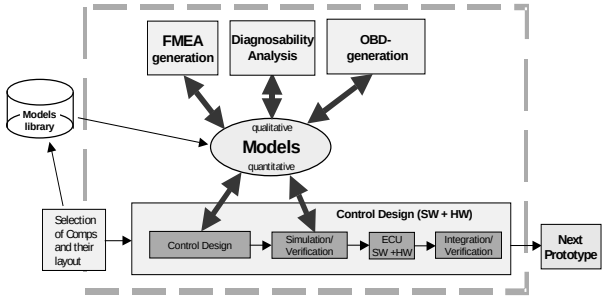

Figure 3 Frame for the new design process

Software Support for the New Process

Accordingly, the actual goal is to provide a new set of functions for supporting the designer, which are realized as ‚software plug-ins' added to the existing software tools for design. Within the scope of IDD, we are considering three plug-ins:

- tools for diagnosability analysis
- tools for supporting the FMEA generation (cf. [Price 98])
- tools for supporting the generation of onboard diagnostics (see e.g. [Bidian et al. 99], [Cascio et al. 99], [Sachenbacher-Struss-Weber 00]).

These tools rely on model-based systems and will be based on a common set of models and a common model-based diagnostic system core.

The new process and the respective tools should be integrated or combined with the simulation tools, that are currently used for the design of control strategies and typically based on quantitative models. In IDD, this is Matlab/Simulink. This requires software that transforms the models created in these environments into qualitative diagnostic models that form the basis for the model-based tools.

Figure 4 summarizes the overall architecture of the new design support system .

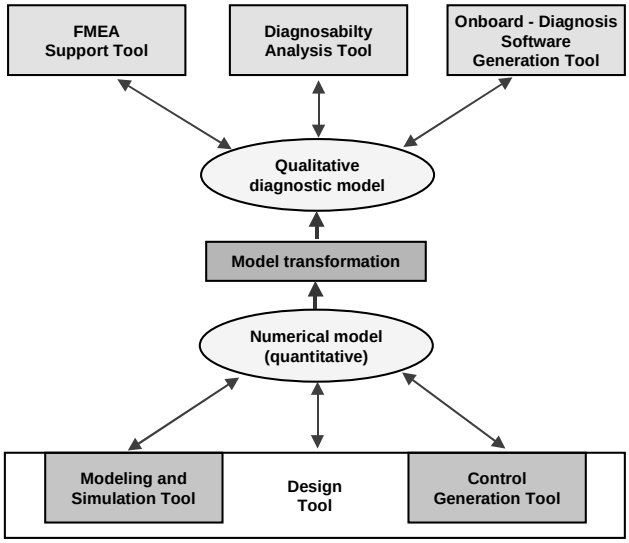

Figure 4 Tools architecture for the new process

A challenge lies in providing

- a common software platform with components that are re-usable in different contexts, and
- the harmonization of models used for different tasks.

The latter is ideally to be achieved by automated transformation routines. In particular the automated transfer of traditional quantitative models (used e.g. for simulation and control design) to qualitative models allowing for automated FMEA and fast, i.e. real-time, on-board diagnosis, is a central target. If indeed successful, the re-use of existing model fragments for **different** tasks will reduce life cycle costs by a significant amount.

IDD envisions three types of application settings:

- an **integrated toolbox** with its own graphical user interface and storage of models. A component-oriented ontology has been chosen to best address modeling requirements in the automotive domain.
- a variety of **plug-ins** to industry-adopted existing tools. In IDD, we have chosen MatLab/Simulink. Models will possibly be stored with these tools and a specific graphical user interface will be limited, if existent at all. The plug-ins provide additional functionality, namely diagnosability analysis, FMEA, and the transformation of design information captured by the Matlab/Simulink model.
- the (on-board) processing scenario for **dedicated applications** such as diagnosis and monitoring. They are dedicated to a particular variant of a device. A diagnosis and monitoring application on a ECU is a typical example.

The IDD toolbox and plug-ins will be running on Microsoft Windows. Therefore, COM (component object model) was chosen as a protocol for the interaction of (binary) components. All the engines, transformers, etc are implemented obeying this standard. This allows for the re-use of functionality in different contexts, and, in particular, the three different application settings. The second cornerstone is given by the use of XML (extended markup language) for describing data in a uniform and exchangeable way. Many of our software components take XML documents as input and produce such documents as output.

COM and XML allow us to build task-related applications that are constructed from components which themselves are aggregated from even more basic components. The components in the layer directly under the application level we call **engines**, our third cornerstone. So, there are (re-usable COM) components that encapsulate a diagnosis engine, an FMEA engine, a predictive engine, a transformation engine, etc. An important consequence of the choice of COM, XML, and engines is that the resulting architecture is an open one, open at any desired degree down to the level of individual methods of low level objects.

At the component level, the IDD consortium has chosen OCC'M's Raz'r [RAZ'R 02] as a basis for implementation. It provides state of the art model-based systems software packaged into COM-components and supplied with XML-interfaces. This allows for further extensions as needed by the consortium requirements. These components include

- an **ATMS** (Assumption Truth Maintenance System) which provides fast consistency checking and handling of time. While still adhering to the basic

framework of assumption-based truth maintenance [de Kleer 86], the employed technology has changed substantially making possible the implementation of on-board systems meeting real-time requirements ([Sachenbacher-Struss-Weber 00]).

- a c**onstraint-based predictive engine** which allows to limit the computational efforts by specifying appropriate foci of attention.
- a **model compiler** which produces system descriptions (**XML** documents) suitable for processing by various engines. For representing constraints, a data structure similar to ordered binary decision diagrams (**OBDD**), but also suitable for direct constraint processing is used as a compact representation [Bryant 92].
- a **diagnosis engine** which accepts a system description and a continuous stream of observations (measurements) as the input and produces an assessment of the current situation by listing the best candidates for diagnosis.
- The **model transformation engine** is central and touches on still open research questions. Therefore, it is a main subject of the consortium's current activities. As already pointed out, automated model transformation is required to obtain qualitative models. Behavioral and structural descriptions are extracted from numerical models (developed in Matlab/Simulink), converted to qualitative models represented in XML form and possibly transformed into more abstract descriptions through a process called task-dependent model abstraction ([Sachenbacher-Struss 01]). The foundations of one of the implementations and a critical discussion of the practical experiences are presented in [Struss 02].

In the following, we discuss the foundations for the diagnosability analysis engine, that forms a specific contribution of the project, in a little more detail.

Diagnosability Analysis Engine

Diagnosability analysis is expected to answer two different types of questions:

"*For a particular design and a chosen set of sensors, determine:*

- **Fault detectability**, *i.e. whether and under which circumstances the possible faults considered can be detected (by the ECU)*
- **Fault** *(class)* **discriminability**, *i.e. whether and under which circumstances the ECU is able to distinguish different classes of faults.*"

The second question is a generalization of the fault identification task ("*Determine the present fault mode unambiguously*"). This generalization is motivated by on-board diagnosis requirements: full fault identification is usually not possible and also not required for on-board purposes, since there is a limited set of possible recovery actions that can be performed by the control unit and which are to be selected dependent on the general type of fault and its severity rather than the individual fault. For instance, only certain critical faults may require immediate shut-off of the engine while others allow continued operation possibly under certain limitations.

Also off-board diagnosis is appropriately characterized as fault class discrimination where the classes comprise the faults of the various smallest replaceable units. More generally, diagnosis is usually a discrimination task whose goal is defined by the available "therapy" actions.

Discriminability is the fundamental task, because detectability can be formulated as discriminability from the normal behavior.

Although the ultimate goal is to discriminate **classes** of behavior modes from each other, the analysis has to based on the discriminability of each pair of **individual** faults taken from any pair of classes, which is unfortunate from a computational point of view.

In our framework, (fault) behavior modes are represented as finite relations, and discriminability analysis becomes the task of computing the observable distinctions between two relations. So, let V_{obs} be the set of observable variables. In an on-board situation, this corresponds to the set of actuator and sensor signals. Since we want to characterize the situations under which detection or discrimination is possible, we introduce a set of variables V_{cause} that are exogenous or "causal " variables w.r.t. the physical system (i.e. the subsystem excluding the ECU). This set includes the actuator signals but also other quantities that influence the behavior of the physical system. Some of the latter may be observables, e.g. the atmospheric pressure, while other are not (directly) measurable, such as the load. Since on-board diagnosis can rely only on what is observable to the ECU, we define:

$$V_{o\text{-}cause} = V_{obs} \cap V_{cause}$$

and

$$V_{obs\backslash cause} = V_{obs} \setminus V_{cause}$$

as well as the respective projections, $PROJ_{obs}$, $PROJ_{o\text{-}cause}$.

The abstract example in Figure 5 will provide an intuition about possible answers to the discriminability question. The vertical axis represents the observable causal variables and the horizontal axis the remaining observables. There may be many unobservable variables, but the shown projection to the space of observables is all that matters.

Two different fault modes (or, more generally, behavior modes) are represented by two relations. As illustrated by the figure, we can distinguish three different cases:

- In the upper section the relations cover each other, i.e. for any causal stimulus in the projection of this intersection area, the observable set of consistent tuples for the two behavior modes are the same, and, hence, they **cannot be discriminated** from each other.

- In the lower section, they are totally disjoint, i.e. any of the respective causal inputs always leads to different system behavior and, thus, **deterministically discriminates** between the two modes.
- For all other causal inputs, the two modes can **possibly be discriminated**, because the actual response of the system may be outside one of the relations, but is not guaranteed to.

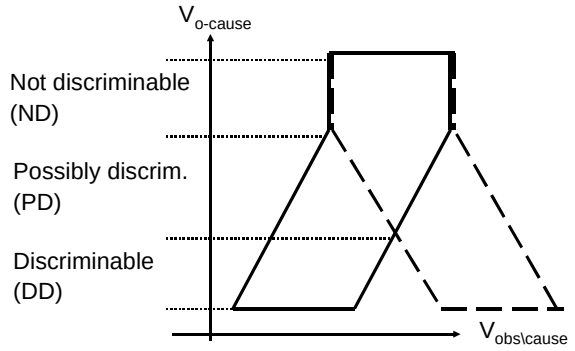

Figure 5 Three categories of discriminability of two behavior modes

With this translation of the task to the analysis of relations, we can also support our previous claim, that, in general, a pairwise comparison of individual modes of required to determine the discriminability of classes of modes. Consider the trivial example of one inverter with two mode classes:

C_1 ={output-stuck-0, output-stuck-1},

C_2 ={shorted, ok}.

Figure 6 a and b display the four faults in the observable space i, o, grouped in the two classes.

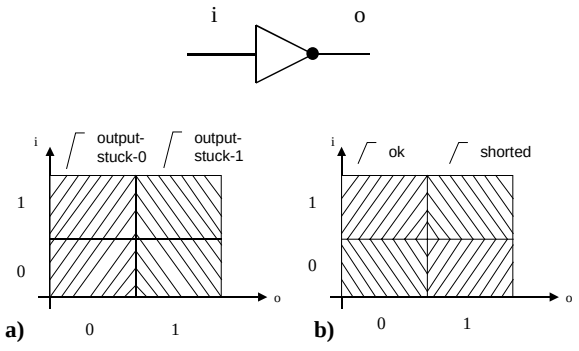

Figure 6 Behavior classes of the inverter for fault classes C_1 (a) and C_2 (b)

Obviously, the faults are pairwise discriminable, and, hence, so are the two classes of faults. However, if we would try to represent each class as the disjunction of its modes and associate with it the union of the respective relations, then both of these class relations cover the entire behavior space and are not distinguishable. The deeper reason is that a fault class represents more than a (exclusive) disjunction of modes. We also make a **persistence assumption**, namely that one particular mode occurs in all inspected situations (i.e. for all inputs).

Before we give formal definitions and computable expressions for the concepts, we introduce one last element: **operating conditions**. This reflects the common practice of distinguishing between ranges of internal or external quantities that result in qualitatively different behaviors and are often reflected by different states of the system and its control. Examples are *engine idle*, *clutch engaged*, *cold engine*, *brake pedal pushed*.

Often, the analysis of fault effects and diagnosability can be restricted to certain operating conditions and is futile for others. For instance, one may not be extremely interested in the detectability of a fault in the air intake system under conditions where the engine is not running (one has to be cautious with such restrictions, though, because firstly, there may be a requirement to perform fault detection beforehand, such as checking the operability of the airbag system or the ABS, and secondly, a broken component could affect operating modes in which it is not intended to be active).

In our approach, an operating condition has to be expressed as a constraint on a subset of model variables. Often, but not always, they will refer to exogenous variables such as the angle of the accelerator pedal or air temperature, and typically, but not exclusively, they are observables (the load, for instance, is not directly observable).

In most cases, the constraint that defines an operating condition will be a conjunction of restrictions on variable values to some interval or state like *temperature>120°C* or *ignition = ON*.

Restricting the analysis to certain operating conditions then boils down to computing the intersection of a behavior relation with their respective constraints.

Definition 1 (Discriminability of behavior modes)

Let $MODEL_{fault1}$, $MODEL_{fault2}$ be the behavior relations of two modes,

OPC_i an operating condition,

and

$SIT \subset DOM(V_{O\text{-}CAUSE})$

a non-empty relation on the observable causal variables.

For OPC_i and SIT, two faults are called

- **not discriminable**, written
  $ND(fault_1, fault_2, OPC_i, SIT)$,

iff

(i) $SIT \subseteq PROJ_{o\text{-}cause}(OPC_i) \setminus PROJ_{o\text{-}cause}$
$(PROJ_{obs}(MODEL_{fault1} \cap OPC_i) \setminus$
$PROJ_{obs}(MODEL_{fault2} \cap OPC_i)$
$\cup PROJ_{obs}(MODEL_{fault2} \cap OPC_i) \setminus$
$PROJ_{obs}(MODEL_{fault1} \cap OPC_i))$

- **deterministically discriminable** , written $DD(fault_1, fault_2, OPC_i, SIT)$,

iff

(ii) $SIT \subseteq PROJ_{o\text{-}cause}(OPC_i) \setminus$
$PROJ_{o\text{-}cause}(PROJ_{obs}(MODEL_{fault1} \cap OPC_i)$
$\cap PROJ_{obs}(MODEL_{fault2} \cap OPC_i))$

- **possibly discriminable** , written $PD(fault_1, fault_2, OPC_i, SIT)$,

iff

$SIT \subseteq PROJ_{o\text{-}cause}(OPC_i) \setminus (SIT_{ND} \cup SIT_{DD})$,

where SIT_{ND} and SIT_{DD} are the maximal relations that satisfy (i) and (ii), respectively.

These definitions characterize the three cases discussed above w.r.t. Figure 6 in a way that can be computed by operations on the extensional constraint representation generated by the model compiler.
Based on the discriminability of modes, discriminability of fault classes can be defined and computed.

Definition 2 (Discriminability of mode classes)

Let $FC_j = \{fault_{i,j}\}$, j =1,2 be two fault classes and OPC_i an operating condition. Let furthermore
$SIT\text{-}SET = \{SIT_{kl}\} \subset P(DOM(V_{o\text{-}cause}))$
be a set of non-empty relations of observable causal variables. FC_1, FC_2 are called

- **not discriminable**, written $ND(FC_1, FC_2, OPC_i)$

iff there exists a pair of modes that is completely non-discriminable:

$\exists fault_{1k} \in FC_1 \quad \exists fault_{2l} \in FC_2$
$ND(fault_{lk}, fault_{2l}, OPC_i, PROJ_{o\text{-}cause}(OPC_i))$

- **deterministically discriminable**, written $ND(FC_1, FC_2, OPC_i, SIT\text{-}SET)$,

iff each pair of modes is deterministically dicriminable for some element of SIT-SET:

$\forall fault_{1k} \in FC_1 \quad \forall fault_{2l} \in FC_2 \; \exists SIT_{kl} \in SIT\text{-}SET$
$DD(fault_{lk}, fault_{2l}, OPC_i, SIT_{kl})$

- **Possibly discriminable**, written $PD(FC_1, FC_2, OPC_i, SIT\text{-}SET)$,

otherwise, iff all SIT_{kl} are in the complement of the non-discriminable situations:

$\forall_{kl} SIT_{kl} \cap SIT_{ND,kl} = \varnothing$

Status and Future Work

As of now, two different alternatives have been implemented to generate the qualitative diagnosis models from existing numerical models which both use Matlab itself to compute the tuples of the modeling relation. In addition, a library of qualitative models will be created manually that allows to configure the model based on the structural description only. Based on a use case analysis, the core of the diagnosability analysis tool and the model-based on-board diagnosis engine have been developed.
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. They include

- The **air delivery system** for diesel engines (Figure 7), comprising the exhaust gas turbocharging system and the exhaust gas recirculation system (EGR. and the Common Rail Injection System (Fiat and Magneti-Marelli).

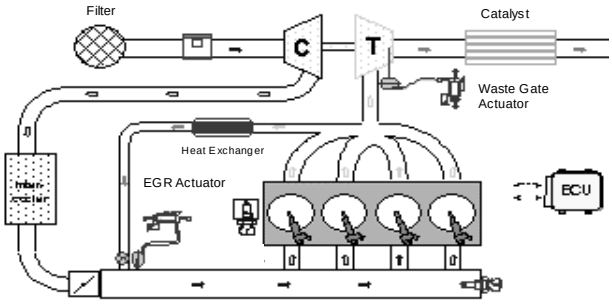

Figure 7 Guiding application: Air delivery system

- The **cooling system** (DaimlerChrysler AG), including an intercooler, which on the one hand increases the efficiency of the engine by cooling the compressed air and, hence, increasing the air charge rate, and on the other hand decreases NOx emissions by keeping the combustion at lower temperature (Figure 8).
- The a**ir conditioning system** (Peugeot Citroën PSA) which consists of two loops that supply a cold heat exchanger and a hot heat exchanger (Figure 9).

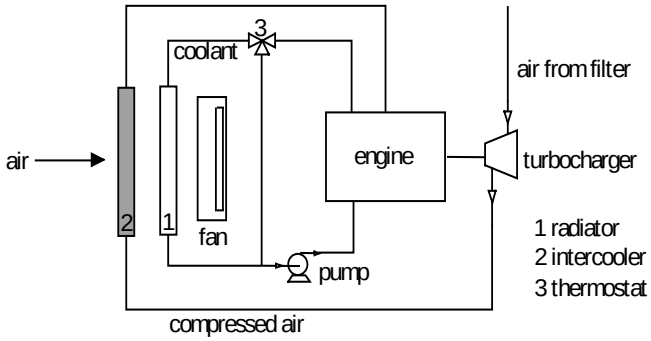

Figure 8 Guiding application: Cooling system

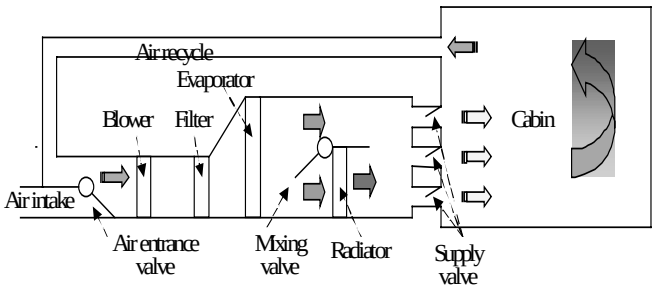

Figure 9 Guiding application: Air conditioning system

- The **multiplexed architecture** (Renault) involving ECUs, sensors, actuators, functions (EF = elementary functions), busses and data frames (Figure 10). The design engineer will be enabled to run a program directly on the representation of a designed architecture and receive the results of an analysis of the interdependency of faults and functions in this architecture.

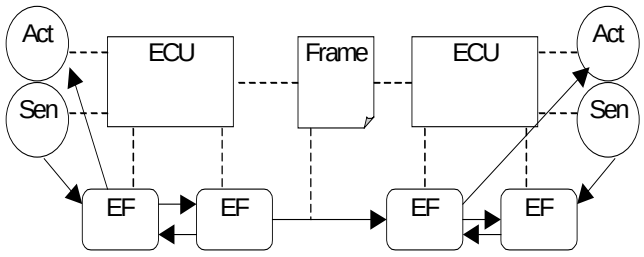

Figure 10 Guiding application: Multiplexed architecture

A first version of models for these guiding applications has been developed and will be used to validate and improve the model abstraction module and to evaluate the tools. By the end of the project in January 2003, we hope to demonstrate the utility of the tools and the benefits of the modified design process based on examples that are close to reality.

References

[Bidian et al. 99] 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

[Bortolazzi-Steinhauer 00] J. Bortolazzi, St. Steinhauer, Th. Weber: Development and Quality Management of In-Vehicle Software. In: Electronic Systems for Vehicles (VDI – Berichte 1547), VDI Verlag , Duesseldorf 2000

[Brignolo et a. 01] R. Brignolo, F. Cascio, L. Console, P. Dague, P. Dubois, O. Dressler, D. Millet, B. Rehfus, P. Struss. Integration of Design and Diagnosis into a Common Process. In: Electronic Systems for Vehicles, pp. 53-73. VDI Verlag, Duesseldorf, 2001.

[Bryant 92] R. Bryant: Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams ACM Computing Surveys, Vol. 24, No. September 1992

[Cascio et al. 99] 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.

[de Kleer 86] J. de Kleer: An assumption-based truth maintenance system, Artificial Intelligence 28, 1986

[Price 98] C. Price: Function-directed Electrical Design Analysis, AI in Engineering 12(4), pp. 445-456, 1998.

[RAZ'R 02] Raz'r Version 1.6, Occ'm Software GmbH, see http://www.occm.de

[Sachenbacher-Struss-Weber 00] 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.

[Sachenbacher-Struss 01] 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

[Struss 02] P. Struss: Automated Abstraction of Numerical Simulations Models - Theory and Practical Experience. In: Sixteenth International Workshop on Qualitative Reasoning, Sitges, Catalonia, Spain, 2002.