

Generating instead of Programming Diagnostics

Oskar Dressler, Peter Struss

Abstract

The complexity of car subsystems and their interaction is growing. This fact, together with the existence of many variants, enforces a trade-off between the quality and coverage of the respective diagnostics and the efforts spent on their development. What is needed is a comprehensive, general (system-independent) and systematic approach to the generation of diagnostics and its automation. This has to be grounded on an appropriate representation of the underlying knowledge about the respective technologies. Libraries containing behavior models of vehicle components capture an essential portion of this knowledge. Models of car subsystems can be automatically composed from such library elements and provide the basis for the automated generation, instead of programming, of diagnostics. The grounding in an explicit model results in an improvement and guarantees of fault coverage and, hence, safety. As of today, the technology is mature enough for introduction into industrial work processes. We explain the foundations of the model-based solution and demonstrate its applicability using a case study on automated generation of on-board diagnostics for the comfort system of a VW Polo with four control units connected via a CAN-Bus. We argue that the representation of expertise in terms of models makes it available – independently of location, time, and people - to computer-based support to various other work processes, such as FMEA, diagnosability analysis, sensor placement, test generation und workshop diagnosis.

Zusammenfassung

Variantenvielfalt und Komplexität von Fahrzeugsystemen und ihrer Interaktion erzwingen derzeit einen Kompromiß zwischen Qualität und Aufwand bei der Erstellung von Diagnostics. Eine durchgängige (systemunabhängige) Systematik und Automatisierung der Diagnose-Erstellung ist nötig, die nur durch geeignete Repräsentation des technologischen Wissens möglich ist. Bibliotheken mit Verhaltensmodellen der Fahrzeug-Komponenten repräsentieren einen wesentlichen Teil des notwendigen technologischen Wissens. Aus einem Systemmodell, das mit Hilfe einer solchen Modellbibliothek erzeugt wird, können die Diagnostics automatisch generiert statt programmiert werden. Die Fehlerabdeckung und damit die Risikobeherrschung werden durch die Automatisierung größer und durch die Modellgrundlage gesicherter. Die Technologie ist mittlerweile reif für die Einführung in den industriellen Prozeß. Wir erklären die Grundlagen und belegen die Anwendungsreife anhand eines Demonstrators für die automatisierte Erzeugung von On-Board-Diagnostics für das Komfortsystems eines VW Polo mit vier über CAN vernetzten Steuergeräten. Wir zeigen ferner, daß dieses Wissen in Gestalt von Modellen nutzbar wird – unabhängig von Ort, Zeit und Personen – für die computergestützte Verarbeitung in verschiedenen Arbeitsprozessen, wie FMEA, Diagnostizierbarkeitsanalyse, Sensorplatzierung, Testgenerierung und Werkstatt-diagnose.

1. Introduction

The task of diagnosing car subsystems becomes increasingly challenging, and computer support to the task is a must, be it on-board a vehicle or in the workshop. This moves the challenge to the creation of diagnostic software and may let the problem appear as a conventional software quality issue. However, we argue that the introduction of advanced software engineering techniques in the process of developing automotive on-board software and, in particular diagnostics, is a necessary prerequisite, but cannot provide a fundamental solution to the problems that are encountered today.

Meeting the challenge and finding solutions requires a radically novel approach based on an understanding of what makes the task difficult. Diagnostic software for vehicles has to detect, predict, localize, or identify malfunctions of the various subsystems and support their repair or recovery. This obvious statement provides the key to identifying the nature of the challenges we are facing:

- The car subsystems are growing more and more complex, because their physical part becomes more sophisticated and because they involve software.
- The subsystems interact both via their physical interaction and the communication among the embedded software, creating complex interdependencies.
- In particular, this can cause multiple effects of faults potentially distant from the origin and faults that emerge from the interaction rather than break-down of individual components.
- Creating diagnostic software that anticipates all this becomes even more demanding due to requirements on the safety of the vehicle and its environmental impact.
- On top of all this, the various types of subsystems and their components come in many variants, coming from different suppliers, built for different OEMs, reflecting varying functionality, type of vehicle, versions, etc., which necessitates the reflection of all these variations and combinations in the software and the software development process.

As a result, we are confronted with a **complex** task that becomes even more costly because it is a **repetitive** one. Since the topic is diagnostic **software**, the application of software engineering techniques promises help. While this cannot be denied, we have to recognize that any software technology that is confined to aspects of the software production process and the inner values of the produced software cannot provide a fundamental solution: If it does not include the consideration of the (physical) car subsystems themselves, it cannot reflect and master the origin of the complexity and the repetitiveness of the task. What is needed is an approach that enables the effective exploitation of knowledge and information about the vehicle technologies themselves, the physical car subsystems, their components, and their structure. We need to base the development of diagnostic software on an explicit computer-represented **model** of the respective car systems.

In this paper, we will not only outline the key ideas and scientific foundations of model-based diagnosis systems (section 3), but also demonstrate the maturity of this technology in the next section by presenting results of a case study on on-board diagnosis of the comfort system of a Volkswagen Polo. The diagnostics for all four doors and their communication via CAN was automatically and completely generated from a model as compact C-code for an electronic control unit (ECU). Finally, we ar-

gue that, beyond the diagnosis task, the model-based technology presented here provides the foundations for a tight horizontal integration of different work processes during the life cycle of a vehicle, starting at early design stages.

2. Automated Code Generation for On-board Diagnostics

2.1 1999: Feasibility Study on a Volvo V70

The feasibility of model-based on-board diagnosis was first demonstrated in the Brite-EuRam project VMBD (Vehicle Model-based Diagnosis¹). Volvo Car Corp., Robert-Bosch GmbH, OCC'M Software GmbH, and the Technische Universität München developed a demonstrator for on-board detection and localization of black-smoke-related faults in a turbo charger Diesel engine (Volvo 850 TDI). The demonstrator vehicle had several built-in faults that could be activated through a switchboard, such as a leakage in the air hose between the turbine outlet and the intake manifold, a mechanical failure in the exhaust gas re-circulation valve, and sensor faults. Sensor and actuator signals from the engine ECU were transmitted via a serial line to a notebook running the model-based diagnosis runtime system (Figure 1). The signals exploited were atmospheric pressure, boost pressure, mass airflow, engine speed, duty cycle of the turbo control valve, and injected fuel quantity, i.e. the ordinary ECU signals with no additional sensors for diagnostic purposes.



Figure 1 Notebook running the model-based diagnosis software on the Volvo demonstrator vehicle

The diagnosis runtime system was based on models of the (correct) physical behavior of the turbo charger system components and achieved fault localization with a

¹ VMBD (Vehicle Model-based Diagnosis) involved Fiat CRF, DaimlerChrysler, Volvo Car Corporation, Robert Bosch GmbH, Magneti-Marelli SpA, GenRad, OCC'M Software GmbH, and several universities and was funded by the Commission of the European Union in the BriteEuRam III program (Project No., #BE 95/2128), (see [Bidian, et al. 99])

performance meeting real-time requirements ([Sachenbacher-Struss-Weber 00]). This prototype demonstrated the principled feasibility of the approach, but, having been developed in a research project and running on a Windows/Pentium PC, prompted two important questions:

- How can the production of such solutions be introduced in the industrial process of on-board software development?
- Is it possible to run model-based diagnosis under the memory and speed restrictions of common ECUs?

2.2 2004: Automatically Generated Code on an ECU

These challenges have been addressed during the last years and the results were evaluated in a case study using a physical demonstrator of a comfort system of a Volkswagen Polo that had been created in the STEP-X project ([Harms et al. 03]). This demonstrator comprises a rack with four window lifters and two mirrors with mirror positioning and the respective switches to operate these systems (Figure 2). The functions are controlled by four ECUs (one for each door) that are connected via CAN bus. To evaluate diagnostics, roughly 100 faults could be switched on, including short and open circuits affecting the electrical motors and Hall sensors of the windows, detuning of switch resistances, and faults of the CAN bus and the connections to the various ECUs. For the benchmark, a fifth ECU, based on an Infineon C167 microprocessor, running the diagnosis software was connected to the CAN bus. CAN messages containing in total about 50 signals were transferred to the diagnosis ECU every 50 ms.



Figure 2 STEP-X demonstrator of a comfort systems (4 doors)

The software that performs diagnosis based on these messages is C code that is **automatically produced by a code generator** based on models of the four subsystems, their communication, and their power supply (Figure 3). This means, the software does not only detect and identify faults at the subsystem level (window lifters and the respective control panels), but also at the level of interacting and interdependent subsystems. It should be noted that the decision to implement the diagnosis on a separate ECU was not due to principled, but only pragmatic reasons, namely the locally distributed development of the control and the diagnostic software. Usually, the subsystem-related diagnostics could be run on the respective ECU and the centralized part on a separate one or one or more of the existing ECUs.

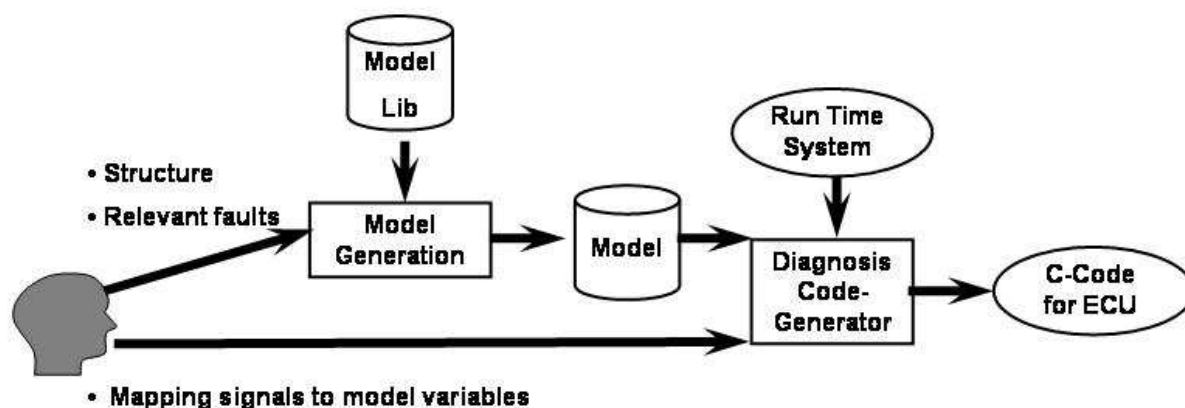


Figure 3 Automated generation of on-board diagnostics

The code generation approach has a major impact on the development process: any change or variation in physical systems to be diagnosed, such as exchanging a component or modifying the structure, requires only the respective modification of the model and re-running the automatic generation procedure in order to reflect the physical change in the diagnostics. No interaction on the software level is needed, but of course the code can be inspected and interfaced with e.g. control software.

The generated C-code when compiled is quite compact, requiring about **25 KB** of memory in total, i.e. including the signal preprocessing, the compressed system model, and the diagnosis algorithm. Even for the implemented centralized version of diagnosis, the system performed in **real-time** and with the processor running at 19,5 MHz used strictly less than **20%** of the **maximum performance capacity**. An important feature of the diagnosis system is that it is not restricted to single faults; even for each single subsystem, it handles **multiple faults** of arbitrary size. In the context of this demonstrator, it delivers **complete** diagnostic information in terms of a fault code for each 50 ms time window, where this code represents the complete **set of all possible faults** in this window (**not** a set of symptoms as current trouble codes). This can be the input for further processing, filtering, debouncing, statistics etc.

An evaluation of the diagnosis results based on the complete set of built-in faults was performed in cooperation by the Technical University of Braunschweig and Carmeq GmbH. A greater than **95 %** correctness of the diagnosis results was attested, where correctness meant that the generated set of possible faults contained

the actual one (provided the fault was detectable at all for principled physical reasons) and did not contain faults inconsistent with the signals. Our analysis of the 5 % incorrect diagnoses revealed that they were due to limitations of the behavior models (rather than indicating defects in the diagnosis algorithm. The demonstrator was shown at the Euroforum Conference “Diagnose von E-/E-Systemen im Automobil” in Stuttgart on July, 7/8 2004.

This benchmark proves that a major step has been achieved:

- Model-based solutions for on-board diagnosis are small and efficient enough to achieve real-time performance on standard ECUs.
- They provide the systematic basis for automated code generation and, hence, a radical improvement in the software development process for on-board systems.

In the next section, we will discuss some of the principles underlying this solution and their practical impact.

3 Principles of Model-based Diagnosis

In order to understand why and how the model-based approach to problem solving in general, and diagnosis in particular, bears the potential of a revolution in software generation, we ask what enables a human expert to perform diagnosis and develop diagnostic code for all kinds of variants of known systems and even for new types of systems. Obviously, it is her or his understanding of how these systems behave and how observed or measured aspects of this behavior relate to possible faults. That she or he is able to cope with novel devices, say an electrical circuit with a new structure, shows that one part of this knowledge is generic. This part is mainly the knowledge about the **behavior** of the individual **components** that establish the device. It is applied in a repetitive way to each single device based on information about its **structure**, i.e. the (device-specific) interaction among the components (Figure).

As long as this foundational knowledge is available only in the heads of the diagnostics developer, the production of the diagnosis software will remain a repetitive one, and this means: costly. Any computer-based solution that addresses this issue has to move this knowledge into an explicit representation in a computer program and have the program apply this knowledge in a step of automated diagnostics generation (Figure), thus liberating the human from this repetitive task. This is what model-based diagnosis does.

Using the device model to generate diagnostics requires a diagnosis algorithm that is **generic**, i.e. independent of the specific device. This is the second contribution of model-based diagnosis. As a result, the **task-specific** software element (diagnosis) is **separated** from the description of the **subject** of the task (the model of the device to be diagnosed) which allows

- the re-use of task-specific algorithms for different devices and, in particular, the automated diagnostics generation and
- the re-use of device-specific models in different tasks (horizontal integration).

It should be obvious that this separation has a strong impact on the utility and cost-effectiveness of the solution in industrial applications and clearly distinguishes it from other approaches that also exploit system models, but interwoven with procedural diagnostic elements. In the following, we will try to provide some insight why the generic diagnosis algorithm has been constructed and then outline some of the modeling principles.

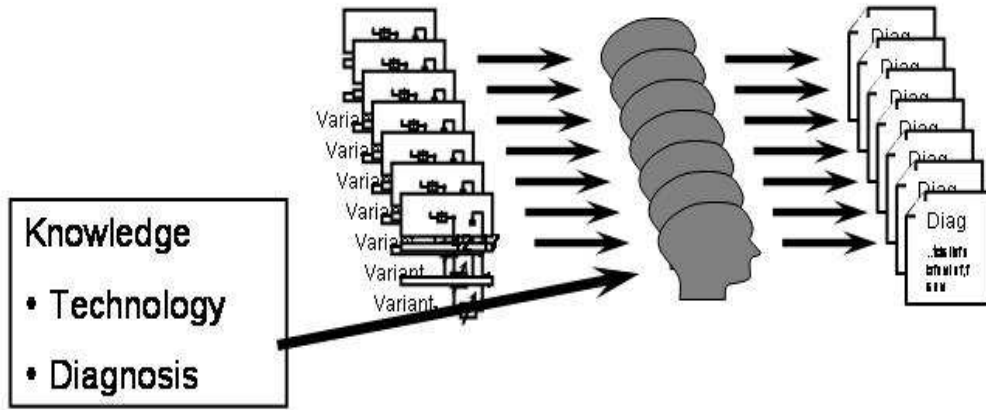


Figure 4 Repetitive process of manual diagnostics generation

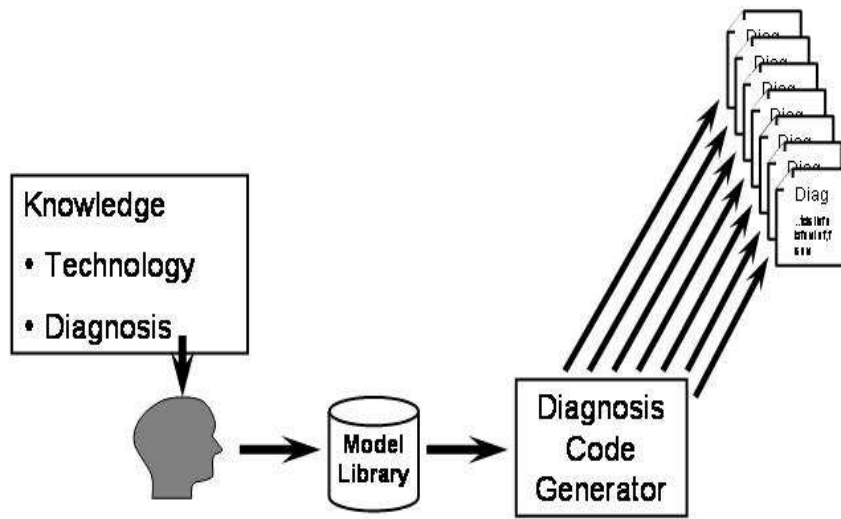


Figure 5 Automated diagnostics generation

3.1 The Foundation of Model-based Diagnosis

We explain the basis of the general diagnosis algorithm using a trivial example comprising a pump which is mechanically driven and pumps air into a container (Figure 6). There are sensors measuring the inflow and the pressure in the container, and also the command that controls the mechanical drive is known. If we assume a scenario where there is a command which requires speeding-up the pump, while the sensors show an increased inflow to the pump and a pressure drop in the container, we are certain that a fault is present, and, moreover, we will reckon that the pump or the container has a leakage or the pressure sensor is failing.

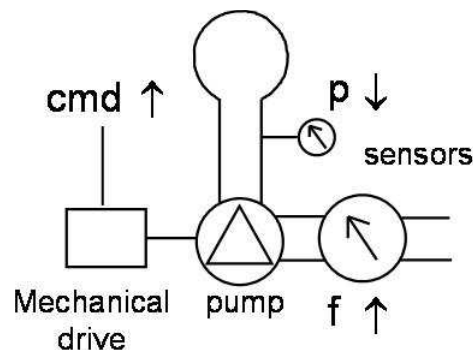


Figure 6 A small diagnosis problem

How could an algorithm that is fed with models of the physical behaviors of the components plus information about their interconnections achieve this result? The core of the algorithm checks the consistency of parts of the device model with the given observations (which is why the approach is called consistency-based diagnosis). For instance, assuming the correct behavior of the pump, the container, and the two sensors is contradicting the sensor readings, because the models of normal behavior of these components imply that both values show the same tendency (sign of the first derivative). A logical implication of this check is

NOT OK(pump) OR NOT OK(container)
OR NOT OK(pressure-sensor) OR NOT OK(flow-sensor)

which indicates the **detection** of a fault. In the same way, the models of the mechanical drive, the pump, the container, and the pressure sensor together are in conflict with the command and the measured pressure which yields

NOT OK(mech-drive) OR NOT OK(pump)
OR NOT OK(container) OR NOT OK(pressure-sensor) .

Together these two “conflicts” imply

NOT OK(pump) OR NOT OK(container) OR NOT OK(pressure-sensor)
OR (NOT OK(flow-sensor) AND NOT OK(mech-drive))

which is a (incomplete) **localization** (isolation) of the fault. We realize that this result not only reproduces the diagnostic hypothesis stated above, but also illustrates that the algorithm can easily generate multiple fault hypotheses (by suspecting the flow sensor and the mechanical drive of the pump). Furthermore, we emphasize that fault localization can be based on models that capture the correct behavior only and does not require models of component faults. They are needed if the fault has to be further **identified**, and they can be checked for consistency with the observations in the same way. For instance, the model of a stuck pump will be refuted, whereas a leaking pump remains a hypothesis. (For a more detailed presentation of the theoretical foundations, see [Dressler-Struss 96])

This admittedly trivial example (which is actually a simplification of the turbo charger application in VMBD) should suffice to illustrate the possibility to generate diagnostic hypotheses based on a behavior model, and, more importantly, to suggest that this

- can be formalized in a rigorous **logical theory** and
- turned into an algorithm which is not reflecting the specific content of the models and, hence, **device- and even domain-independent**.

In the STEP-X benchmark, electrical, mechanical, and communication features were handled uniformly by the same algorithm. And the same principles have been applied to a variety of technical systems ranging from digital circuits and high-voltage power-transportation networks to ballast water tank systems.

The example also indicates that a model (and its execution) suited for this diagnostic task must have certain properties which we discuss in the following.

3.2 Automated Compositional Modeling

The algorithm sketched in section 3.1 is based on the recognition of inconsistent **parts** of the overall system model. A black box model of the entire device would suffice for fault detection, but not yield any contribution to fault localization. For this, we need a **compositional** model, i.e. a model that allows the algorithm to identify the origin of a detected inconsistency in terms of the diagnosis-relevant entities of the device.

This property meets an important requirement that arises from the perspective of software development: given a generic diagnosis algorithm, the main step for generating device-specific diagnostics lies in the provision of a suitable device model. Since we try to address the variant problem and want to avoid unnecessary repetition of tasks, producing the model for each individual device and its variants from scratch is prohibitive. The solution is to collect models of (types of) components in a library and have the device model generated automatically by composing it from the library elements based on the structural description of the respective device. Thus, the production of a device-specific diagnosis system becomes a sequence of two automated generation steps, as shown in Figure 3: First, the device model is generated based on the library and the “blueprint” of the device. Second, this model is compiled together with the general diagnosis algorithm into C-code for the ECU. Once the library exists and contains the necessary model elements, editing the structure and parameterization of the device or, ideally, importing it from a design system is the essential step in the development of diagnosis software. What remains to be done, is to specify how the available signals have to be mapped to the variables occurring in the model.

Also these features,

- the automated composition of the device model from a library, and, hence,
- the re-usability of the library elements for different devices,

are essential for high gains in the development process, and again mark a distinction from other model-based solutions that depend on the development of device-specific models.

The rate of re-use of the library elements is further influenced by the fact that, for diagnostic purposes, often a qualitative description of the component behavior suffices.

3.3 Qualitative Modeling

The example in section 3.1 illustrates that very coarse information, like the sign of the derivatives of the measured quantities, can suffice to derive at least a first set of diagnostic hypotheses. Using numerical values would not improve the diagnostic result in this case. As a consequence, the behavior models need to represent only distinctions that are relevant to discriminate correct from faulty behavior.

To give an example from the STEP-X demonstrator: the information about the position of the windows provided by the Hall sensors can be abstracted to the qualitative positions “top”, “bottom”, and “between” which completely suffices to check their consistency, say, with information about the direction of change. This turns the model into a very generic and re-usable one which requires only a mapping between the numerical position and the three qualitative positions. This mapping may even be adapted dynamically to shifting values during the use of the system (although this was not implemented in the demonstrator).

Besides increased re-usability, qualitative models are essential to the effectiveness and efficiency of the model-based on-board diagnostics: they provide the basis for

- the representation of the model as a finite one which allows for the application of efficient algorithms for manipulation of relations and consistency checking,
- a compact representation of the model, and
- a significant reduction of computational efforts at runtime, because the input signals can be abstracted to qualitative values, and only changes in this qualitative information needs to be fed to the diagnostic algorithm (which was es-

sential to the real-time performance of the VMBD turbo charger demonstrator).

4 Opportunities

The work presented here provides evidence that this technology has grown mature and is ready for making a difference to current practice. The techniques presented here effectively automate the entire production process of on-board diagnostics once a model library is provided. This becomes the focus of the next steps required for the actual industrial exploitation of this technology.

Since one cannot underestimate this non-trivial activity, the question arises whether the expected pay-off for on-board diagnosis justifies such an effort. As we emphasized above, re-use is not only a guiding principle concerning the diagnosis algorithm; it also applies to the models. A model library can be understood as an important repository of corporate knowledge about the technologies applied. In this form, the knowledge becomes explicitly and formally represented and available to both human experts and software programs for different work processes independently of persons, time, and location. Models turn into an important basis for a horizontal integration of these work processes easing the exchange of knowledge and making it coherent.

A model that captures knowledge about the possible behavior and misbehavior of components and systems which supports diagnostic problem solving, is also relevant to other tasks. For instance, Failure-modes-and-effects analysis (FMEA) can be automated by performing model-based prediction of effects entailed by component faults. AutoSteve, a system developed for performing automated electrical FMEA and sneak circuit analysis at Ford and Jaguar (see [Struss-Price 04]) is meanwhile marketed by Mentor Graphics ([MentorGraphics 04]) as part of their cabling and harness toolset. Currently, companies from the aerospace industries, together with software suppliers and academia develop a general tool for FMEA based on models of hydraulic, mechanical, and electrical components in the AUTAS project ([Picardi et al. 04]). Again, the possibility to formulate models at a qualitative level without full numerical detail is essential for performing this analysis at early stages of the design and for generating results for classes of faults and effects.

Therefore, the technology is a fundamental contribution, in particular, to the consideration of diagnostic aspects early in the design process. The IDD project² has joined European OEMs and suppliers in producing a toolbox which allows one to perform model-based diagnosability and detectability analysis and to generate diagnostics (also in the form of decision trees) from the same model ([Dressler-Struss 03]). This project included a case study that highlighted the utility of the model-based technology at the level of interacting subsystems. Current multiplex architectures combining equipment from various suppliers with varying features and properties create a large space of potential interactions. This affects the effects of failures, their diagnosability etc. The possibility to formulate and exploit models at a very high level of abstraction is the basis for software tools that help the developer to cope with these combinatorial problems and take good design decisions.

² IDD (Integrated Design Process for Onboard Diagnosis) involved Fiat CRF, Magneti-Marelli SpA, PSA Peugeot Citroen, Renault, DaimlerChrysler AG, OCC'M Software GmbH and several universities and was funded by the Commission of the European Union (Project no. G3RD-CT199-00058). See [Brignolo et al. 01].

Besides the integration of activities related to design and on-board diagnosis, it is fairly obvious that model-based systems are able to support tasks in production and quality control (e.g. generation of test suites) and after sales. In particular, they provide a basis for seamless solutions for on-board and workshop diagnosis.

In summary, the technology presented in this paper does not replace current efforts to exploit advanced software engineering methods for the development of on-board software. Rather, it offers a systematic knowledge-based solution to mastering the complexity of the development process that originates from the fact that the software has to be developed in reflection of its mechatronic environment with all its complexity and variations.

References

[Bidian, et al. 99] Bidian, P., Tatar, M., Cascio, F., Theseider-Dupre D., Sachenbacher, M., Weber, R., Carlen, C.: Powertrain Diagnostics: A Model-Based Approach, proceedings of ERA Technology Vehicle Electronic Systems Conference '99, Coventry, UK, 1999 .

[Brignolo et al. 01]R. Brignolo, F. Cascio, L. Console, P. Dague, P. Dubois, O. Dressler, D. Millet, B. Rehfus, and P. Struss: Integration of Design and Diagnosis Into a Common Process, pages 53-73. VDI Verlag, Duesseldorf, 2001.

[Dressler-Struss 96] Dressler, O. , Struss, P. The Consistency-based Approach to Automated Diagnosis of Devices. In Brewka, G. (ed.), Principles of Knowledge Representation, CSLI Publications, Stanford, 1996, p. 267-311.

[Dressler-Struss 03] Dressler, O., Struss P. A Toolbox Integrating Model-based Diagnosability Analysis and Automated Generation of Diagnostics. International Workshop on Principles of Diagnosis DX'03, 2003, Washington, USA.

[Harms et al. 03] M. Harms, T. Ehlers, J.-U. Varchmin, A. Breuer. Diagnose im strukturierten Entwicklungsprozess STEP-X Haus der Technik: Elektronik im Kraftfahrzeug Stuttgart, 18.06.2003

[MentorGraphics 04]

http://www.mentor.com/products/cabling_harness/analysis_solutions/index.cfm

[Picardi et al. 04] C. Picardi, L. Console, F. Berger, J. Breeman, T. Kanakis, J. Moelands, S. Collas, E. Arbaretier, N. De Domenico, E. Girardelli, O. Dressler, P. Struss, B. Zilbermann: AUTAS: a tool for supporting FMECA generation in aeronautic systems. In: Proceeding of the 16th European Conference on Artificial Intelligence August 22nd - 27th 2004 Valencia, Spain, pp. 750-754

[Sachenbacher-Struss-Weber 00] Sachenbacher, M.; Struss, P; Weber, R. 2000. 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.

[Struss-Price 04] Struss, P., Price, C. Model-based Systems in the Automotive Industry, Artificial Intelligence Magazine, Winter 2004