Diagnosis in Automotive Applications

A Case Study with the Model Compilation Approach

Benno Stein[1] and **Oliver Niggemann**[2] and **Heinrich Balzer**[3]

Abstract. This paper addresses intricate diagnosis situations in modern cars. It employs a rather new and little known diagnosis paradigm, which is a mixture of model-based and heuristic diagnosis and which is called model compilation [5, 25, 26]. The basic idea is to simulate a model of the system under study in various fault modes and over its typical input range to compile a simulation database $\mathcal{C}$. A simplified rule-based behavior model is constructed from $\mathcal{C}$, in which long cause-effect chains are replaced with much simpler associations and which is optimized for heuristic classification of the fault being studied.

The main contribution of the paper is the application of this idea to the complex discrete event / continuous system models of modern cars. Our experiments, which are based on the latest simulation models from the automotive industry, are encouraging and may indicate a paradigm shift away from the diagnosis approaches used so far in this field.

Keywords automotive diagnosis, model compilation, simulation, data mining

1 INTRODUCTION AND DIAGNOSIS SETTING

Today's cars are complex mechatronic systems. Their reliability has improved despite their ever-increasing technological complexity. As a consequence, the diagnosis process itself became highly involved: Hardware systems and software systems together form a complex setup; data from various sensors is used by a complex network of distributed and inter-dependent software modules—modules that are specified at different phases in the development process and that are developed by different companies.

The present paper addresses this situation. It introduces a new technology for the diagnosis of automotive systems, discusses its pros and cons, and presents first application results. The paper is organized as follows: The remainder of this section describes the diagnosis setting and introduces the terminology and a taxonomy of possible and addressed fault types. Section 2 discusses existing model-based diagnosis approaches and, in particular, reviews developments in the field of automotive applications. Section 3 introduces the model compilation approach as both a formal framework and a concrete implementation, comprising an experimental setup, simulation tasks, and classification results.

[1] Bauhaus University Weimar, Germany
benno.stein@medien.uni-weimar.de
[2] dSPACE GmbH, Paderborn, Germany
ONiggemann@dspace.de
[3] University of Paderborn, Germany
hbalzer@upb.de

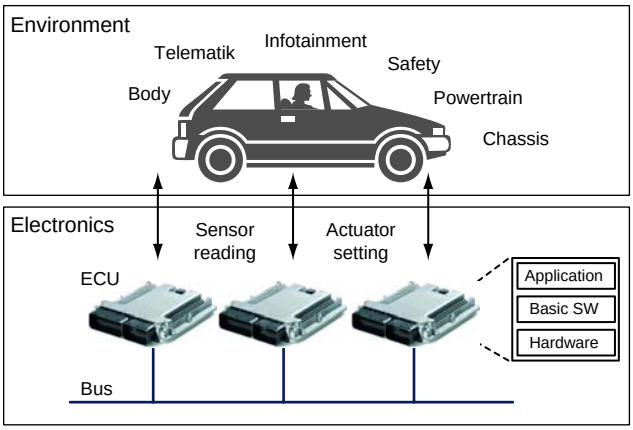

Figure 1. System of connected ECUs and vehicle environment. In today's cars up to 70 ECUs are connected to mechanical devices as well as to other ECUs and perform dedicated control tasks.

1.1 A Classification of Fault Types

Errors and faults might occur at different places within a vehicle. As shown in Figure 1, the system "vehicle" can be divided into two subsystems: electronics and environment. The electronics comprises the electronic control units (ECUs), the buses connecting ECUs, and the sensors and actuators which act as the ECU's interface to the environment.

The main purpose of ECUs is to control parts of the environment, e. g., the engine or the gear. ECUs do so by executing application software modules. In addition to the applications, ECUs contain the basic software that is needed to execute applications and to connect them to sensors, actuators, and buses. The environment comprises the rest of the vehicle and includes the driver, the road, the weather, and the following vehicle domains:

- *Powertrain.* Examples: engine, gear, drive shaft
- *Chassis.* Examples: brakes, damping, steering
- *Body.* Examples: light, wipers, climate control, key-less entry
- *Safety.* Examples: airbag, active safety belts
- *Telematics/Infotainment.* Examples: radio, navigation, telephone

A taxonomy of fault locations can be defined for this system; an extract can be seen in Figure 2. In the environment subtree mainly the already mentioned vehicle domains are differentiated. In the electronics subtree, faults can occur (*i*) in ECUs, (*ii*) in connection with buses, (*iii*) in the context of the power supply, or (*iv*) in sensors, actuators, or their wiring. Note that both the hardware and software in ECUs can fail. The latter is differentiated with respect to errors in application software or in basic software.

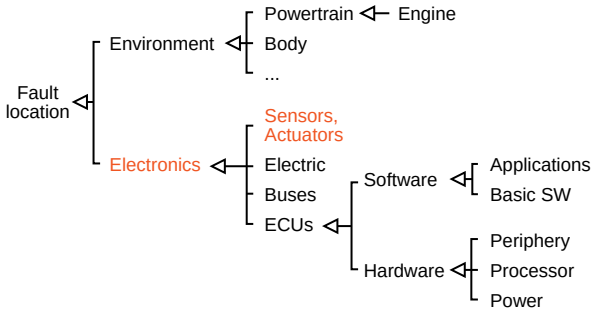

Figure 2. A taxonomy of fault locations. The highlighted classes of faults are addressed in this paper.

1.2 Diagnosis Based on Symptom Detection

The purpose of the diagnosis systems in a vehicle is to identify faults and to initiate appropriate countermeasures. Fault identification suffers from an inherent problem: Faults cannot be observed directly, only their effects can be measured by the vehicle's electronics. A fault in the engine, for example, leads to unusual readings for sensors such as engine temperature or revolution. So diagnosis can be seen as a three step procedure:

1. Identification of unusual vehicle conditions by comparing the expected vehicle behavior with the observed behavior. Detected discrepancies are called *symptoms*.
2. Abduction of faults. Based on the symptoms, possible underlying faults are inferred.
3. Initiation of countermeasures, such as a warning to the driver.

Symptoms are detected in the software modules on the ECUs. Figure 3 shows a simplified ECU structure from a software point of view: Sensor readings or bus signals are read by application software modules via different hardware and software layers (upper half of the diagram) and written to actuators or buses (lower half of the diagram). Note that symptom detection may be implemented only by driver software and application software modules; i. e., all symptoms from all fault locations are detected by these software modules, making it difficult to differentiate between faults. Symptoms fall into two main categories:

- *Local Symptoms.* Such symptoms occur directly at the fault location. An example is a short circuit in a cable connected to an ECU. Faults causing such symptoms can be identified fairly easily. In the example, the corresponding I/O driver may detect the wrong voltage level. Generally speaking, there are several established methods for detecting such faults.
- *Remote Symptoms.* Other symptoms occur at a distance from the fault location. Examples are (*i*) faults in the environment, e.g., engine problems or (*ii*) bus errors that influence all the connected ECUs. Clearly, the detection of faults causing symptoms like this is significantly more challenging.

Remote symptoms have occurred much more often over the last few years, leading to an increasing number of intricate diagnosis scenarios. This is mainly due to the development towards more complex applications, say, towards large and distributed software systems. Driver assistant functions such as active cruise control (ACC), active front steering (AFS), and lane keeping support are examples. These systems have in common that they are distributed over several ECUs and that they often rely on other, existing software modules. Faults occurring at one module of such a distributed software system

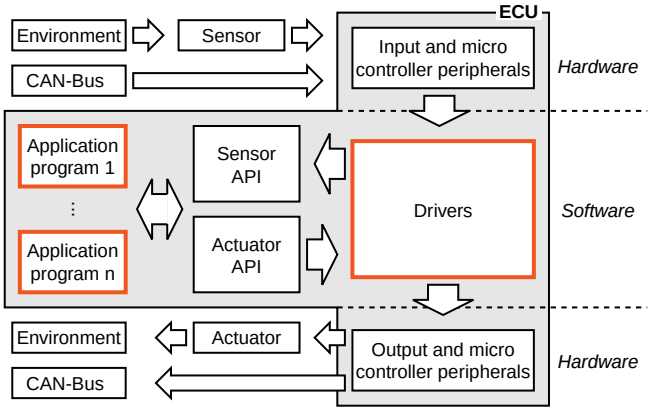

Figure 3. An ECU is organized as different hardware and software layers. Faults may occur in any layer but are not observable until the driver layer or the application layer.

may trigger further faults within other connected software modules. This leads to the detection of a large number of symptoms on several ECUs.

The model compilation methodology presented in this paper is especially suited to detecting these new fault types. Section 3 will outline the main reasons for this.

2 STATE OF THE ART

The diagnosis of technical systems as a research field is largely governed by modeling questions:

- Which modeling strategy is appropriate, a deep behavior model or a shallow rule model?
- Is a distinction between a correct behavior model and a fault model necessary?
- Is the diagnosis process controlled by heuristic expert knowledge, and, if so, how should it be represented?
- How can diagnosis knowledge be maintained or kept up-to-date?
- Is a modeling able to integrate knowledge of unforeseen faults?

Approaches over the last few years have predominantly relied on the model-based diagnosis paradigm [6, 20]. The basic idea is to have a model running in parallel to the real system and to use the simulation results to detect discrepancies between the expected and the observed behavior. Such a model is called a behavior model here, and denoted $\mathcal{M}$; Subsection 3.1 provides a precise specification of its elements.

In theory, possible faults causing symptoms can be found by analyzing the behavior model, $\mathcal{M}$, in reverse direction, from symptoms to faults. In practice, however, such an inverse simulation of $\mathcal{M}$ leads to a complex analysis problem that is generally not tractable. Hence, a specialized *diagnosis model*, $\mathcal{M}_D$, is often constructed by domain experts, where fault deduction is treated in a forward reasoning manner.

Model-based approaches employ various algorithms. A typical approach to fault detecting during operation is described in [27]: The control software of autonomous robots is monitored by means of observers, which are a certain form of a behavior model running in parallel. Here, $\mathcal{M}$ is modeled with propositional logic, and Reiter's hitting set algorithm is employed along with a theorem prover to infer faults from symptoms [8, 20].

[1] introduces a so-called "timed failure propagation graph", which is in the role of $\mathcal{M}$ and which defines the way a fault propagates through the system. Additionally, information about the time range for fault propagation between the system components is exploited. Discrepancies between the expected behavior of a signal and its observed behavior trigger diagnostic inference. The same graph is used for this, say, $\mathcal{M} = \mathcal{M}_D$.

[18] discusses diagnosis for systems that comprise hardware components and software components. A probabilistic, hierarchical, constraint-based automaton is used for both $\mathcal{M}$ and $\mathcal{M}_D$. The diagnosis task is formulated as a constraint satisfaction problem and is solved by optimization techniques.

Since [4] showed that model-based diagnosis can be applied to program debugging, many researchers started working in this area, among others [9, 14, 17]. In [14], a technique similar to constraint propagation is used in combination with model checking. In [9], component-based software is monitored, and the behavior of the software components is modeled with Petri nets.

There are also approaches to detecting faults in distributed systems or multi-agent systems, and recent results can be found in [12, 15, 16, 22, 23, 31].

2.1 Diagnosis in Automotive Applications

In the automotive industry, model-based approaches are used quite frequently. A main distinguishing feature is whether they are used onboard, to diagnose faults during a vehicle's operation, or offboard, in a garage for instance. Note that legal requirements concerning gas emission and safety regulations require more effective diagnosis systems for onboard diagnosis.

[19] employs a correct behavior model $\mathcal{M}$ and various fault models to identify sensor faults and leakages of the air-intake system of an engine. The diagnosis system uses a framework of structured hypothesis tests to decide which of the fault models can explain the measured data.

Various model-based approaches rely on a qualitative model, such as in [30], where consistency-based diagnosis is used for onboard diagnosis. Qualitative models, or more precisely, qualitative deviation models for onboard diagnosis, are also used in [3], where diagnostic situations are simulated. The outcome of the simulation is used to build a decision tree for the diagnosis system.

Other model-based approaches are discussed in [13, 24], where a specially constructed constraint network is used to model the behavior of combustion engines. Rules for the diagnosis, say, $\mathcal{M}_D$, are automatically generated from $\mathcal{M}$.

The approach applied in this paper is based on the model compilation paradigm that was introduced in [11, 25]; the following section applies the idea to real-world automotive diagnosis.

3 THE MODEL COMPILATION APPROACH

Model compilation is a diagnosis approach that combines the model-based paradigm and the heuristic paradigm in the following four steps [25]:

1. *Simulation.* A database, $\mathcal{C}$, is compiled by simulating the system under study in various fault modes and over its typical input range.
2. *Symptom Computation.* By comparing the faultless simulation to simulation runs in fault modes a symptom database $\mathcal{C}_\Delta$ is built up.
3. *Generalization.* Using cluster analysis or vector quantization, the numerical values in $\mathcal{C}_\Delta$ are abstracted towards intervals.
4. *Learning.* Data mining and machine learning are applied to learn a mapping from symptoms onto the set of fault modes; the resulting classifier can be seen as a "compiled diagnosis model".

Since this process can be completely automated, the approach has the potential to combine the advantages of the model-based philosophy, such as behavior fidelity and generality, with the efficiency and robustness of a heuristic diagnosis system. Especially in connection with the diagnosis in automotive applications, the following advantages shall be emphasized:

- The substantial number of existing libraries of behavior models, along with the expertise to handle them, can be reused in Step (1).
- The behavior models are analyzed in their intended inference direction, i. e., no special diagnosis model needs to be developed, and no inverse simulation problem needs to be solved.
- The classifier learned in Steps (3) and (4) integrates seamlessly with existing diagnosis approaches in the automotive domain, such as fault trees.
- The compiled diagnosis model has a very small computational footprint. Other state-of-the-art diagnosis approaches require the execution and analysis of a behavior model at runtime.

3.1 Formal Framework

This subsection formally introduces the notion of a behavior model, sometimes also called a plant or controller model. The definition will help up us to define both the different diagnosis tasks and the compilation principle in mathematical terms.

A car is a complex system that consists of several connected (sub)systems, such as engine, drivetrain, engine control, etc. For simulation purposes, we regard each interesting subsystem as being adequately represented by a behavior model $\mathcal{M}$. Depending on the associated subsystem and the simulation purpose, $\mathcal{M}$ may be input-free or input-dependent, or memoryless or dynamic. The most important distinction relates to the time base of dynamic models, which can be continuous time, discrete time, or discrete event.

Definition 1 (Behavior Model [25]) *A behavior model $\mathcal{M}$ is a tuple $\langle F_U, F_Z, F_Y, \mathcal{V}, \Delta, \Lambda\rangle$. $F_U \cap F_Z = \emptyset$, whose elements are defined as follows.*

- *F_U, F_Z, and F_Y are sets of input variables, constraint variables, and output variables.*
- *For each variable $v \in F_U, F_Z$, and F_Y there is an arbitrary, possibly infinite set U_v, Z_v, and Y_v respectively, called the domain of v.*
  For each $v \in F_U$ there is an additional domain, U_v^T, of partially defined functions in the parameter time, $U_v^T := \{u \mid u : T \to U_v, t \mapsto u(t)\}$. Depending on the model's time base, which may be continuous time, discrete time, or discrete event, T may be an interval from $\mathbf{R}^+$, an interval from $\mathbf{N}$, or a linearly ordered finite set.
  $\mathcal{V}$ comprises the domains of all variables. As a matter of convenience, the Cartesian products of the domains of the variables in F_U, F_Z, F_Y are designated with $\mathcal{U}, \mathcal{U}^T, \mathcal{Z}$, and $\mathcal{Y}$. E.g., $\mathcal{Y} := Y_{v_1} \times Y_{v_2} \times \ldots \times Y_{v_{|F_Y|}}$, $v_i \in F_Y$.
- *Δ is a function, called the global state prescription function. Δ declares a set of state variables, $F_X \subseteq F_Z$, and a state space, $\mathcal{X}$, which is the projection of $\mathcal{Z}$ with respect to F_X. Given a state vector $\mathbf{x} \in \mathcal{X}$, a vector of input functions $\mathbf{u}(t) \in \mathcal{U}^T$, and some point in time $t \in T$, Δ determines a constraint vector $\mathbf{z} \in \mathcal{Z}$ including a new state, say, $\Delta : \mathcal{X} \times \mathcal{U}^T \times T \to \mathcal{Z}$.*

- Λ *is a function, called the output function. The output function might be a function of constraint variables and input or only a function of constraint variables. Given a constraint vector* $\mathbf{z} \in \mathcal{Z}$ *and an input vector* $\mathbf{u} \in \mathcal{U}$, Λ *determines an output vector* $\mathbf{y} \in \mathcal{Y}$, *say,* $\Lambda : \mathcal{Z} \times \mathcal{U} \to \mathcal{Y}$ *or* $\Lambda : \mathcal{Z} \to \mathcal{Y}$.

A model of even a small part of a car is likely to combine behavior models of different types, say, different time bases; such a model is called "hybrid".

Let $\boldsymbol{\mathcal{M}} = \{\mathcal{M}_1, \ldots, \mathcal{M}_k\}$ be a possibly hybrid car model, then $\mathcal{M}_i$ and $\mathcal{M}_j$ are coupled if they share output variables and input variables, i. e., if $F_{Y_i} \cap F_{U_j} \neq \emptyset$, with $F_{Y_i} \in \mathcal{M}_i$, $F_{U_j} \in \mathcal{M}_j$, $i \neq j$. For example, $\mathcal{M}_i$ may be a continuous time model of the engine, and $\mathcal{M}_j$ may be a discrete time model of the engine control. In the car this coupling is realized by a sensor whose signal is passed to an ECU where the physical quantity is represented in the software abstraction layer as an input variable of the engine control software.

The above definition can be regarded as a specification of a correct behavior model of a system. For our diagnosis approach we also need models of fault behavior in the sense of the GDE+ [7, 28, 29]. A fault behavior model is an extension of the above definition: There is an additional set of state variables, F_D, along with respective domains $\mathcal{D}$, and a state prescription function Δ'. F_D defines fault states of the components, such as those mentioned in Section 1. Thus, the domain of Δ' is $\mathcal{D} \times \mathcal{X} \times \mathcal{U}^T \times T$.

Let $\sigma(\boldsymbol{\mathcal{M}}, \mathbf{u}(t))$ and $\sigma(\boldsymbol{\mathcal{M}}'_i, \mathbf{u}(t))$ designate simulation results of the faultless model $\boldsymbol{\mathcal{M}}$ and some fault model $\boldsymbol{\mathcal{M}}'_i$ for a given vector of input functions $\mathbf{u}(t)$. By applying a model- and quantity-specific difference operator, $\ominus$, between the simulated faultless values and the related faulty values a database $\mathcal{C}_\Delta$ with symptom vectors can be compiled:

$$\sigma(\boldsymbol{\mathcal{M}}, \mathbf{u}(t)) \ominus \sigma(\boldsymbol{\mathcal{M}}'_i, \mathbf{u}(t)) = \mathcal{C}_\Delta, \qquad i = 1, \ldots, k \tag{1}$$

In a second step, based on filtering, machine learning, and data mining methods, a classifier can be constructed that maps from symptoms onto faults:

$$\mathcal{C}_\Delta \longrightarrow F_D \tag{2}$$

An alternative to the previous steps is to apply a learning approach directly to the simulation results:

$$\mathcal{C} \longrightarrow F_D \quad \text{with} \tag{3}$$

$$\mathcal{C} = (\sigma(\boldsymbol{\mathcal{M}}, \mathbf{u}(t)) \cup \sigma(\boldsymbol{\mathcal{M}}'_i, \mathbf{u}(t))), \qquad i = 1, \ldots, k$$

Note that this alternative burdens the learning approach with the task of both discovering and implicitly computing the $\ominus$-operation, and hence it is less powerful. On the other hand, it allows the learned diagnostic associations to be used directly without having to simulate $\boldsymbol{\mathcal{M}}$ and apply the $\ominus$-operator.

3.2 Case Study

To demonstrate the applicability of the approach, a case study was conducted by the authors: the diagnosis of wrong sensor readings in an engine ECU. Simulating the behavior of the engine and its ECU requires advanced models that define realistic physical behavior. Moreover, the engine and its ECU do not operate autonomously, but interact with several other components.

In this case study we were able to use the "Gasoline Engine Simulation Package" from dSPACE's Automotive Simulation Models, ASM.[4] This is a complete model of a 6-cylinder 2.9l gasoline engine and integrates models for the engine ECU, the drivetrain, vehicle dynamics, and the environment. The vehicle dynamics model takes external forces into account, such as air resistance and braking; the environment model considers road conditions and driver interactions. The model is implemented in Matlab®/Simulink®, and Figure 4 shows an abstracted graphical overview. All the models together form the hybrid car model $\boldsymbol{\mathcal{M}}$.

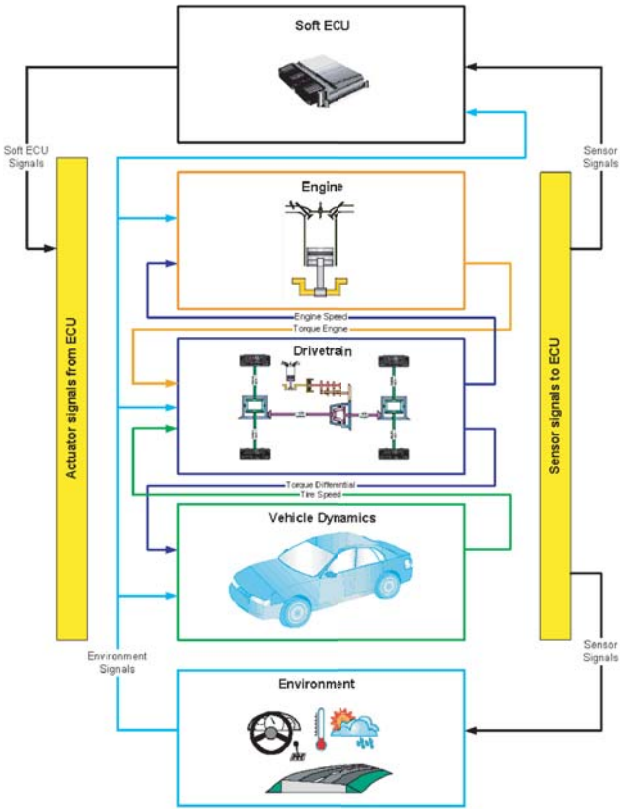

Figure 4. Overview of the hybrid car model, $\boldsymbol{\mathcal{M}}$, with the submodels for the ECU, the engine, the drivetrain, the vehicle dynamics, and the environment.

3.2.1 Behavior Model

The engine model has an accelerator pedal sensor that relays the signal to the ECU, where the data is used to set actuators such as the throttle position in the engine (cf. Figure 5). In the following description the engine model is denoted $\mathcal{M}_1$, the model for the engine ECU is $\mathcal{M}_2$, and the model for the sensor and actuator software and hardware is $\mathcal{M}_3$. Accordingly, the faulty variant of $\mathcal{M}_3$ is denoted as $\mathcal{M}'_3$.

$\mathcal{M}_3$ does nothing but relay the signals between $\mathcal{M}_1$ and $\mathcal{M}_2$. Its faulty counterpart, $\mathcal{M}'_3$, manipulates either (*i*) the sensor data so that the engine *control* receives disturbed values or (*ii*) the actuator data so that the engine receives disturbed values. Figure 6 shows the interplay between these models.

[4] See www.dspace.de/ww/en/pub/home/products/sw/automotive_simulation_models.cfm for more detailed information.

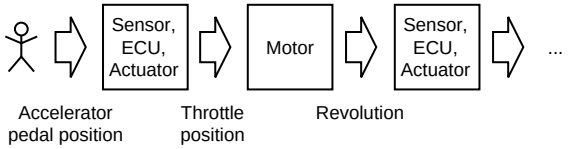

Figure 5. Signal flow in the case study; the starting point is the user input for the accelerator pedal.

The elements of Definition 1 instantiate in our scenario as follows:

- $F_{Y_{\mathcal{M}_1}}$ contains variables such as mean effective engine torque.
- $F_{U_{\mathcal{M}_1}}$ contains variables such as crank angle per cylinder and throttle angle. Note that $F_{U_{\mathcal{M}_1}} = F_{Y_{\mathcal{M}_2}}$.
- Examples of input variables of $F_{U_{\mathcal{M}_2}}$ are accelerator pedal position, ignition signal, and engine speed.
- $F_{U_{\mathcal{M}_3}}$ and $F_{Y_{\mathcal{M}_3}}$ do not contain any variables that are not used in $\mathcal{M}_1$ and $\mathcal{M}_2$. Hence $F_{U_{\mathcal{M}_3}} = (F_{U_{\mathcal{M}_2}} \cup F_{U_{\mathcal{M}_1}}) = F_{Y_{\mathcal{M}_3}}$. This is because $\mathcal{M}_3$ is used only to manipulate input and output variables of the engine control.
- $F_{X_{\mathcal{M}_2}}$ contains the throttle position among others.
- $F_{X_{\mathcal{M}_1}}$ contains variables for the manifold temperature and the air flow through the throttle.
- $\Delta_{\mathcal{M}_1}$ and $\Delta_{\mathcal{M}_2}$ are formulated as differential equation systems and define the relations between the derivatives of the state variables and the input variables. Since (the correctly working) $\mathcal{M}_3$ functions as a short circuit, $\Delta_{\mathcal{M}_3}$ is the identity function. $\Delta_{\mathcal{M}'_3}$, however, manipulates one or more of the input values in order to simulate offsets, interruptions, or white noise on the signal of the accelerator pedal.
- Λ maps the values from $\mathcal{Z}$ to $\mathcal{Y}$. Thus, $\Lambda_{\mathcal{M}_i}, i \in \{1, 2, 3\}$, is the identity function.

3.2.2 *Considered Fault Types*

In this case study the following typical faults occurring in sensors and actuators are modeled:

1. The signal is reduced to 90% of its correct value (-10% offset).
2. The signal is 110% of the correct value (+10% offset).
3. The signal is superimposed with noise.
4. The signal is dropped out.

3.2.3 *Simulation of $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{M}}'$*

$\boldsymbol{\mathcal{M}}$ consists of $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, and the models for the drivetrain, the vehicle dynamics, and the environment; it contains more than 100 states. $\boldsymbol{\mathcal{M}}' = (\boldsymbol{\mathcal{M}} \setminus \mathcal{M}_3) \cup \mathcal{M}'_3$. The main input variables are driver actions as observed by the respective sensors and include the positions of the brake pedal, the clutch pedal, the accelerator pedal, and the engaged gear.

For simulation purposes p different vectors of input functions $\mathbf{u}_1(t)$, ..., $\mathbf{u}_p(t) \in \mathcal{U}^T$ were defined. These vectors are called scenarios and represent a specific driving behavior for a defined time period. Thus, $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\mathcal{M}}'$ can be executed in different driving situations. Since it is unrealistic for all faults to occur at the same time, faults are inserted into the model at several random points in time. Each simulation run is characterized by the scenario, by the fault type (including the faultless scenario), by the faulty component (e. g. the actuator for the throttle), and by the point in time the fault occurs.

Various scenarios were simulated using the fault types of the accelerator pedal position sensor. The values of all the relevant signals,

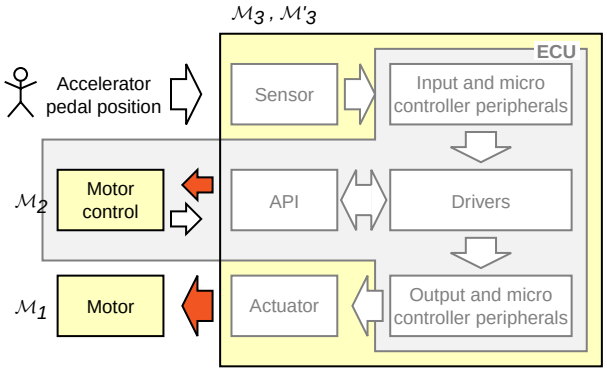

Figure 6. Interplay of three models: $\mathcal{M}_1$ (engine), $\mathcal{M}_2$ (engine control), and $\mathcal{M}_3$ (connecting subsystems between user input, engine, and engine control). In the faultless behavior situation, $\mathcal{M}_3$ functions as a short circuit directly connecting both the user input with $\mathcal{M}_2$ and $\mathcal{M}_1$ with $\mathcal{M}_2$; in a fault situation the fault model $\mathcal{M}'_3$ inserts particular signal disturbances (see dark arrows).

i. e., the signals that may have been influenced by the faulty accelerator pedal sensor, were logged for every simulation run and written to a simulation database, $\mathcal{C}$. Information on each fault type along with the time when the fault occurred was also stored in the database.

Figure 7 and Figure 8 show an extract of the results of a faultless and a faulty behavior simulation. Comparing the two figures reveals that the fault in the accelerator pedal sensor influences other values, such as the throttle position value and the number of revolutions.

3.2.4 *Learning the Diagnosis Function*

After the system is simulated in different scenarios with different faults (including the faultless scenario), machine learning is applied in order to learn the mapping specified in Equation 3. The basic idea is to use the data from the faultless simulation and the data from one fault simulation to detect discriminating features. The knowledge gained can be used as a basis for constructing a classifier which decides whether the accelerator pedal sensor was faulty or not. The hope is obviously that this classifier will have a strong inductive bias, say, that it can also be applied to new measured data from a real vehicle. The task is therefore to find a machine learning approach that will learn the classification function as well as possible but without too much overfitting.

The case study investigated two learning algorithms:[5]

- *Linear Regression.* This common method employs least square estimation to determine parameters $a_1, ..., a_n, a_i \in \mathbf{R}$, for the following diagnosis function template:

$$d_i = a_1 \cdot v_1 + ... + a_n \cdot v_n \quad \text{with}$$

$$d_i = \begin{cases} 0 & \text{if fault } i \text{ occurs} \\ 1 & \text{else} \end{cases}$$

  $v_1, \ldots, v_n$ are the variables measured during the simulation. Details can be found in [10, 32].
- *Decision Trees.* A decision tree uses a series of decisions to reach a classification. Learning such a tree is done by recursive partitioning the input space using a given optimization criterion. Details can be found in [2, 21].

[5] The R programming environment was used (cf. http://www.r-project.org).

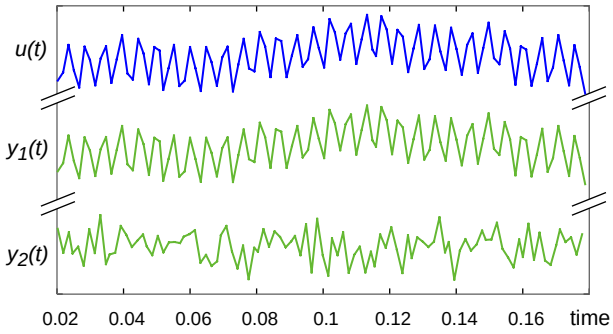

Figure 7. Results of the simulation of the faultless model $\mathcal{M}$. $u_1(t)$, $y_1(t)$, and $y_2(t)$ designate the accelerator pedal position, the throttle position, and the number of revolutions respectively.

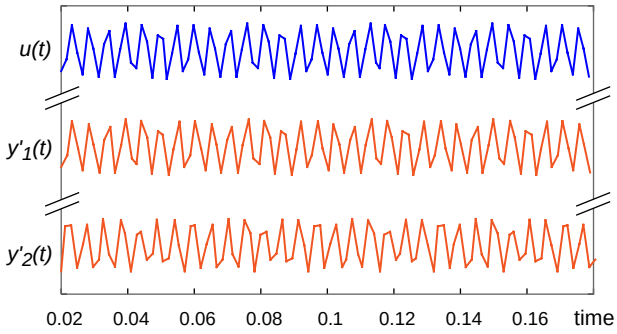

Figure 8. Results of the simulation of the faulty model $\mathcal{M}'$ that corresponds to the faultless simulation in Figure 7. The introduced fault pretends noise on the pedal position signal.

Note that a diagnosis function learned for a specific fault type may not adequately deal with data measured in a situation where another fault type has occurred. To differentiate between fault types, the data for learning a diagnosis function of a specific fault type was extended by the data of the other three fault types. This new data was classified as faultless scenarios. The disadvantage of this procedure is that the number of fault cases is no longer equivalent to the number of faultless cases, making it more difficult to assess the learning result.

3.2.5 Experimental Results

Though only standard algorithms for the learning of the diagnosis function were used and though no preprocessing steps were implemented, the classification results are convincing. Table 1 shows the error rates of the learned diagnosis functions for the four fault types "-10% offset", "+10% offset", "noisy signal", and "dropped out signal". Five runs were done for each of the fault types; their average is presented in the table.

The error rate is the percentage of cases diagnosed incorrectly by the algorithm. The gray rows show the error rates that were achieved by runs with the training data. The rows below show the error rates that were achieved with input data the algorithm did not use to learn the diagnosis function; they demonstrate the generalizability of the classifiers.

-10% offset	+10% offset	noisy signal	dropped out signal
Linear regression			
15.04%	16.13%	3.84%	0.04%
14.85%	16.53%	3.55%	0.11%
Decision tree			
0.88%	0.32%	0.33%	0.10%
1.13%	0.46%	0.31%	0.17%

Table 1. Error rates of the linear regression classifier and the decision tree classifier for the four fault types.

Observe that the decision tree algorithm performs much better than linear regression. Only when the signal is dropped out is linear regression able to perform a little better than the decision tree. Also note that there is nearly no difference between the results in the first row and the results in the second row. If the diagnosis function is also used to differentiate between fault types, classification performance is not as good as before; nevertheless, the results still show that it is possible to achieve satisfying results.

4 CONCLUSION AND CURRENT WORK

The paper introduced the model compilation paradigm for diagnosis on complex technical systems. Model compilation is fairly involved and combines modern simulation technology with methods from data mining and learning theory: The models $\mathcal{M}$ and $\mathcal{M}'$ of a system under study are simulated with respect to expected inputs along with possible faults, and a compiled diagnosis model is distilled from the huge set of generated data. At heart, the compiled model is a classifier, which is not only able to detect the simulated faults, but which will also generalize with respect to unseen situations.

Model compilation is particularly attractive since the original simulation models $\mathcal{M}$ from the application domain can be utilized. In fact, in the case study from the automotive domain that was presented, $\mathcal{M}$ comprises a vehicle's plant and controller models; it is hybrid and contains more than 100 states. The outlined diagnosis situations address realistic signal faults that occur between the environment and the vehicle electronics. Two learning approaches, linear regression and decision trees, were applied, leading to acceptable (85%) and excellent (99%) fault detection rates. These results show that this approach is worth pursuing further.

Our current and future work focuses on four aspects:

1. More complex and different fault scenarios, which also include software faults in ECUs.
2. The analysis of multiple faults.
3. The application of stronger data filtering techniques during the data mining step, such as vector quantization and cluster analysis.
4. Refined methods to differentiate between a large number of faults. Note that the choice and the adaptation of the machine learning algorithms are keys to the success of the model compilation paradigm, and that association rules or Bayesian networks have the potential to outperform decision trees on large data sets.

References

[1] Sherif Abdelwahed, Gabor Karsai, and Gautam Biswas, 'A Consistency-based Robust Diagnosis Approach for Temporal Causal Systems', in *16th International Workshop on Principles of Diagnosis, DX-05*, pp. 73–79, Monterey, California, USA, (June 2005).

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth, 1984.

[3] F. Cascio, L. Console, M. Guagliumi, M. Osella, A. Panati, S. Sottano, and D. Dupre. On-board diagnosis of automotive systems: from dynamic qualitative diagnosis to decision trees, 1999.

[4] Luca Console, Gerhard Friedrich, and Daniele Theseider Dupré, 'Model-Based Diagnosis Meets Error Diagnosis in Logic Programs', in *IJCAI*, pp. 1494–1501, Chambery, France, (1993).

[5] Adnan Darwich, 'On Compiling System Descriptions into Diagnostic Rules', in *Proceedings of the 10th International Workshop on Principles of Diagnosis*, Scotland, (June 1999).

[6] Johan de Kleer and Brian C. Williams, 'Diagnosing Multiple Faults', *Artificial Intelligence*, **32**(1), 97–130, (1987).

[7] Johan de Kleer and Brian C. Williams, 'Diagnosis with Behavioral Modes', in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI 89)*, pp. 1324–1330, Detroit, Michigan, (1989).

[8] Russell Greiner, Barbara A. Smith, and Ralph W. Wilkerson, 'A Correction to the Algorithm in Reiter's Theory of Diagnosis', *Artificial Intelligence*, **41**(1), 79–88, (1989).

[9] Irene Grosclaude, 'Model-based monitoring of component-based software systems', in *15th International Workshop on Principles of Diagnosis, DX-04*, Carcassonne, France, (June 2004).

[10] Jens Hartung, *Statistik*, Oldenbourg, 1999.

[11] Uwe Husemeyer, *Heuristische Diagnose mit Assoziationsregeln*, Dissertation, University of Paderborn, Department of Mathematics and Computer Science, 2001.

[12] Meir Kalech and Gal A. Kaminka, 'Towards Model-Based Diagnosis of Coordination Failures', in *16th International Workshop on Principles of Diagnosis, DX-05*, pp. 37–42, Monterey, California, USA, (June 2005).

[13] Frank Kimmich, Anselm Schwarte, and Rolf Isermann, 'Fault detection for modern Diesel engines using signal- and process model-based methods', *Control Engineering Practice*, **13**(2), 189–203, (2005).

[14] Daniel Köb, Rong Chen, and Franz Wotawa, 'Abstract model refinement for model-based program debugging', in *16th International Workshop on Principles of Diagnosis, DX-05*, pp. 7–12, Monterey, California, USA, (June 2005).

[15] James Kurien, Xenofon Koutsoukos, and Feng Zhao, 'Distributed Diagnosis of Networked, Embedded Systems', in *13th International Workshop on Principles of Diagnosis, DX-02*, pp. 179–188, Semmering, Austria, (May 2002).

[16] Gianfranco Lamperti and Marina Zanella, *Diagnosis of Active Systems*, Kluwer Academic Publishers, Hingham, MA, USA, 2003.

[17] Wolfgang Mayer and Markus Stumptner, 'Approximate Modeling for Debugging of Program Loops', in *15th International Workshop on Principles of Diagnosis, DX-04*, pp. 87–92, Carcassonne, France, (June 2004).

[18] Tsoline Mikaelian, Brian C. Williams, and Martin Sachenbacher, 'Diagnosing Complex Systems with Software-Extended Behavior using Constraint Optimization', in *16th International Workshop on Principles of Diagnosis, DX-05*, pp. 19–24, Monterey, California, USA, (June 2005).

[19] Mattias Nyberg, 'Model-based diagnosis of an automotive engine using several types of fault models', *IEEE Transaction on Control Systems Technology*, **10**(5), 679–689, (September 2002).

[20] Raymond Reiter, 'A Theory of Diagnosis from First Principles', *Artificial Intelligence*, **32**(1), 57–95, (1987).

[21] Brian D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.

[22] Nico Roos, Annette ten Teije, and Cees Witteveen, 'A Protocol for Multi-Agent Diagnosis with spatially distributed Knowledge', in *Autonomus Agents and Multi Agent Systems, AAMAS-2003*, pp. 655–661, (July 2003).

[23] Indranil Roychoudhury, Gautam Biswas, Xenofon Koutsoukos, and Sherif Abdelwahed, 'Designing Distributed Diagnosers for Complex Physical Systems', in *16th International Workshop on Principles of Diagnosis, DX-05*, pp. 31–36, Monterey, California, USA, (June 2005).

[24] Werner Seibold and Bernhard Höfig, 'Sichere Fehlerdiagnose in der Automobilwartung mit RODON 3', *Automotive electronics*, 70–75, (2004).

[25] Benno Stein, *Model Construction in Analysis and Synthesis Tasks*, Habilitation, Department of Computer Science, University of Paderborn, Germany, June 2001.

[26] Benno Stein, 'Model Compilation and Diagnosability of Technical Systems', in *Proceedings of the 3rd IASTED International Conference on Artificial Intelligence and Applications (AIA 03), Benalmdena, Spain*, ed., M. H. Hanza, pp. 191–197. ACTA Press, (September 2003).

[27] Gerald Steinbauer and Franz Wotawa, 'Detecting and locating faults in the control software of autonomous mobile robots', in *IJCAI*, pp. 1742–1743, (2005).

[28] Peter Struss, 'Model-Based Diagnosis—Progress and Problems', in *Proceedings of the International GI-Convention*, volume 3, pp. 320–331, (October 1989).

[29] Peter Struss and Oskar Dressler, '"Physical Negation"—Integrating Fault Models into the General Diagnostic Engine', in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 89)*, volume 2, pp. 1318–1323, (1989).

[30] Peter Struss and Chris Price, 'Model-based systems in the automotive industry', *AI Magazine*, **24**(4), 17–34, (2004).

[31] R. Su, W. Wonham, J. Kurien, and X. Koutsoukos, 'Distributed Diagnosis for qualitative Systems', in *Proceedings of the 6th International Workshop on Discrete Event Systems, WODES-02)*, eds., M. Silva, A. Giua, and J.M. Colom, pp. 169–174, Zaragoza, Spain, (October 2002).

[32] T. Wonnacott and R. Wonnacott, *Regression: A second course in statistics*, John Wiley & Sons, New York, Chichester/Brisbane/Toronto, 1981.