Model-Based Systems in the Automotive Industry

Peter Struss and Chris Price

■ The automotive industry was the first to promote the development of applications of model-based systems technology on a broad scale and, as a result, has produced some of the most advanced prototypes and products. In this article, we illustrate the features and benefits of model-based systems and qualitative modeling by prototypes and application systems that were developed in the automotive industry to support on-board diagnosis, design for diagnosability, and failure modes and effects analysis.

Problems and Requirements in the Automotive Industry

The automotive industry was the first to promote the development of applications of model-based systems technology on a broad scale and, as a result, has produced some of the most advanced prototypes and products. Car manufacturers and their suppliers face increasingly serious challenges particularly related to fault analysis and diagnosis during the life cycle of their products. On the one hand, the complexity and sophistication of vehicles is growing, so it is becoming harder to predict interactions between vehicle systems, especially when failures occur. On the other hand, legal regulations and the demand for safety impose strong requirements on the detection and identification of faults and the prevention of their effects on the environment or dangerous situations for passengers and other people. Also, customer satisfaction is important to remain competitive and means that the manufacturer must eliminate breakdowns and reduce maintenance time and the number of misdiagnoses.

Another problem is that garage workshop staff need more training and assistance to make correct diagnoses for complex systems (for example, electronic control units are often replaced, and when examined by the manufacturer, no fault is found with them).

Finally, cars come in many variants of details and supplements, dependent on the make, year, or even almost individual customization (this problem is also relevant to suppliers delivering functionally similar subsystems to different manufacturers or for different models). This issue, termed the *variant problem,* is a major cost factor, multiplying the efforts dedicated to different diagnosis-related work processes.

All these issues must be taken into account during the many work processes composing the product life cycle. These work processes include the following:

Design for diagnosability: Has the system and, in particular, the placement of sensors been designed in a way that allows the detection, localization, and discrimination of faults?

Failure modes and effects analysis (FMEA): What is the impact of each possible failure of a system component?

Sneak circuit analysis: Are there states of a designed circuit that lead to an unintended (de)activation of certain functions?

Creation of on-board diagnostics: Design and implement algorithms that generate diagnostic hypotheses based on the sensor values available to the control units on the vehicle.

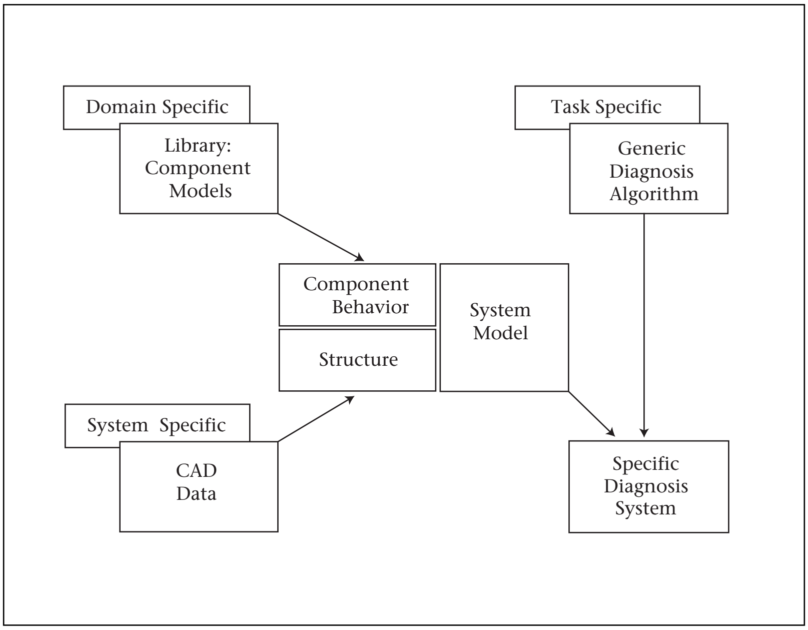


Figure 1. Automated Generation of Model-Based Diagnostic Systems.

Workshop diagnosis: Create diagnostics that guide and exploit tests performed on the vehicle in a workshop.

These tasks are knowledge intensive. Their support requires knowledge-based systems. A major part of the knowledge is knowledge about the technology. This knowledge is best captured by models. The next section explains why model-based reasoning is the best way to capture and use this knowledge and is followed by examples of how the models are used during the product life cycle.

The Answer Provided by Model-Based Systems

Model-based systems provide a good basis for solving the problems analyzed in the first section because they are based on a separation of the problem-solving algorithm from the model and on the compositionality of this model, which addresses the variant problem. Once a library of appropriate component models has been established, only a structural description of the respective device (for example, obtained from design data) is required to automatically generate a system model and, based on it, a problem-solving system dedicated to this device. Figure 1 illustrates this idea for the production of a model-based diagnostic system.

Because vehicles are assembled from standard components, and the behavior (and misbehavior in case of a fault) emerges from the behavior of these components, establishing a model library is feasible and entails collecting models of (correct and faulty) behavior of such standard components. Thus, this kind of model-based reasoning cannot be performed if the overall behavior of the system cannot be composed from the behavior of the components and the way in which they are linked.

Model-based systems enable the support of common engineering analysis tasks, such as those listed earlier, at early design stages because they are based on first principles and do not require experiential or empirical data from a physical prototype. In contrast, rule-based, case-based, or machine learning approaches cannot support early design analysis because they need experiential knowledge. They provide well-founded algorithms for automated problem solving that provide the guarantees for coverage and completeness of solutions required for safety-critical applications.

Models and a model library capture a considerable portion of the knowledge and information underlying various work processes during the life cycle, some of which were listed earlier. Hence, model-based systems provide a means for explicitly storing corporate technological knowledge and sharing and communicating it between different work processes (*horizontal integration*). This knowledge becomes accessible independently of time, location, and people.

The reusable nature of the knowledge and the guarantees of coverage on the algorithms promise reductions in design costs, a shortened product development life cycle, and reductions in time-to-market for new products.

Engineers generally work with numeric models, but in this type of work, the capability to exploit qualitative models (Forbus 1988) turns out to be crucial for several fundamental reasons:

First, in particular, in early design phases, only a partial specification of components and parameters is available, which prevents the use of numeric techniques.

Second, many tasks, such as FMEA or the generation of diagnostic manuals, aim at statements about classes of (fault) behavior and symptoms rather than specific instances. For example, the effect of a leakage of any size, rather than just a leakage of a specified size, has to be predicted.

Third, faults are defined as qualitative deviations from normal functioning (for example, flow through a pipe is reduced) rather than arbitrary discrepancies with respect to precise values (for example, flow is 4.12 gallons/minute but should be 6.73 gallons/minute).

Fourth, precise values often do not exist because the vehicle is operated in a noisy and widely unmeasurable environment, and only incomplete data are available (for example, about properties of the road surface).

Fifth, qualitative models provide an appropriate level of abstraction for modeling complex systems and processes where standard mathematical models do not exist or are not tractable (consider the combustion process or communication among the control units).

Sixth, they enable an intuitive representation and presentation of knowledge and information to the users.

The qualities outlined here mean that qualitative models often provide appropriate answers for a wide range of systems, with incomplete knowledge. Thus, automation of reasoning is enabled about the complex systems found in modern vehicles, early identification of safety and reliability issues, and generation of good diagnostics. Such automation can be done for many different vehicle variants with little extra effort.

In the following sections, we illustrate the features and benefits of model-based systems and qualitative modeling by prototypes and application systems that were developed in the automotive industries to support on-board diagnosis (Automating Production of On-Board Diagnostics), design for diagnosability (Integrated Design Process for On-Board Systems), and FMEA (Detecting Design Defects).

Automating Production of On-Board Diagnostics

Modern passenger vehicles contain a growing number of processors, which could be more than 100 for a high-end limousine. This computational power, originally utilized mainly to control the normal operation of various subsystems, such as the engine, the antilock braking system, airbags, beams, and air conditioning, is utilized more and more to also run software that deals with faults and abnormal behavior. It has a threefold purpose:

First is the detection of faults, which is, for example, required by U.S. regulations for emission-related problems.[1]

Second is the triggering of so-called recovery actions, that is, a different control scheme for a subsystem that allows for its continued, though limited, operation under fault conditions, for example, by limiting certain performance parameters.

Third is providing information for subsequent fault localization in the workshop, which typically happens by storing a fault code that represents a symptom (for example, "open circuit") rather than a particular component fault and, hence, is only a starting point for further testing.

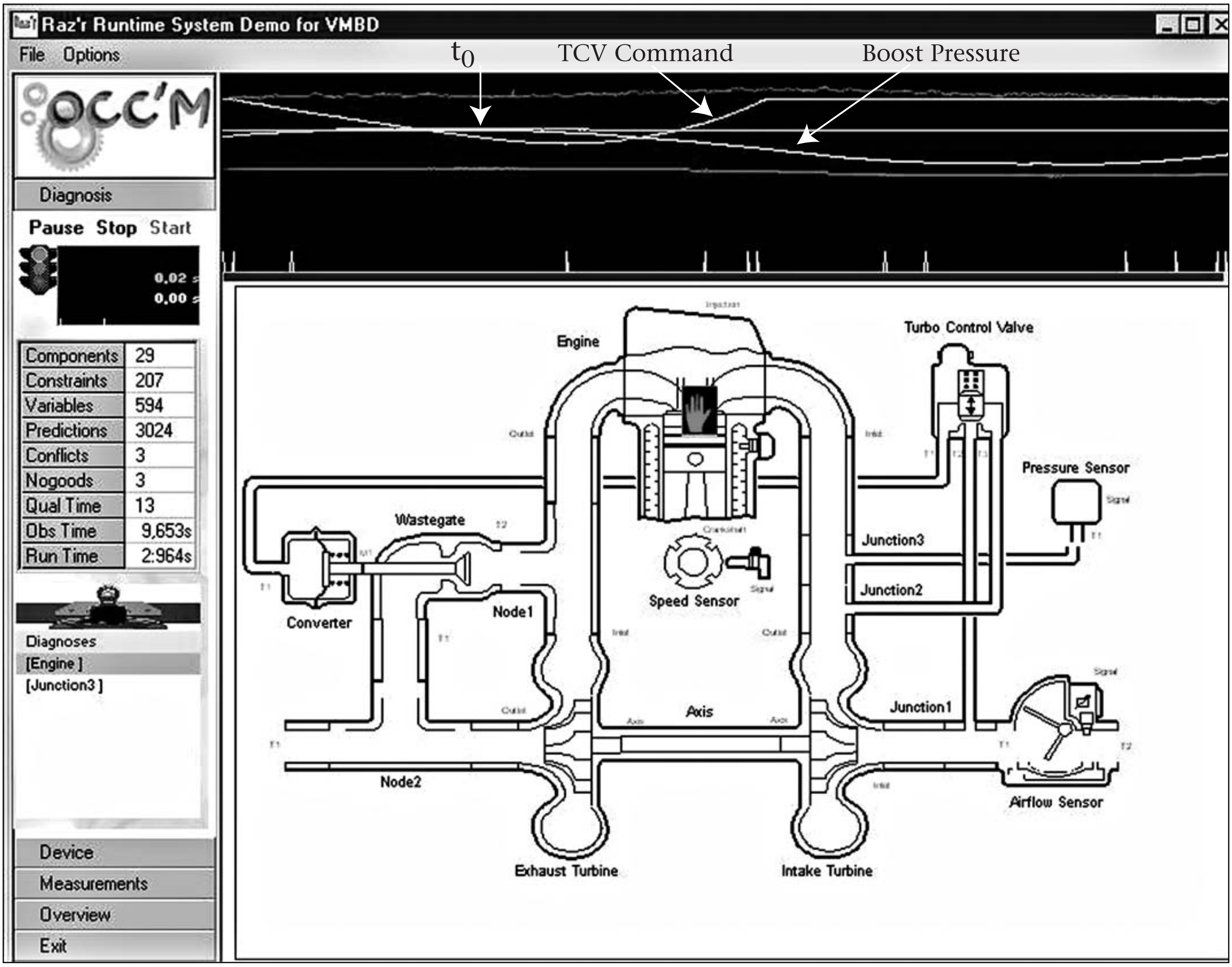
The pressure on car manufacturers to improve on-board diagnostics is high. It is needed to achieve compliance with legal restrictions, avoid overly restrictive recovery actions and customer dissatisfaction, and reduce after-sales costs by providing a narrower focus for maintenance in the workshop. In particular, supporting the production of after-sales costs is crucial for the worldwide operation of car companies because it is close to impossible to guarantee the requested high and up-to-date skills and information level of maintenance staff all over the world.

Any attempt to produce optimal on-board diagnostics faces the general problems discussed in the introduction, particularly high and repetitive efforts because of the variant problem and the necessity to reflect this goal early in the design process. Model-based systems promise to provide a new methodology and new software solutions that are needed to address the requirements for both reliable and efficient diagnostics of vehicles and systematic and economic processes for generating them. This promise has prompted the strong interest of the European car industries in this technology and why the Vehicle Model-Based Diagnosis (VMBD) Project was started in 1997.[2]

The Task: On-Board Diagnosis of Black Smoke Problems

The goal of VMBD was to run model-based diagnosis on board real demonstrator vehicles. In the following, we present a case study (more details are given in Sachenbacher, Struss, and Weber [2000]). Another on-board demonstrator is described in Cascio et. al (1999). Because increased legislative and customer demands have led to new requirements for aspects related to emissions and performance of the system, the case study focused on effects that involved incomplete fuel combustion in a diesel engine because of an excessive quantity of fuel injected or insufficient airflow to the engine, resulting in increased carbon emissions (called *black smoke* problems). The experimental environment was provided by the turbocontrol system of a Volvo 850 TDI demonstrator vehicle.

A schematic of this system is contained in the screenshot of the demonstrator system in figure 2. The air is taken in and the airflow measured on the right bottom part of the structure. The intake turbine compresses the air (with the pressure measured by a sensor) and feeds it to the combustion chamber of the engine (top middle). The exhaust gases exiting to the left drive the exhaust turbine, which is



Components	29
Constraints	207
Variables	594
Predictions	3024
Conflicts	3
Nogoods	3
Qual Time	13
Obs Time	9,653s
Run Time	2:964s

Diagnoses
[Engine]
[Junction3]

Figure 2. Screenshot of the Model-Based Run-Time Diagnosis Prototype for the Turbocontrol Subsystem.

connected to the intake turbine. Its speed can be influenced by the waste-gate valve, which is controlled by the pressure-driven converter. This pressure in the control pipe is, in turn, determined by the turbocontrol valve (top right), an actuator controlled by the engine electronic control unit (ECU).

Types of failures that can lead to black smoke symptoms involve leakages in pipes, malfunctions of valves (for example, stuck-at-open or stuck-at-closed), increased friction in bearings (resulting in a delay of actuators), or signal disturbances because of electrical failures. The demonstrator vehicle included facilities to create some of these failures. For example, it had a valve installed in the air hose between the turbine outlet and the engine intake manifold that could be opened to simulate a leakage. If such a leakage is too large to be compensated for, it will lead to insufficient oxygen supply to the engine and, hence, the potential for incomplete combustion.

The on-board diagnosis prototype was to use only signals available to the standard ECU without additional sensors. These signals were from the boost pressure, airflow, and engine speed sensor and the actuator signals indicating the amount of fuel quantity injected and the position of the turbocontrol valve.

This application imposes a number of requirements that are typical for on-board diagnosis of a broader class of subsystems:

First, the system has a dynamic behavior described by continuous variables.

Second, there are relatively few observables.

Third, part of these observables are very noisy signals (see the signals displayed in the upper window of the screenshot in figure 2).

Fourth, real-time performance has to cope with a high frequency of signals (with data coming in roughly every 10 milliseconds, this process is one of the fastest on a vehicle).

The Basis for the Model-Based On-Board Diagnosis System

The solution produced in the project was based on a compositional qualitative model of the turbocontrol system and its exploitation in a so-called consistency-based diagnosis system. In the following, we briefly describe basic principles of this solution.

Qualitative Deviation Models According to the principles discussed in The Answer Provided by Model-Based Systems, a library of component models was built rather than a closed model of the entire turbocontrol system. For most components of the system, such as pipes and valves, there exist straightforward mathematical models for describing their physical effects, mainly in terms of airflow and pressure. Rather than using these models as numeric ones, they were abstracted to a qualitative form. The resulting qualitative models capture, in particular, interdependencies among deviations of values of parameters and variables. For example, the model of a valve contains a constraint

$$[\Delta q] = [A] \cdot ([\Delta p_1] - [\Delta p_2]) + [\Delta A] \cdot ([p_1] - [p_2]) - [\Delta A] \cdot ([\Delta p_1] - [\Delta p_2])$$

on the signs of the deviations of pressure ($[\Delta p_i]$), flow ($[\Delta q]$), and area ($[\Delta A]$), where $[x]$ means sign(x). This constraint allows, for example, to infer that an increase in p_1 ($[\Delta p_1] = +$) will lead to an increase in the flow ($[\Delta q] = +$) if p_2 and the area remain unchanged ($[\Delta p_2] = 0$, $[\Delta A] = 0$), and the valve is not closed ($[A] = +$). The deviations can be related to any reference value, often a nominal value, but also a previous value, in this case representing a kind of qualitative derivative.

Based on the structural description, graphically represented in the main window of figure 2, a model of the turbocontrol system is generated automatically from the component models and imported by the diagnosis runtime system (as an XML file).

Consistency-Based Diagnosis

The diagnosis run-time system was provided by RAZ'R, a commercial system of OCC'M Software,[3] which is an implementation of consistency-based diagnosis (de Kleer, Mackworth, and Reiter 1992; Dressler and Struss 1996).

This technique considers diagnosis as a search for device models that are consistent with the given observation about the actual behavior. Based on the given observations and the device model, conclusions are computed about system parameters and variables (observed and unobserved). For each derived prediction, the set of component models involved in it is recorded. This information can be determined by the diagnosis system because the device model has a structure that reflects the device constituents. If a contradiction is detected —that is, conflicting conclusions for a variable occur (fault detection)—the set of components involved in it indicates which components possibly deviate from their intended behavior. Based on this information, diagnosis hypotheses are generated, that is, sets of faulty components that account for all detected contradictions (fault localization).

As an illustration, consider the scenario with the leakage at the engine intake manifold (Junction3 in the schematic in figure 2). The plots of the signal in figure 2 show that because of this leakage (the open valve), the sensed value of the boost pressure starts to drop at t_0. The ECU responds by changing the position of the turbocontrol valve (to a certain limit), which should counteract the pressure drop but fails to achieve this counteraction. To us, the qualitative characterization of the signals, in conjunction with the qualitative understanding of the intended functioning of the components, entails the conclusion that at least one of the participating components must be faulty. The same result is obtained by the model-based diagnosis system on the basis of the qualitative deviation model and an appropriate abstraction of the signals.

The demonstrator system uses a signal abstraction component that transforms each incoming vector of signals to the qualitative level the model is stated at. Only if this abstracted signal vector represents a new qualitative state is it entered into the diagnosis system. The resulting reduction of input and, hence, of diagnostic inferences, is immense, as illustrated by the following example: Instead of more than 1000 numeric vectors, only 12 qualitative ones (indicated by the peaks at the bottom of the signal window in figure 2) have to be processed.

One of these qualitative vectors states that the boost pressure drops, but all other signals (including the turbocontrol valve position) do not change. This result is in contradiction to the deviation model, which predicts a constant pressure from the steadiness of the valve position and the engine speed. The set of components whose models are involved in this prediction makes up the control path (turbocontrol valve, converter, and waste-gate valve) and the feedback loop from the intake turbine using the engine and exhaust turbine. One component in this fairly large set must be broken. The local-

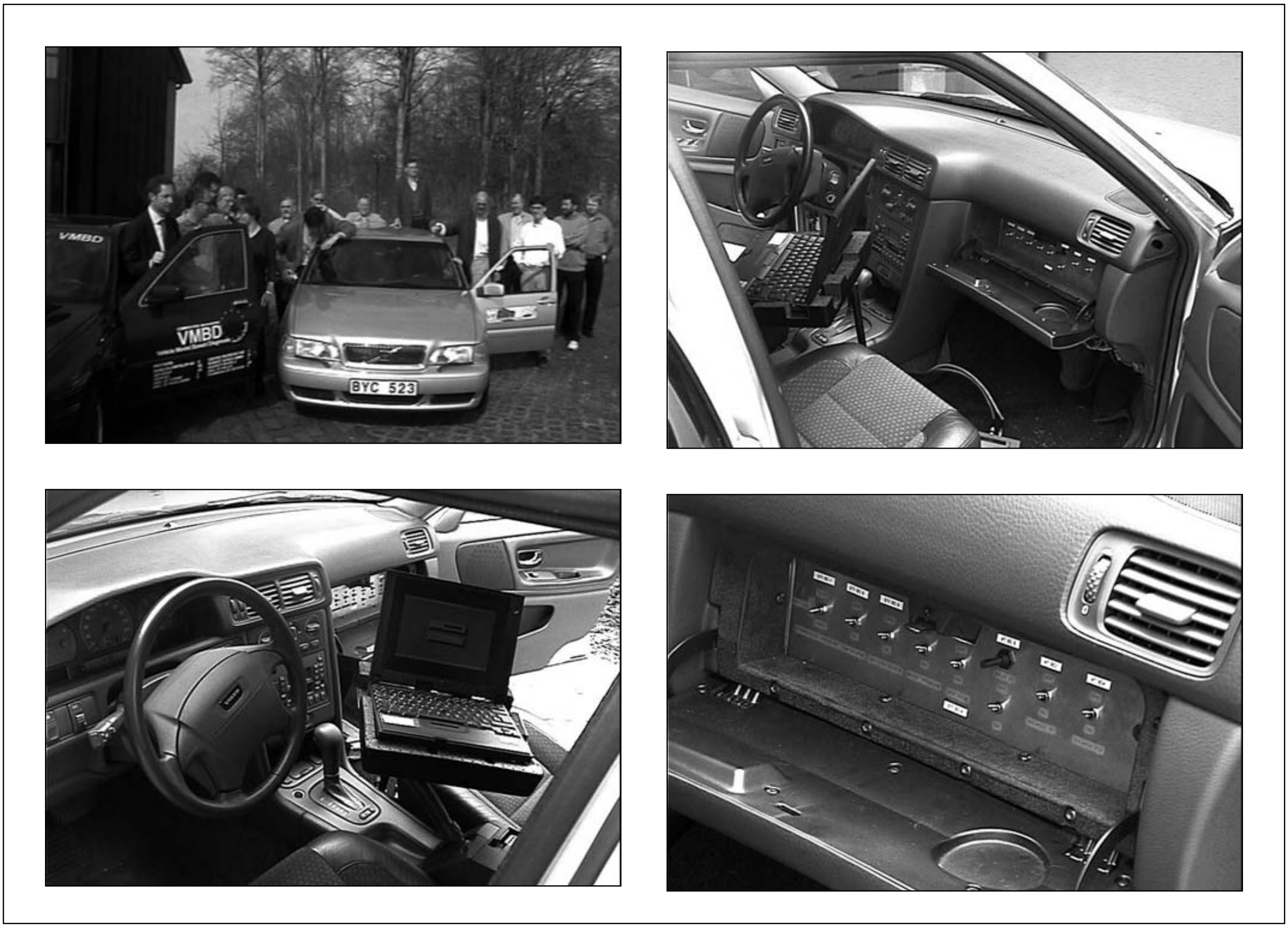


Figure 3. The VMBD Demonstrator System.

Top left: The two test vehicles used in the project (a Lacia and a Volvo). *Top right:* The Volvo test vehicle with the control unit (on the floor) and the cables that connect to the laptop. *Bottom left:* The laptop running the model-based diagnosis software. *Bottom right:* Switches in the glove compartment for turning on various fault conditions.

ization of the fault can be confined, combining evidence from several detected discrepancies; for example, an increasing airflow signal contradicts the decreasing boost pressure, yielding a different conflict set of components.

It is worth noting that the earlier inferences use only models of correct component behavior and no description of possible faults. If, in addition, models of faulty behavior are provided, the same technique (checking consistency of a model with the observations) can be used to discard particular faults (fault identification) or conclude correctness of certain components if the set of modeled faults is considered complete.

The Demonstrator System and Its Results

The demonstrator system was realized on a notebook that received the actual data from the ECU using a serial line while the vehicle was stalled, simulating full-load conditions. Figure 3 shows the installation on the demonstrator vehicle.

The screenshot in figure 2 shows the diagnostic results for a slowly opening leakage during an engine stall. The measurement runs for 9.75 seconds and yields 1064 quantitative observation vectors. The signal transformation module reduces them to only 12 qualitative observation vectors. The two single fault hypotheses generated by the system (displayed in the bottom left section) contain the component where the failure was actually induced (Junction3). The run time for the example was 2.87 seconds (on a Pentium PC running WINDOWS). Similar results were obtained for the other failures that could be induced on the car (but because of the available sensors, not always with a comparable quality of the fault localization). Thus, for the considered subsystem and scenarios, the performance of the on-board system is in the order of magnitude of real time.

There are several cornerstones of the success of this experiment:

First, the fundamental cornerstone is the model-based technology, more specifically, a compositional modeling methodology that is based on a library of component-behavior models. The component-oriented granularity ensures reusability of the model fragments and provides the model structure required for component-oriented diagnosis. Associating fault models with the respective components provides a principled way of capturing knowledge about faults in a modular and reusable way. This knowledge capture method contrasts with other AI approaches based on storing associations between symptoms and faults for each device in terms of rules or cases, such as neural nets and case-based reasoning, as well as other engineering approaches (trying to identify parameter deviations in a closed mathematical model of the entire device).

Second, consistency-based diagnosis was applied as state-based diagnosis; thus, the consistency check (models versus observations) is conducted only for each snapshot given by a vector of qualitative values, not that the actual behavior is tracked over time and compared to the simulated model. It can be performed as a set of constraint-satisfaction tasks and is important to achieving the required real-time behavior, what cannot be expected under a simulation-based technique, in particular, when a number of fault models have to be simulated. Nevertheless, the results remain the same under certain conditions (see Struss [1997]).

Third, the use of qualitative models is crucial for several reasons. They provide a finite, compact representation of the behavior of the system, which is important given the limitations of the on-board processors (the possibility of further reduction is discussed in Integrated Design Process for On-Board Systems). The noise in the signals vanishes behind the qualitative abstraction. The most important effect is the reduction of the number of input vectors and, hence, calls to the diagnosis algorithm, which enables the real-time performance.

The results of VMBD achieved the goal of providing evidence for the feasibility of using model-based systems and qualitative models for on-board diagnosis and strengthened the interest of the companies in introducing this technology. However, they were achieved by a specialized team of engineers and AI researchers who were familiar with this technology in a process that was unrelated to the current practice of developing on-board systems. Technology transfer is not simply the application of a principled solution to a real problem; it also has to deliver a work process for producing the application and, moreover, a work process that takes into account the people currently involved in performing the task, their education, skills, existing tools, and current practice. This challenge was the starting point of a follow-up project, integrated design process for on-board diagnosis (IDD).[4]

Integrated Design Process for On-Board Systems

Introducing model-based systems for on-board diagnosis into industrial practice in the automotive domain does not mean introducing a new task and activity. Thousands of engineers are already carrying out this task. Therefore, it is useless to simply invent a brand new process that is unrelated to the current one. In addition, it is impossible to determine and specify the software tools required without considering the ways tasks are viewed and solved and the tools used today. In particular, there are, of course, models developed and used in the current process, and a frequently asked question has always been, How do your diagnostic models relate to the currently used models?

Analysis of the Current Process of Design and Generation of Diagnostics

The IDD project started with an analysis of the current design processes of each of the industrial partners, with a focus on the integration of diagnosis-related processes into the whole design process of mechatronic subsystems. Based on this analysis, a merged process has been developed that is based on the similarities recognized, ignoring details and small differences. The abstraction of this process is used as a comprehensive reference for the current design processes. This analysis and its consequences are presented in more detail in Brignolo et al. (2001). Here, we focus on the inner design loop, which is concerned with the design of the ECU-based control system and components. Each iteration of this loop involves the design and verification of the control algorithms, FMEA, the development of on-board diagnostics, and the implementation of the ECU (hardware and software), as shown in figure 4. The verification step at the end of the first iteration is performed using models (software-hardware in the loop), whereas later in the design process, the physical system is used. Depending on the achieved results, there are several iterations, each one of them producing an advanced prototype.

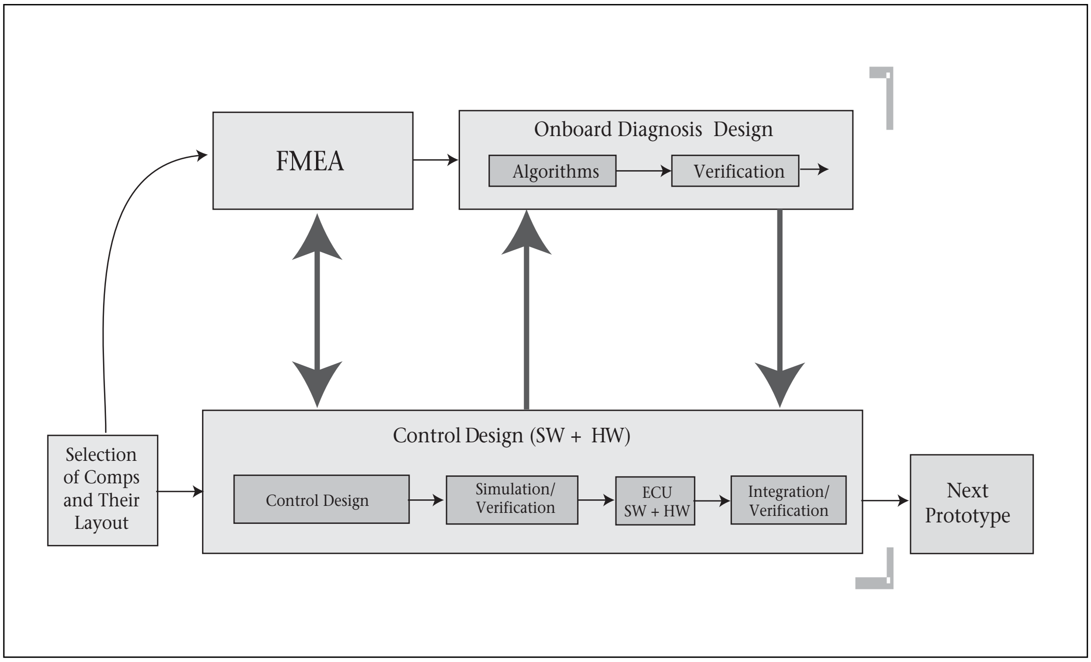


Figure 4. The Reference Process, One Iteration of the Inner Design Loop.

The result of the analysis showed that in current practice, the design process and the selection of components (particularly sensors and their location) is mainly guided and dominated by control design. FMEA and diagnostics development are carried out as subordinate or even subsequent tasks, often postponed until the final iterations to avoid the costly repetition of these tasks after every change in the control-oriented design steps. As a consequence, the considerations of failure effects and diagnosis have little impact on the design, and the developers of diagnostics simply have to accept a system structure and sensors determined by control purposes. If their demand for changes cannot be denied, it leads to further design iteration and, hence, additional cost and delays. The bottom line is that, at present, the important role of diagnosis in on-board systems is not reflected by the role that diagnostics development plays in the automotive design process.

Model-Based Tools for a New Process

It would be much better to conduct the steps of failure analysis and diagnostics development early in the design process, tightly integrated with the control design process. In this way, requirements from the various work processes can become explicit and influence the design early on, providing the potential for avoiding some design iterations and generating better results for diagnosability. Software tools that help to achieve this goal have to address (at least) two major requirements, which are addressed by model-based systems:

First, a fast and efficient flow of information about changes in the design between the different processes and a model of the system being designed must play a central role in a new process. There is a good basis for this solution in the current process because the verification of control algorithms in early phases is based on system models and simulation.

Second, the effort needed to perform FMEA and diagnostics generation has to be reduced, and they can effectively be supported or automated by computer tools based on the model; that is, they have to be model-based tools.

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 process was MATLAB/SIMULINK. Ideally, models created in these environments should automatically be transformed into qualitative diagnostic models that can be imported to the model-based tools.

Figure 5 shows the overall architecture of the new design support system, part of which was implemented as the prototypical IDD toolbox (see Dressler and Struss [2003]). The information flows consist of XML documents passed between the modules.

The project dedicated significant efforts to research on the automated abstraction of numeric models and the implementation of sev-

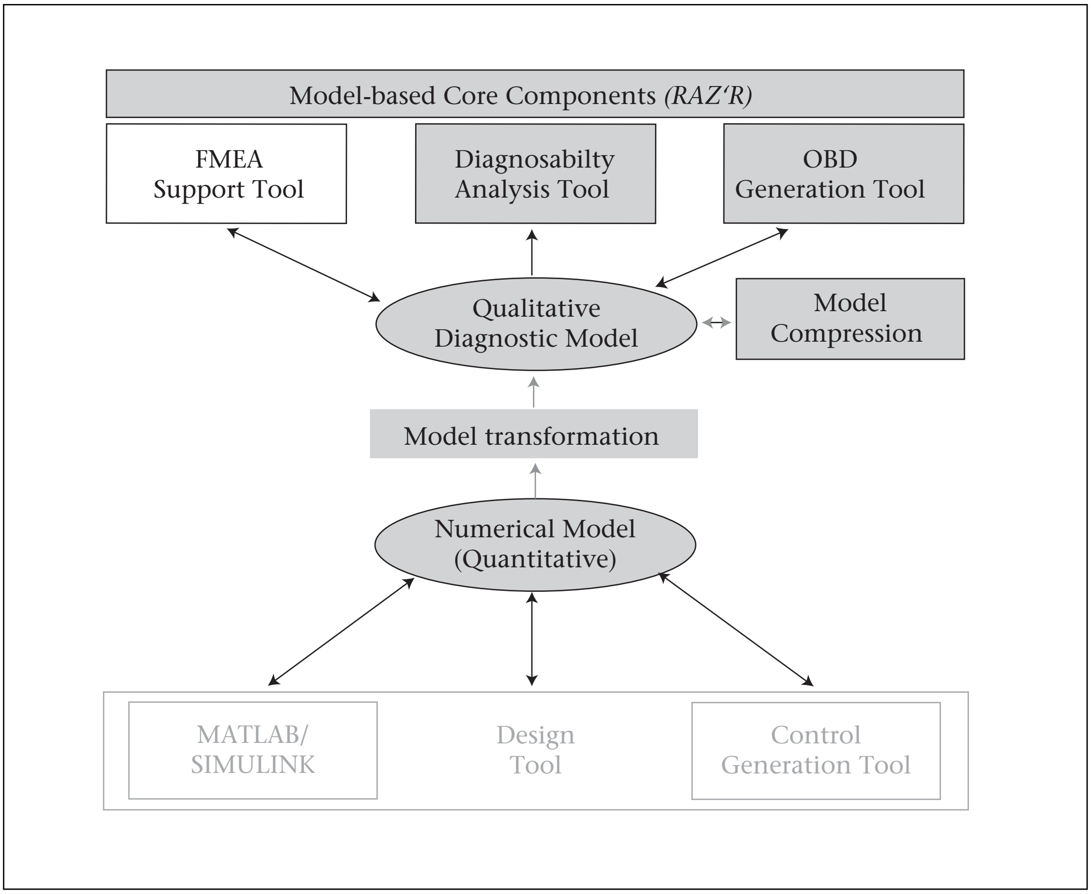


Figure 5. Tools Architecture for the New Process.

eral prototypes that generate qualitative models from MATLAB/SIMULINK models. The foundations of one of the implementations and a critical discussion of the practical experiences are presented in Struss (2002).

The model compression component eliminates irrelevant variables from the model. In the context of on-board diagnosis, this compression allows one to exploit the fact that the set of (potentially) observable variables is fixed and that all that matters is the association between values of these variables and the consistent diagnoses. All other variables (in particular, intermediate variables that are only the result of the compositional origin of the model) can be dropped, affecting not only the size of the on-board model but also the complexity, completeness, and performance of the algorithms that operate on the model.

On-board diagnostics generation delivers results in various forms: (1) a compressed model (in XML) as an input to the RAZ'R diagnosis runtime system that was introduced earlier; (2) C-CODEM, which is suited for 16-bit microcontrollers, including the Infineon C166/7 series; and (3) decision trees (XML and HTML) as specification input for diagnostics developers.

In the following discussion, we briefly present the intuitive foundation of the novel diagnosability analysis tool (for more details, see Dressler and Struss [2003]; Struss et al. [2002]).

Diagnosability Analysis

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

First, for a particular design and a chosen set of sensors, determine *fault detectability,* that is, whether and under which circumstances the possible faults considered can be detected (by the ECU), and *fault (class) discriminability,* that is, 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 diagnostic requirements: Full fault identification is usually not possible and also not required for on-board purposes because there is a limited set of possible recovery actions that can be performed by the control unit. They are to be se-

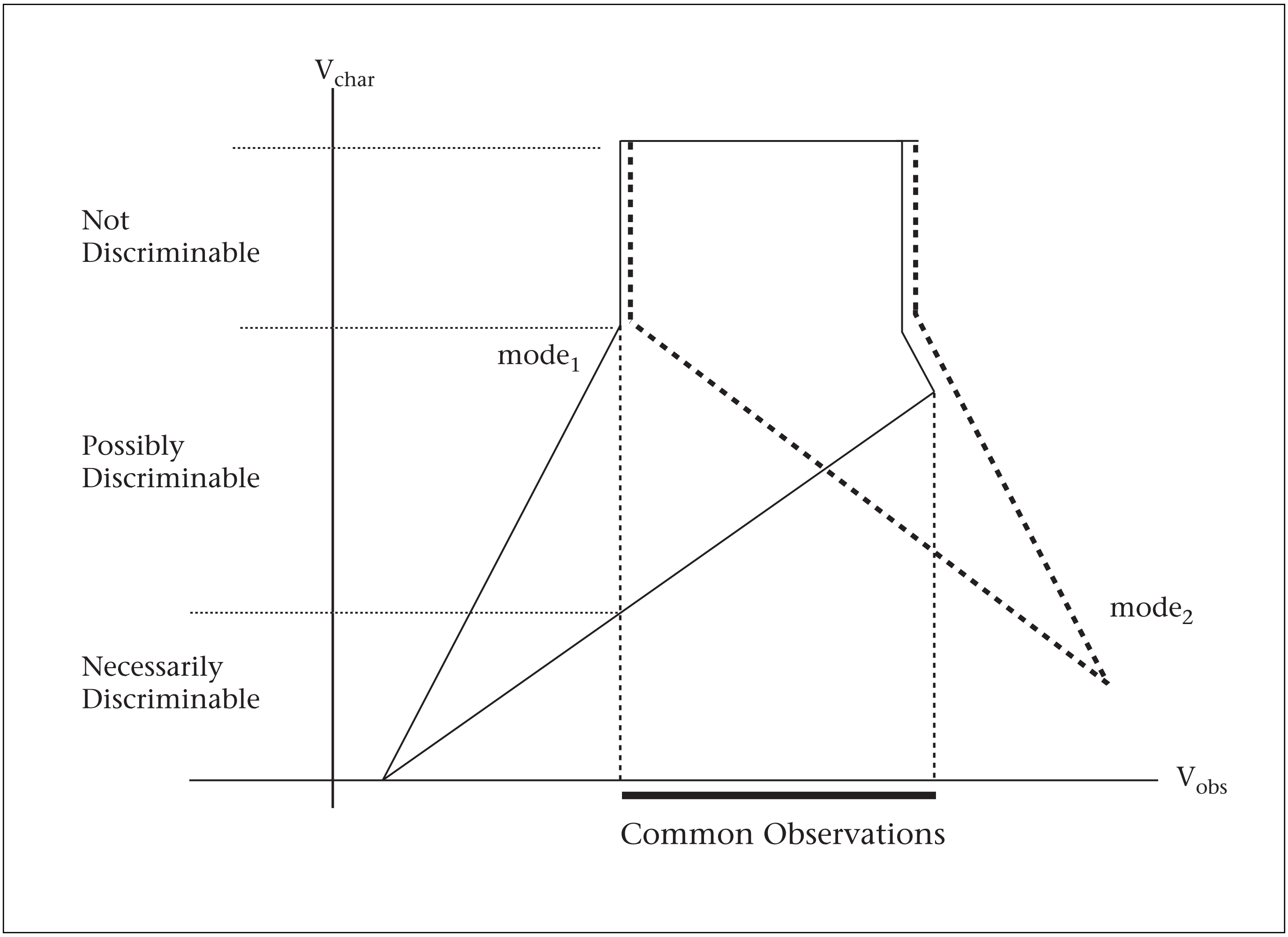


Figure 6. Three Categories of Discriminability of Two Behavior Modes.

lected dependent on the general type of fault and its severity rather than the individual fault. For example, only certain critical faults can require immediate shutoff of the engine, but 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 be 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. For each analysis, there is a fixed set of observable variables. In an on-board situation, this set is that of actuator and sensor signals. Whether or not two faults lead to different observations can depend on the situation they occur in. The situation can be characterized by quantities such as the engine temperature, the speed, and the position of the brake pedal. To describe the situations under which detection or discrimination is possible, we introduce a set of characterizing variables.

The abstract example in figure 6 provides an intuition about possible answers to the discriminability question. The vertical axis represents the characterizing variables and the horizontal axis the observables. There can be many unobservable variables, but the shown projection is all that matters.

Two different fault modes (or, more generally, behavior modes) are represented by two relations. As illustrated in figure 6, we can distinguish three different cases:

First, in the upper section, the relations cover each other; that is, for any situation in the projection of this intersection area, the observable set of consistent tuples for the two behavior modes is the same; hence, they cannot be discriminated from each other.

Second, in the lower section, they are totally disjoint; that is, any of the respective situations always lead to different system behavior and, thus, necessarily discriminate between the two modes.

Third, for all other situations, the two modes can possibly be discriminated because the actual response of the system might be outside one of the relations but is not guaranteed to be so.

It is obvious that precise definition can be given to these concepts (see Dressler and Struss [2003]). If the behavior relations are finite, as is the case for qualitative models, then the computation of the sets of situations and, hence, discriminability and detectability can be computed. To support sensor placement for diagnosability, the designer can vary the set and location of sensors and compare the results.

Insights Gained

The ambitious project provided a number of important results and insights, some of the most important being outlined in the following:

The model-centered solution provides a solid basis for the horizontal integration of different work processes that are disparate today, in our case, control design and and diagnostics generation.

In particular, the link to numeric models currently developed and used in the engineering domain is crucial. However, there are some major roadblocks. First, the prevailing modeling practice of engineers and the resulting models are not a suitable starting point for model-based systems: They are determined by their purpose in simulation (and even by the special simulation algorithm), are targeted at control under normal conditions, lack models of faulty behavior, and tend not to be component oriented. Second, the automated generation of qualitative models from numeric models requires more work on theoretical foundations and efficient algorithms, in particular, concerning answers to the fundamental question of how to compute the important qualitative distinctions (Struss 2002).

The use of qualitative models was essential not only for on-board diagnosis generation but even more for diagnosability analysis because they allow grounding the solution in operations on finite relations.

Contrary to the original plan, no FMEA tool was developed in IDD. However, another product provides evidence for the feasibility of FMEA. Unlike IDD, the AutoSteve system does not deal with arbitrary mechatronic systems but analyzes electrical subsystems.

Detecting Design Defects

The complexity of modern vehicles means that it is difficult for engineers to examine the interaction caused by every possible combination of input to the system or to consider the effects of every possible failure combination on the behavior of the overall system. Model-based simulation provides a basis for automatically detecting potential design problems in these kinds of cases. This section provides examples of tools to address these issues.

Failure Modes and Effects Analysis

FMEA is an application area where the benefits of model-based reasoning are evident. FMEA involves identifying the failure behavior of a system in the presence of any possible component failure. When performed by engineers, it essentially involves them in mentally simulating the behavior of the system for hundreds or even thousands of possible different component failures.

Because it involves so much effort, it is often not carried out until late in the design process, when the design has undergone any changes that are likely to occur, thus avoiding the prospect of repeating the generation of the FMEA. However, some of the major benefits of performing FMEA are lost. Where problems are detected by performing FMEA, then changes need to be made to the design, and like any changes to a design, the earlier in the process that they are made, the cheaper it is to make them.

Model-based reasoning provides an excellent basis for automated generation of FMEA reports. Qualitative models for components can be provided as soon as the general behavior of components is known, with comparatively little effort, and can be used to simulate the behavior of the system—both the correct behavior and the behavior when component failures are present. This approach has been particularly successful for generating FMEA reports for electrical systems because of the composition and the widespread reuse of electrical components. The following paragraphs describe the AutoSteve system (Price 2000), a commercially available FMEA generator for

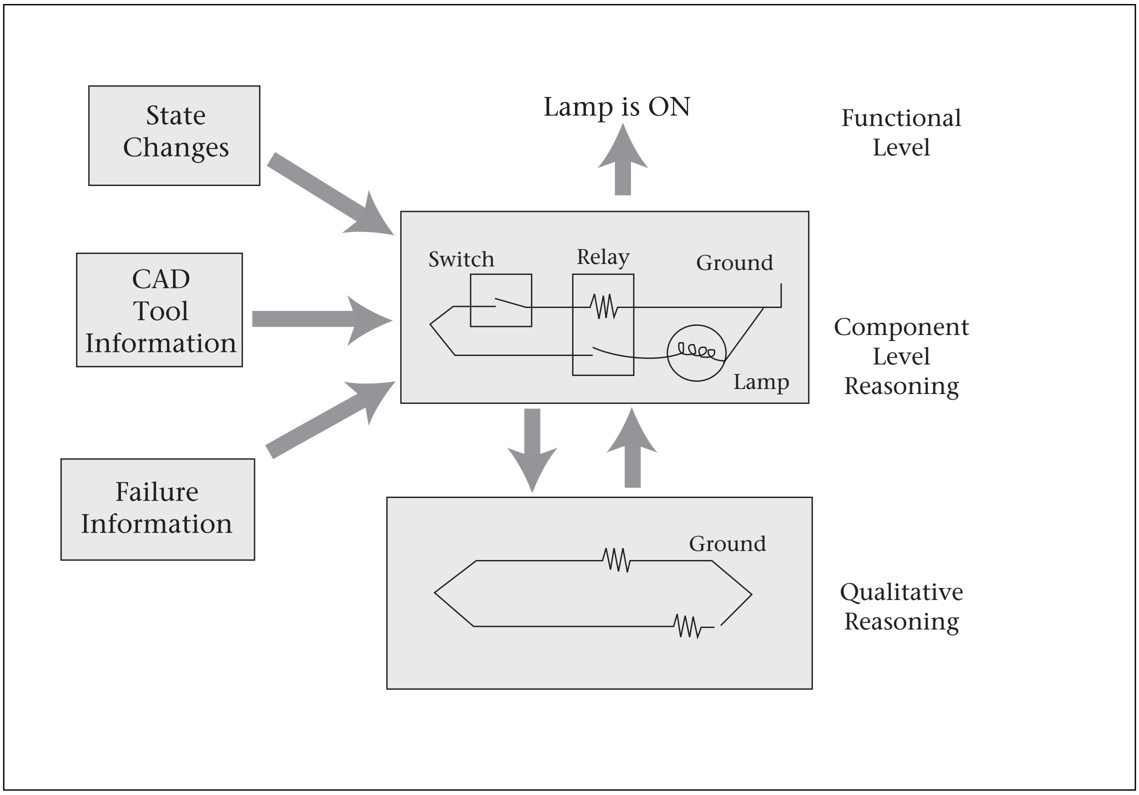


Figure 7. Simulation Levels in AutoSteve.

electrical systems based on qualitative modeling. This system is used by a number of major automotive and aeronautic manufacturers for performing electrical FMEA.

There are three levels to the simulation that is performed by AutoSteve, illustrated in figure 7.

At the bottom level, qualitative simulation is carried out on a network of resistors (Lee 1999), producing results that show where in a circuit current is flowing and in which direction.

The middle level is where the engineers specify models that map onto electrical and electronic components (Snooke 1999). A component description consists of a representation of the states that the component can be in, along with the resistive properties of the component in each state and the failure behavior of the component. This information is used to generate an appropriate network of resistors for the current state of the circuit, for use at the bottom level, and the results from the bottom-level simulation are used to calculate changes to the state of the circuit. This process continues until the circuit reaches a quiescent state or repeats previous states.

At the top level, the detailed behavior of the circuit is abstracted to obtain a description of the overall behavior of the system in terms appropriate to the engineers (Price 1998). Typically, this description will focus on the effectors of the system (motors, lamps, controllers) and show whether the system is achieving its intended functions.

Given the simulation capabilities described earlier, the production of an FMEA report is straightforward. First, the expected behavior of the circuit can be obtained by exercising the circuit simulator for the desired input states with a correct version of the circuit. Next, for each failure mode of each component in the circuit, the AutoSteve system simulates the behavior of the faulty version of the circuit for the same set of input states. The state of the circuit is abstracted to the functional level after each change, and differences from the expected functions in each state are noted. The difference between the expected functions and the actual functions in each state indicate the effects for that failure mode. For example, in a car security system, if the driver's door lock was switched, and doors started to lock, but one lock motor did not work, so all the doors were unlocked again, the failure would be that the locked function failed to occur when expected.

Automating the process of generating an FMEA report has a number of benefits, both for the engineers performing the FMEA and for their company.

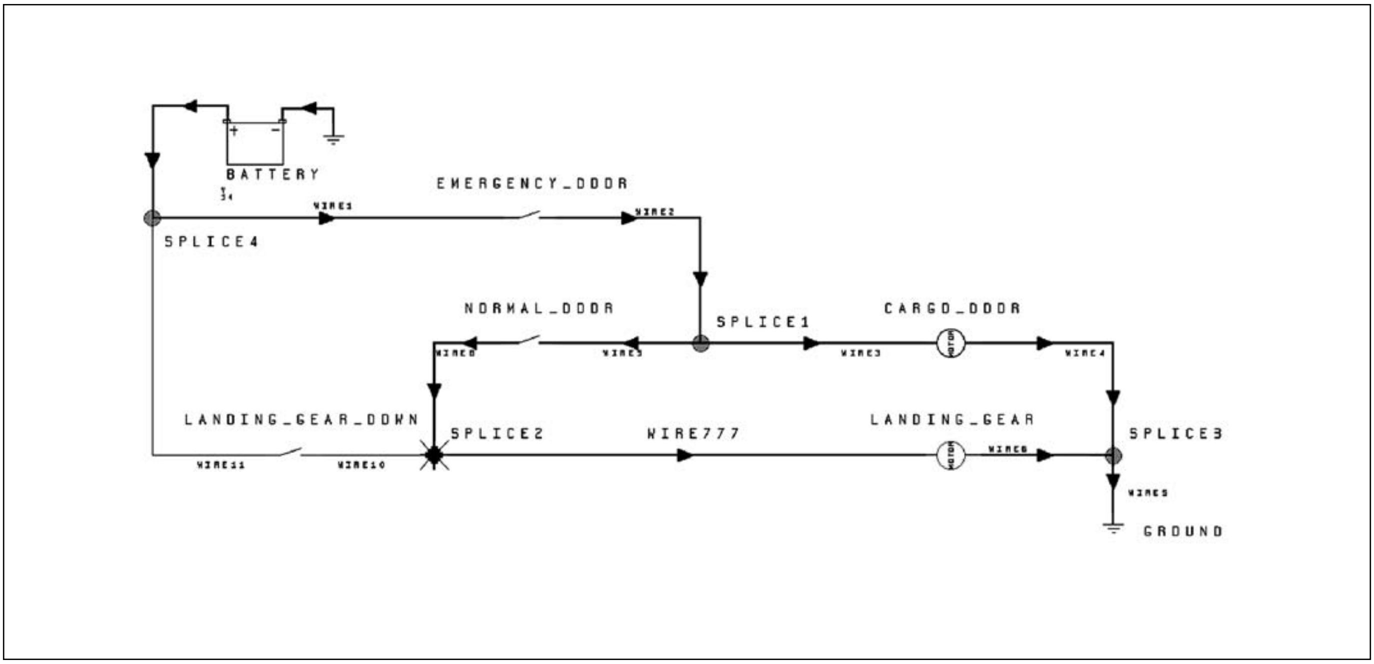


Figure 8. Illustration of Cargo Door Sneak Path.

First, the engineers receive much better feedback on the correct behavior of the system they have designed.

Second, FMEA is a tedious task, and automating the generation of failure effects allows the engineers to focus their time on more important tasks.

Third, the FMEA results are consistent (not always the case with hand-generated results).

Fourth, possible problems are identified much earlier in the design process, perhaps 12 months earlier than would have otherwise been the case.

Fifth, virtual prototypes of this kind enable the company to cut down on the number of expensive physical prototypes that need to be built.

Component-based model-based reasoning means that these benefits are obtained with minimum effort from the engineers. The variant problem discussed earlier, where there are different versions of a design for different countries and different price ranges, can be solved with very little extra effort for each variant. Typically, FMEA results can be made available for a different variant with the press of a button because all component models will exist for the new variant.

The original research in automating FMEA was driven by engineers who were tired of performing FMEA without automated tools. Now that the model-based FMEA software is commercially available, adoption of the technology within a company is often driven by engineers who need such tools to do their job properly as much as by the clear business case.

Sneak Circuit Analysis

A classic example problem (Savakoor, Bowles, and Bonnell 1993) concerns the cargo bay doors of a particular aircraft design, where operating the emergency switch for the cargo doors can cause the landing gear to lower unintentionally. This problem is illustrated in figure 8. Typically, such problems are caused when a wire, which was expected to provide current in one direction, is used in the opposite direction, causing a sneak path.

Sneak circuit analysis is the process of identifying and eliminating such sneak paths where they might occur. Where a wire is allowing current to flow in an unexpected direction, it can often be prevented with the addition of a diode to the design, but cost, weight, and reliability considerations mean that extra diodes should not be added to a design unless they are really needed.

To achieve sneak circuit analysis in AUTOSTEVE, it is necessary to declare the legal combinations of input under which each separate function will be active. AUTOSTEVE then performs an attainable envisionment (Forbus 1990) for the circuit,[5] exploring all combinations of input to the circuit (switches, sensors) and all internal states of components. Sneak circuits are detected as a function operating under an illegal set of input or not operating under a legal set of input (Price and Hughes 2002).

Foundations of Consistency-Based Problem Solving in a Nutshell

One of the achievements of AI in the area of model-based systems is the development of a rigorous theoretical basis stated in logical or mathematical terms that allows the defining of various tasks and their intended results and the proving of properties of solution algorithms for these tasks.

A behavior model is, interchangeably, regarded as a logical theory or a relation R over a set of variables that characterize a component or system

$$R \subset \mathrm{DOM}(\mathbf{v})$$

where **v** is a vector of system variables with the domain DOM(**v**). Under a component-oriented perspective, the elementary model fragments R_{ij} are related to behavior modes $\mathrm{mode}_i(C_j)$ of components C_j. An aggregate system (under correct or faulty conditions) is specified by a mode assignment

$$MA = \{\mathrm{mode}_i(C_j)\}$$

which specifies a unique behavior mode for each component mentioned. Its model is obtained as the join of the model relations R_{ij} where the description of the structure, that is, the interaction paths between the components, defines which variables are shared across the component models.

On this basis, various problems in design, monitoring, diagnosis, and maintenance can be described and solved by checking the consistency of the system models with some criterion external to the model. For example, monitoring and fault detection means checking whether the system model that corresponds to a system with all components working normally, MODEL_{OK}, is consistent with the behavior specification, GOALS, given a set of observations (for example, sensor readings):

$$\mathrm{MODEL}_{OK} \cup \mathrm{OBS} \cup \mathrm{GOALS} \vdash^{?} \bot,$$

or, if R_{OK} denotes the respective model relation:

$$R_{OK} \cap \mathrm{OBS} \cap \mathrm{GOALS} =^{?} \varnothing$$

If it is assumed that the system at hand is well designed, which means it satisfies the specification if all components operate normally,

$$\mathrm{MODEL}_{OK} \vdash \mathrm{GOALS}$$

the check can be reduced to

$$R_{OK} \cap \mathrm{OBS} =^{?} \varnothing$$

Fault localization and fault identification are solved by searching for mode assignments MA whose models are consistent with the observations

$$\mathrm{MODEL(MA)} \cup \mathrm{OBS} \nvdash \bot$$

or, stronger, entails the observed system response, $\mathrm{OBS}_{\mathrm{out}}$, given the observed state and external stimulus, $\mathrm{OBS}_{\mathrm{in}}$,

$$\mathrm{MODEL(MA)} \cup \mathrm{OBS}_{\mathrm{in}} \vdash \mathrm{OBS}_{\mathrm{out}},$$

which is called *abductive diagnosis*.

Consistency-based diagnoses can be generated by some sort of best-first search according to criteria such as maximal probability, minimal cardinality of the set of faulty components, and other orders on faults models.

Diagnosis is followed by therapy, that is, manipulating the system to reestablish its compliance with the behavior specification or some modified or intermediate behavior goals. Unless this therapy is confined to replacement of broken components, it can involve reconfiguration under exploitation of system redundancy (for example in networks, aircrafts, or spacecraft). Given a mode assignment as a result of diagnosis, reconfiguration can be achieved by searching for states of components (for example, valves and switches) that establish consistency with the (potentially modified) GOALS:

$$\mathrm{MODEL}(MA) \cup \mathrm{STATES} \cup \mathrm{GOALS} \nvdash \bot$$

More complex therapies involve structural changes.

The analysis of faults is also relevant during the design of systems. To support FMEA, one has to determine that the effects of a certain component fault (represented as a mode assignment MA) violate an intended function of the system. If the function is considered as part of GOALS, then the task might mean to check whether the fault is inconsistent with the function

$$\mathrm{MODEL}(MA) \cup \mathrm{GOALS} \vdash^{?} \bot$$

or might allow situations that need to be avoided

$$\mathrm{MODEL}(MA) \cap \text{NEG-GOALS} \neq \varnothing$$

Furthermore, the designer should perform a fault-detectability analysis, that is, an analysis of whether a fault can be distinguished from the normal behavior by observing a certain set of system variables (for example, given by the used sensors). If the projection to these variables is denoted p_{obs}, then a fault is definitely detectable if

$$p_{obs}(R_{fault}) \cap p_{obs}(R_{OK}) = \varnothing$$

Discriminability analysis looks for observable distinctions of the fault relations:

$$(p_{obs}(R_{fault1}) \setminus p_{obs}(R_{fault2})) \cup (p_{obs}(R_{fault2}) \setminus p_{obs}(R_{fault}1)) \neq^{?} \varnothing$$

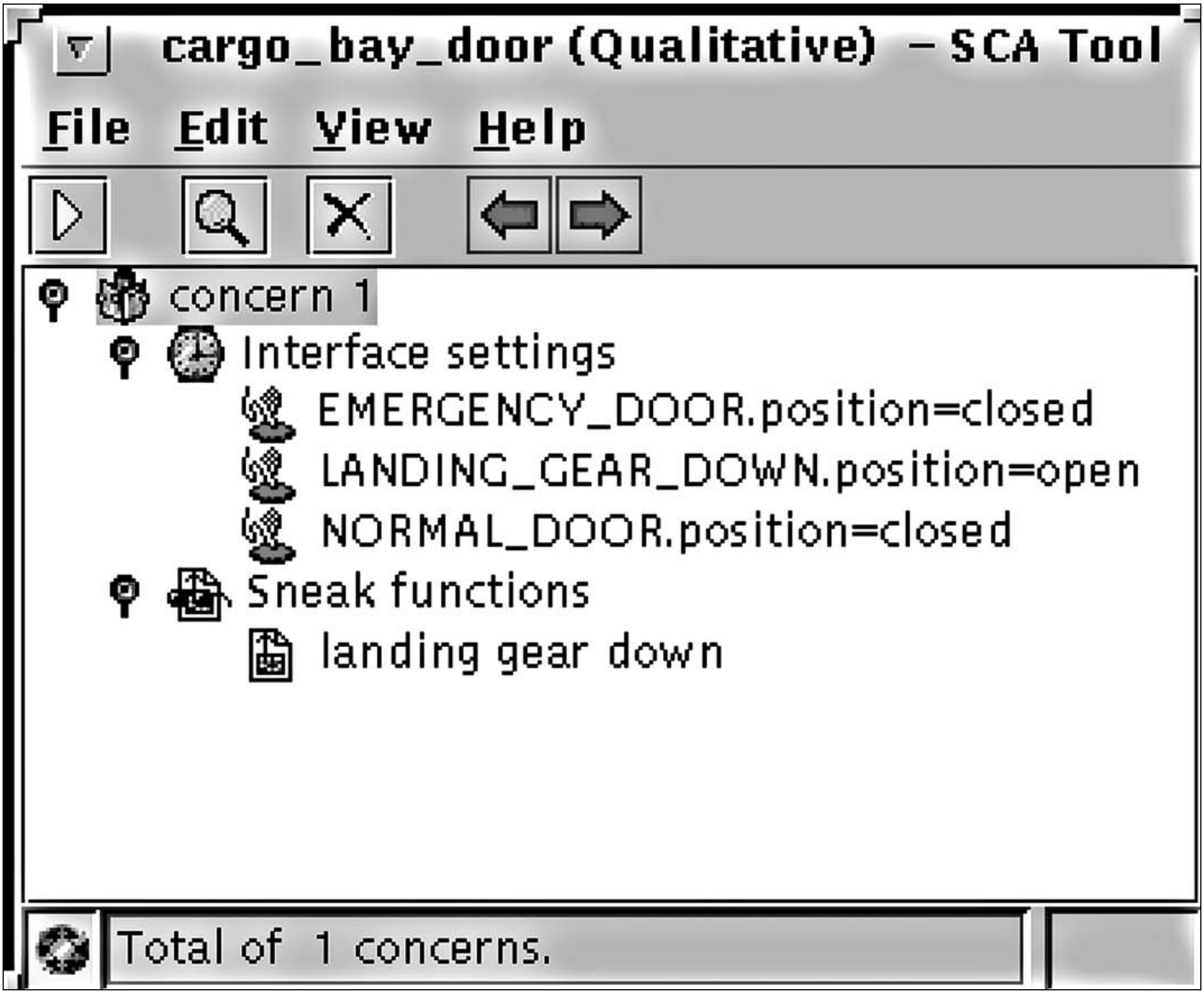
Test generation aims at finding stimuli to a system that are guaranteed to produce different output for two mode assignments, MA_1 and MA_2 (testing for correctness if one is the normal behavior; testing for discrimination if both are fault modes). If p_{cause} is the projection to the variables that can be influenced, then this set of stimuli can be computed as the complement of the set of stimuli that (potentially) generate the same observable behavior:

$$p_{cause}\,(DOM(\mathbf{v})) \setminus p_{cause}(p_{obs}(R_{MA1}) \cap p_{obs}(R_{MA2}))$$

Diagnosis and therapy generation (and also design) perform search in a space of mode and state assignments, respectively. In practice, it is prohibitive to perform this search and consistency check exhaustively on a set of enumerated assignments. There are two fundamental implications:

First, the generation of candidate assignments has to happen in a focused manner, which can be achieved by an incremental revision of partial assignments found to be inconsistent.

Second, the respective system models to be checked for consistency have to be generated by the problem solver rather than constructed manually beforehand. Hence, automated model composition becomes a requirement for the modeling system underlying the consistency-based problem solver.



concern 1
Interface settings
EMERGENCY_DOOR.position=closed
LANDING_GEAR_DOWN.position=open
NORMAL_DOOR.position=closed
Sneak functions
landing gear down

Total of 1 concerns.

Figure 9. AutoSteve Sneak Report for the Cargo Door Example.

This approach is more efficient and more accurate than other attempts at automated sneak circuit analysis, which operate by detecting current flowing the wrong way in components. For classic documented sneaks, it detects all possible sneak combinations and does not generate any spurious problem reports. Figure 9 shows the output from AutoSteve for the cargo bay door circuit example. Although this example is fairly simple, it works for much more complex circuits with many input.

Summary and Conclusions

This article has presented some selected examples of applying the AI technology of model-based systems and qualitative modeling in the automotive industry. It has illustrated some of the main requirements in this area, the foundations of model-based solutions, and their benefits. These examples are far from being comprehensive. Cautiously stated, the majority of car manufacturers and suppliers are at least exploring the potential of model-based systems, either by purchasing and evaluating commercially available tools or by building prototypical solutions and carrying out feasibility studies. A number of manufacturers and suppliers are already deploying solutions in industrial work processes and on vehicles.

The VMBD and IDD projects presented here joined several European manufacturers (Fiat, DaimlerChrysler, Renault, PSA Peugeot-Citroen, Volvo), suppliers (Bosch, Magneti-Marelli), and OCC'M Software as a supplier of AI technology. BMW and Volkswagen are carrying out efforts to move diagnostic technology on board. The Mercedes S has a diagnosis control unit whose database is generated with the support of models. Truck companies also face diagnostic challenges, in particular, under the requirements of emission-related on-board diagnosis (OBD 1993). DAF Trucks runs a project (together with Siemens and Click Software) that supports after-sales diagnosis by models generated by FMEA. Scania is aiming toward model-based diagnosis using a Power PC on the trucks. There is a demand for commercial AI software. R.O.S.E., OCC'M Software, and FirstEarth Limited provide tools for modeling and building model-based systems.

Ford Motor Company's use of AUTOSTEVE provides some indication of the value of this technology. The group of engineers at Ford who pioneered the use of AUTOSTEVE were recently awarded a Ford European Technical Achievement Award for their contribution to advancing electrical design analysis within Ford. One of the award winners stated, "The benefits of AUTOSTEVE are important. We can test and debug electrical systems before we ever wire them up for tryouts, so that confidence is close to 100 percent that the first prototype will be right the first time. The automated FMEAs have already confirmed adequate robustness of the design. This saves time in the development cycle with lower engineering resources and development costs. In fact, the pressure of program work is becoming so great that electrical CAE simulations will soon be the only way we can handle development and signoff of some subsystems."[6]

This list of parties interested in the automotive use of model-based reasoning, although incomplete, signals a significant AI success story. A research area that has begun addressing fundamental AI issues and developed a rigorous mathematical and logical theory has reached a stage where transfer of its results becomes a hot topic because it addresses urgent industrial needs. Application of the theoretical results was possible because a number of researchers in the field explicitly exposed themselves to the rough world of industrial problems. One of the insights gained in this enterprise was that this step did not bring research to an end or turn it into boring application programming exercises but provided stimuli and orientation to the research, brought new emphasis to fundamental research goals, and raised new goals by revealing serious limitations of existing solutions.

Further Research

Some of the most important lessons learned from deploying model-based reasoning in the automotive industry provide challenges for research.

Automated modeling was raised as an interesting research topic for model-based reasoning quite early on but has been somewhat neglected. Now, supporting and automating modeling is becoming a necessity for practical reasons: The creation of model libraries has to be efficient because otherwise, model-based technology will be less attractive. For many systems, models are already created in the standard engineering processes (particularly during design), but they are often numeric special-purpose or black box models that do not meet the requirements of modularity and generality. Nevertheless, current models, modeling practice, and modeling systems have to be seen as a starting point for model-based systems, and AI tools are needed to support appropriate modeling methodologies and the transformation of models.

Automated model abstraction from the numeric models available is expecially becoming an important issue because the required granularity of qualitative component models depends on the structure and parameters of the respective subsystem and the problem to be solved (validation of control requires a different model than on-board diagnosis, which can be different from a model needed for off-board diagnosis). Again, if a task-specific model with appropriate granularity has to be created manually rather than being generated from some base model, a model-based solution might become too costly (see Struss [2002]).

The main feature that model-based systems offer is the automation of reasoning steps. However, in many cases, mere automated reasoning does not pay off as such; supporting human activities is where the benefit comes in. For example, the automated generation of diagnosis hypotheses and useful tests might have no practical use if the result is not turned into plan that minimizes time and cost of human actions, such as disassembling a device and installing equipment. Thus, here is a challenge to model-based planning methods and techniques along the lines of Williams et al. (see article, also in this issue).

From a more general perspective, some successes and problems discussed in this article highlight a shortcoming that is typical of many AI fields: Often, researchers confine their interest to solutions of abstract "tasks." For example, diagnosis is still mainly understood as the task of inferring faults from a set of observations and some background knowledge (for example, a behavior model), that is, as a pure reasoning task, as stated earlier. Although a theoretical statement of the diagnostic task is appropriate for clearly specifying the desired solutions and developing inference mechanisms, there is no guarantee that it leads to any usable and useful solutions. To achieve useful solutions, it is necessary to consider the relevant work processes. It requires analyzing and modeling how people solve the problem in a real context, which education and skills they have, which kinds of tools and information they use, what the practical constraints are, how they cooperate and organize their work, and so on. We argue that modeling and supporting work processes should not only be done after solving the tasks and when addressing technology transfer issues but that it should be done before as well to define the proper tasks in the first place.

Ultimately, consideration of work processes also touches on a more social or cultural issue: The tools and solutions we offer are intruding on a vast field of existing theories, established techniques, educated people, and organized activities, many of them outside computer science. Bridging the gap to the world of engineering (and of engineers!) becomes an important task in the transfer of the model-based technology. Progress in this respect influences the transfer process as much as the technical results.

Notes

1. California's OBD-II Regulation, Section 1968.1, Title 13. 1993. California Code of Regulation, Resolution 93-40. See www.obdiiesu.com.

2. Vehicle model-based diagnosis (VMBD) 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 BE 95/2128). See Bidian et al. (1999).

3. Raz'r Version 1.6, Occ'm Software GmbH. See www.occm.de.

4. Integrated design process for on-board diagnosis (IDD) joined 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 G3RD-CT199-00058).

5. *Envisionment* refers to all the states a simulation can reach. See Forbus (1990).

6. See www.firstearth.co.uk/company/pr/Ford-ETA-Award.php.

References

Bidian, P.; Tatar, M.; Cascio, F.; Theseider-Dupré, D.; Sachenbacher, M.; Weber, R.; and Carlén, C. 1999. Powertrain Diagnostics: A Model-Based Approach. Paper presented at ERA Technology Vehicle Electronic Systems Conference '99, 9–10 June, Coventry, United Kingdom.

Brignolo, R.; Cascio, F.; Console, L.; Dague, P.; Dubois, P.; Dressler, O.; Millet, D.; Rehfus, B.; and Struss, P. 2001. Integration of Design and Diagnosis into a Common Process. In *Electronic Systems for Vehicles,* 53–73. Düsseldorf, Germany: Springer-Verlag.

Cascio, F.; Console, L.; Guagliumi, M.; Osella, M.; Panati, A.; Sottano, S.; and Theseider-Dupré, D. 1999. Strategies for On-Board Diagnostics of Dynamic Automotive Systems Using Qualitative Models. *AI Communications* 12(1–2): 33–43.

Console, L., and Torasso, P. 1991. A Spectrum of Logical Definitions of Model-Based Diagnosis. *Computational Intelligence* 7(3): 133–141.

de Kleer, J.; Mackworth, A.; and Reiter, R. 1992. Characterizing Diagnoses and Systems. *Artificial Intelligence* 56(2–3): 197–222.

Dressler, O., and Struss, P. 1996. The Consistency-Based Approach to Automated Diagnosis of Devices. In *Principles of Knowledge Representation,* ed. G. Brewka, 267-311. Stanford, Calif.: CSLI Publications.

Forbus, K. D. 1990. The Qualitative Process Engine. In *Readings in Qualitative Reasoning about Physical Systems,* eds. D. Weld and J. de Kleer, 220–235. San Francisco, Calif.: Morgan Kaufmann.

Forbus, K. 1988. Intelligent Computer-Aided Engineering. *AI Magazine* 9(3): 23–36.

Lee, M. H. 1999. Qualitative Circuit Models in Failure Analysis Reasoning. *Artificial Intelligence* 111(1–2): 239–276.

Price, C. J. 2000. AUTOSTEVE: Automated Electrical Design Analysis. In Proceedings ECAI-2000, 721–725. Amsterdam, The Netherlands: IOS.

Price, C. J. 1998. Function-Directed Electrical Design Analysis. *Artificial Intelligence in Engineering* 12(4): 445–456.

Price, C. J., and Hughes, N. 2002. Effective Automated Sneak Circuit Analysis 2002. In Proceedings of the Annual Reliability and Maintainability Symposium, 356–360. Washington, D.C.: IEEE Computer Society.

Sachenbacher, M.; Struss, P; and Weber, R. 2000. Advances in Design and Implementation of OBD Functions for Diesel Injection Systems Based on a Qualitative Approach to Diagnosis. Paper presented at the SAE 2000 World Congress, 6–10 March, Detroit, Michigan.

Savakoor, D. S.; Bowles, J. B.; and Bonnell, R. D. 1993. Combining Sneak Circuit Analysis and Failure Modes and Effects Analysis. In Proceedings of the Annual Reliability and Maintainability Symosium, 199–205. Washington, D. C.: IEEE Computer Society.

Snooke, N. 1999. Simulating Electrical Devices with Complex Behavior. *AI Communications* 12(1–2): 45–-59.

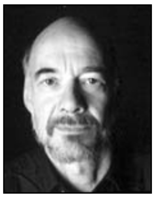
Struss, P. 2002. Automated Abstraction of Numerical Simulations Models—Theory and Practical Experience. Paper presented at the Sixteenth International Workshop on Qualitative Reasoning, 10–12 June, Sitges, Catalonia, Spain.

Struss, P. 1997. Fundamentals of Knowledge-Based Diagnosis of Dynamic Systems. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97), 480–485. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Struss, P. 1994. Testing Physical Systems. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94), 251–256. Menlo Park, Calif.: American Association for Artificial Intelligence.

Struss, P.; Rehfus, B.; Brignolo, R.; Cascio, F.; Console, L.; Dague, P.; Dubois, P.; Dressler, O.; and Millet, D. 2002. Model-Based Tools for the Integration of Design and Diagnosis into a Common Process—A Project Report. Paper presented at the Thirteenth International Workshop on Principles of Diagnosis, 2–4 May, Semmering, Austria.

Dressler, O., and Struss, P. 2003. A Toolbox Integrating Model-Based Diagnosability Analysis and Automated Generation of Diagnostics. Paper presented at the Fourteenth International Workshop on Principles of Diagnosis, 11–14 June, Washington.



Peter Struss is a professor of computer science at the Technical University of Munich and a managing director of OCC'M Software GmbH. He obtained his master degree in mathematics at the University of Goettingen, a Ph.D. in computer science at the University of Kaiserslautern, and his habilitation at the Technical University of Munich. The focus of his work is on qualitative modeling and model-based systems and the transfer of the technology into industrial applications. His e-mail address is struss@in.tum.de.

Chris Price is professor and head of the Computer Science Department at the University of Wales, Aberystwyth. Previously, he was managing director of FirstEarth Limited, a company formed to exploit the kind of technology described in this article. His research interests include the application of model-based reasoning and case-based reasoning to engineering problems. His e-mail address is cjp@aber.ac.uk.