Role-Based Diagnosis for Distributed Vehicle Functions

Jens Kohl[1], **Andreas Bauer**[2]

[1] *BMW Group and Technische Universität München*
Jens.Kohl@bmw.com
[2] *National ICT Australia (NICTA) and The Australian National University*
Andreas.Bauer@nicta.com.au

ABSTRACT

With distributed functions taking over more and more safety-relevant functions in modern cars, their possible faulty behaviour has to be detected and dangerous effects to be prevented or mitigated. Additionally, information about the fault's root cause has to be provided to support repairs in the garage. These are the central tasks of automotive diagnosis, which is arguably a prime application for the methods and tools developed by the diagnosis community at large. However, our experiences have shown that the constraints imposed by this domain, which has a great need for efficient and accurate diagnoses, make it very difficult to use some of the existing methods out of the box. The main contribution of this paper is therefore to introduce a methodology centred around the different stakeholders of automotive diagnosis, in order to facilitate the employment of model-based diagnosis approaches for the diagnosis of vehicle functions as they are part of most modern cars. Besides, we provide an in-depth discussion of some of the challenges imposed by automotive diagnosis, and relate these to existing diagnosis methods where feasible, thereby also further motivating our methodology.

1 INTRODUCTION

Automotive software functions contribute a great part to a modern car's innovations. In the last years automotive software development has shifted from a component-centric view towards the realisation of features as distributed functions. With distributed functions taking over more and more safety-relevant functions, their possible faulty behaviour has to be detected and dangerous effects to be prevented or mitigated. Additionally, information about the fault's root cause has to be provided to support repairs in a garage. These are the tasks of automotive diagnosis. The distributed functionalities' interconnections contribute to an increasing complexity of the car's system architecture which, as we argue in this article, poses a huge challenge for automotive diagnosis. The importance of diagnosis is increased by warranty regulations of currently up to four years which make it a significant economic factor in a car's life-cycle. Hence controlling the distributed functions with diagnosis will be a key competence for all automotive OEMs (i.e., "original equipment manufacturers" as vehicle manufacturers refer to themselves).

There are at least two different topologies used for the design of a car's system architecture: On a physical level, functionalities are distributed in terms of *electronic control units* (ECUs), which are basically embedded systems that control the electric systems in a car, and are themselves interconnected via up to five different bus systems. On a logical level, most functionalities are realised by *software components*, which are connected mainly by the exchange of messages.

Consequently, a functionality which is perceived by the driver as a single service (e.g., automatic cruise control (Ioannou, P. and Chien, C., 1993; Prestl, W. *et al.*, 2000)) is in fact, realised by a large number of software components that operate concurrently within the network of physically distributed ECUs. The heterogeneity of these functionalities and the way they are realised in a modern car make it difficult to come up with a single, holistic diagnosis model that captures, from a diagnosis point of view, all the relevant components, their behaviour, timings, and interactions. Therefore, arguably, a diagnosis approach is needed which reflects the different artefacts that matter in diagnosing faults of such systems, and which is not bound to a single type of system or fault model that captures only a fraction of the relevant information.

While there are many applications for, more or less, classical diagnosis approaches in today's cars, such as the diagnosis of an engine management system, which is mostly about the timely observation, analysis, and control of the physical combustion process from a single spot and the comparison of residual equations modelling these physical processes, cf. (Nyberg, M. *et al.*, 2001; Isermann, R., 2004), there is currently no "one-size-fits-all approach" for the on-*and* off-board-diagnosis for all the logically as well as physically distributed system built in a car, be it software, electro-mechanical or purely mechanical components. One would expect that, due to the fact that these systems can be described in terms of their components, be it software or hardware, that these are prime applications for the methods and tools developed in the area of model-based diagnosis, where the aim is to detect and isolate faulty components, such as gates in an integrated circuit, cf. (Reiter, R., 1987; Kleer, de J. and Williams, B. C., 1987; Sampath, M. *et*

al., 1995). But our experience in trying to apply some of these methods has shown that many of the real-world constraints imposed by our domain can only be poorly, if at all, reflected in the various approaches to model-based diagnosis currently found in the literature. What is currently missing, as we argued in this article, is a *methodology* that helps embed these approaches in a domain like ours.

In this paper we therefore propose a role-based methodology for the diagnosis of distributed automotive functions. Central to this methodology is a layered system abstraction based on which diagnosis is performed. The layers, in turn, are defined wrt. the different roles involved in the automotive diagnosis process, i.e., customer view, garage view, technical view. Fig. 1 shows the main dependencies between these layers from an automotive diagnosis point of view. Essentially, the layers aim at reducing the complexity of the diagnostic task, since only the relevant entities are taken into account from the respective role's point of view; that is, following this methodology one does not need to create a single, holistic diagnostic model of a vehicle function taking all the different aspects, which are currently abstracted from in the layers, into account.

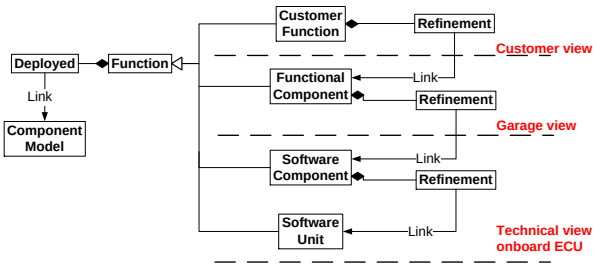

Figure 1: Role-based diagnosis layers.

What is more, we take into account the *online diagnosis* of automotive systems, i.e., on a detailed technical level that is required to perform the diagnostic task at runtime, and *off-line diagnosis*, i.e., in the garage, where the main aim is to replace physical components that may be responsible for observed faults. Reflecting these different views is necessary to cover all the possibly occurring faults in our domain.

Basically, for the detection and reasoning of the root cause, we propose model-based diagnosis as originally proposed in (Reiter, R., 1987; Kleer, de J. and Williams, B. C., 1987), but in a more general setting as presented, e.g., in (Bauer, 2007; Tripakis, S., 2009). In a nutshell, this allows us to reason about discrepancies between observed and specified behaviour of components, and is thus capable to diagnose components with almost a black-box view onto each other. And since these approaches do not impose any restrictions on the granularity of components, this allows us to tailor the diagnostic models to the different layers of abstraction introduced above.

Another aspect of our methodology is that we also take a well-known and widely used quality assurance method into account, namely *Failure Mode and Effects Analysis* (FMEA, (Stamatis, D. H., 2003)), which helps describe faults that are anticipated by the OEM. Essentially it does that by listing potential faults and supporting a structured analysis of their effects on the system or its operational environment. Moreover, a complete FMEA also includes information about some of the faults that can be detected and even prevented. Based on this information it is then possible to classify faults and to prioritise them based upon their severity, i.e., according to the effects on the system and environment. Note that we will make use of the terminology for errors, faults and failures as defined by (Avižienis, A. *et al.*, 2004). They define a *system failure* as an event that occurs when the delivered service of a system deviates from correct service. An *error* is part of the system state which is liable to lead to subsequent failure. The adjudged or hypothesized cause of an error is called a *fault*. The causal chain between these definitions is that a fault causes an error and an error can lead to a failure. For instance, a mechanical fault of the window motor leads to a failure of the window moving functions.

Outline: In the next section, we detail on the problems we faced with some of the existing diagnosis methods, when used in our domain, and also motivate the methodology presented in this paper. Sec. 3, introduces a case study to which we have applied our methodology, namely the controller of a car's window (sometimes called power window). Note that we haven chosen this case study, not because it represents an unsolved diagnosis problem in the automotive domain (cf. (Struss, 2006) for a similar case study), but because it is a distributed and easily intelligible component that can be well explained within the limits of this paper. Sec. 4 discusses how to obtain fault models on the different levels of abstraction from the FMEA, whereas Sec. 5 shows how this is performed on our case study. Finally, Sec. 6 concludes.

2 THE NEED FOR A NEW METHODOLOGY

In this section, we briefly discuss why the application of well-known approaches to diagnosis found in the literature cannot compensate for a lack of methodology, when used in our domain. For example, *expert-based diagnosis systems* as defined first in (Buchanan, B. and Shortliffe, E., 1984), are rather easy to understand and straightforward to create, which is the reason why they are widely used in this area. However, they are also difficult to maintain as the system and its architecture evolve over time. In the automotive domain, expert-based diagnosis is used primarily for software-intensive, non-safety-relevant components such as various infotainment devices.

Diagnosis of discrete event systems (DES) as originally introduced in (Sampath, M. *et al.*, 1995), is difficult to apply in practice when one assumes that for each component to be diagnosed in a distributed system one has to create a dedicated behavioural DES model. Diagnosis then would consist of evaluating these components in parallel which additionally bears a high level of complexity. Moreover, it is not always possible to create DES models in the first instance since some components need to be handled as black-boxes, meaning that they have been supplied by a third party, according to a detailed specification by

the OEM, and behavioural models are therefore difficult to extract.

Model-based diagnosis (MBD) as introduced in (Reiter, R., 1987; Kleer, de J. and Williams, B. C., 1987) is often used for reasoning over electrical components especially in safety-relevant domains (Isermann, R., 2004). The focus is on isolating faulty components by observing their input and outputs and relating the observed behaviour with a specified behaviour, which seems to be a good match with the communicating automotive components. However, MBD faces similar problems in the automotive domain as the use of DES diagnosis since both method's system models are tedious, if not impossible, to create for engineers. Furthermore, although the behaviour of these systems or components can be modelled on an abstract level on behalf of the OEM, it would be difficult to include these systems in a holistic diagnosis model on the detailed and complex level of software components (i.e., source code and operating system tasks). Additionally, the size of a diagnosis system model is exponentially increased by the diversity of a single car's product line and different legal requirements that impact on the car's architecture, cf. (Broy, M., 2006; Pretschner, A. *et al.*, 2007), and thus the model. It is therefore important that diagnosis on different levels of abstraction is supported, depending on the specific stakeholder of the diagnostic process (i.e., customer, garage, technical view), and to be able to cope with "black-box behaviour", where necessary or useful to avoid a too high complexity. Moreover, it is important to leverage the effort in creating models that are useful for the diagnostic process, and that can easily evolve with the vehicle's main architecture, which is not the case if we apply any of the above approaches without a methodology that would support us in the decision making on which level of abstraction is the right one, which components can be modelled as black-box, etc.

The wish for an extensive diagnosis system model can lead to a practically unmanageable complexity of diagnosis systems. Some modern components such as driving assistant or infotainment systems tend to have several hundred monitors, each monitoring system properties that indicate the correct functioning of the system, and storing up to 400 *diagnostic trouble codes* (DTC), which support the garages in further diagnosis and subsequent repairs. Despite the high number of DTCs, which indicate the cause of observed faults, the resulting task of fault localisation is challenging for the garages and thus results in high costs. This is especially severe when repeated repairs are undertaken for the same faults and the time spent in trying to find the root causes. Nevertheless, weighing between an extensive or reduced diagnosis system model, we argue that the amount of monitors and DTC can be reduced without deteriorating the repair process. We are convinced that this reduction even makes the diagnosis and its results more understandable for garages and thus can even help cut costs. Essentially, we propose that for expensive components a more detailed diagnosis can be beneficial, but for other components, perhaps ones used for a long time, a reduced diagnosis and model bears no disadvantages from the OEM's point of view. Clearly, such considerations are not solely technically motivated, but methodologically and aim at making the use of existing diagnosis methods easier in our domain.

A reduced diagnostic model further benefits the *reuse of diagnosis* which is another important aspect in our domain. Economic constraints such as wish to shorten the car's or component's time to market and the creation of platform strategies to save development costs force OEMs and suppliers to reuse software. Efforts in software product line methodology, cf. (Thiel, S. and Hein, A., 2002), have been increased in the last years for automotive functions, yet not for diagnosis. This is the more important since code for performing diagnosis contributes a substantial part to overall component's source code. The reuse of diagnosis code, however, is currently limited by the lack of a holistic diagnostic approach or process that would cover all involved roles within a diagnosis incident. Previous attempts such as (Picardi, C. *et al.*, 2002) made an important step in this direction, but, for example, did not include customers and the feedback from the garage in the process. The following scenario illustrates how this can be important though: For software-intensive functions, there exist failures that can be detected and prevented within the ECU alone, and which are known by the OEM. For example, if the diagnosis detects an overheating of a window motor, the ECU can disable the motor for a specific time to cool down, and thus preventing a permanent failure. Yet the majority of the on-board-detectable failures cannot be prevented by the ECU. Thus the on-board diagnosis has to store symptoms to support the subsequent off-board repair in a garage, e.g., storing a DTC if the on-board diagnosis detects components such as an incorrect attachment of the window Hall-effect sensors. For a more detailed explanation of the Hall-effect sensors refer to sec. 3 or (Ramsden, E., 2006).

However, for mechanical failures such as a defect of the mechanical window button, the on-board-diagnosis cannot provide any support. These failures can only be detected off-board by the garage. However, there are failures which are unknown to OEM and garage prior to their occurrence and for which no direct detection and repair measures exist. Therefore feedback data from the garage and the customer is necessary such that OEM and supplier can analyse the failure's root cause and work on repair and detection measures to perhaps reflect this in the future. Moreover, it happens that features are misinterpreted by customers as failures. For instance, a customer unfamiliar with the power window controller's safety concept regarding the prevention of jamming obstacle, could report the window's automatic reversing commanded by the ECU in case of a detected jamming as a complaint. Hence, the customer has to be informed of the safety concept. As a result, a holistic diagnosis including on-board and off-board is necessary which additionally includes the customer and diagnosis feedback data to cover all possible automotive failures.

3 A CASE STUDY: WINDOW CONTROLLER

Let us now introduce the case study used in the remainder of the paper. Contrary to many peoples' perceptions, the window controller is not an isolated, monolithic system, but a distributed system communicating

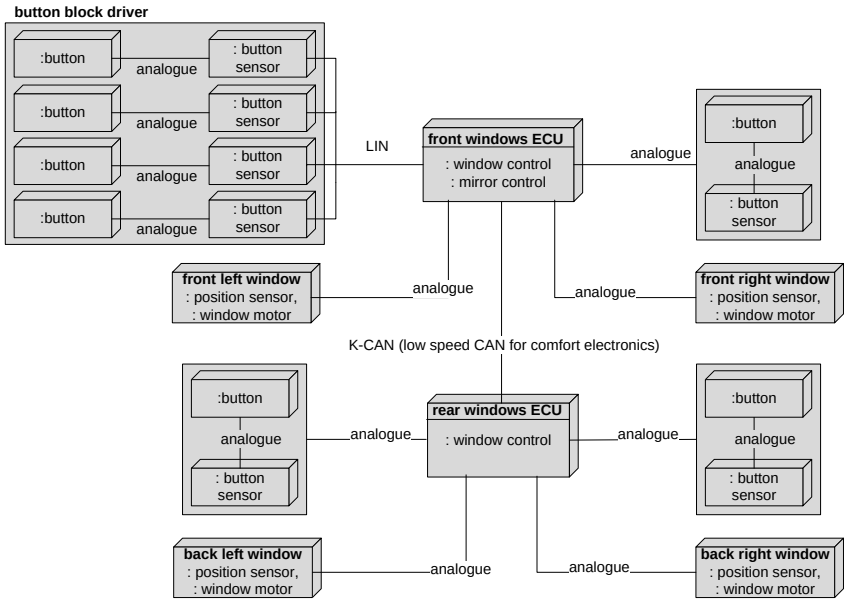

Figure 2: Deployment diagram for a power window controller.

over different kinds of network protocols, and involving various functions realised in software. The window controller's prime tasks are the stepwise and automatic opening and closing of the window. Automatic opening of the window means that, unless the window run is stopped by a window movement in the opposite direction, the window moves until its upper or lower border where it stops. The window controller is designed for use in all cars of a product line and thus includes special functions depending on the particular model chosen. An example for a special function is the short-stroke opening function for frameless doors, such as used in convertibles: in order to enable an opening of a convertible's doors the window has to be lowered as soon as a door opening is detected. Another example is the automatic opening or closing of all windows initiated by a long-time pressing of the car key which is called comfort opening. Since a closing of the window can jam obstacles, the power window controller has to detect a jamming. The jamming, however, can be overruled by the driver in case of an emergency which is called the panic mode. To avoid faults in relation with the window such as overheating of the window motor, the ECU can deactivate the window.

Fig. 2 shows the abstract deployed technical architecture of a car with four windows: As the driver controls all windows there are four different buttons and thus four different signals, transmitted via a single-wire bus protocol called Local Interconnect Network (LIN). LIN therefore replaces the direct analogue wire connections to the ECU. Driver commands for the rear windows are sent via a specific Control Area Network-Bus Protocol (Low CAN) from the front windows ECU to the rear windows ECU. For brevity, we do not give further details on the respective bus systems that are employed by this function. Instead, we refer the interested reader to (Nolte, T. *et al.*, 2005) for a more detailed overview of automotive bus systems.

Fig. 3 shows the different views on the automatic window closing function according to our defined layers. The layers are separated as they represent the perspective of the different roles involved in the diagnosis of the window controller. Observable events for the respective layers are denoted using bold arrows, unobservable events using dashed arrows. They are indexed with a leading *C*, *G*, or *T*, indicating the respective abstraction. Grey rectangles mark the hardware parts (or, replaceable units), similar to the deployment diagram in Fig. 2, as seen from the different layers of abstraction. Finally, the white rectangles are the components performing individual functions. One can easily see that the model's level of detail increases with the further down we go in the abstraction layers, and also more observations become available. However, including the button, the front and rear windows ECU as well as the window itself, only three replaceable units for the garage exist, constituting in a total of 12 monitors in the ECU.

Naturally, the customer has a black-box view on the window which is controlled with the button (C1), resulting in an observable window movement (C2).

For a garage, the window unit consists of three exchangeable units with the button unit, the window itself and an ECU with the software for the power window controlling functions. A button sensor and the mechanical button form the button unit. The ECU is connected over a wire connection with the button unit. This connection transmits an analogue, continuous signal (G1) that can be measured by the garage to analyse if the button sensor transmits signals. The ECU determines in which direction the window shall be moved and if an automatic or manual movement of the window is requested. The command is then transferred over an analogue wire to the window unit consisting of a window motor which moves a wire cable and thus the window's glass. Again, the connection between

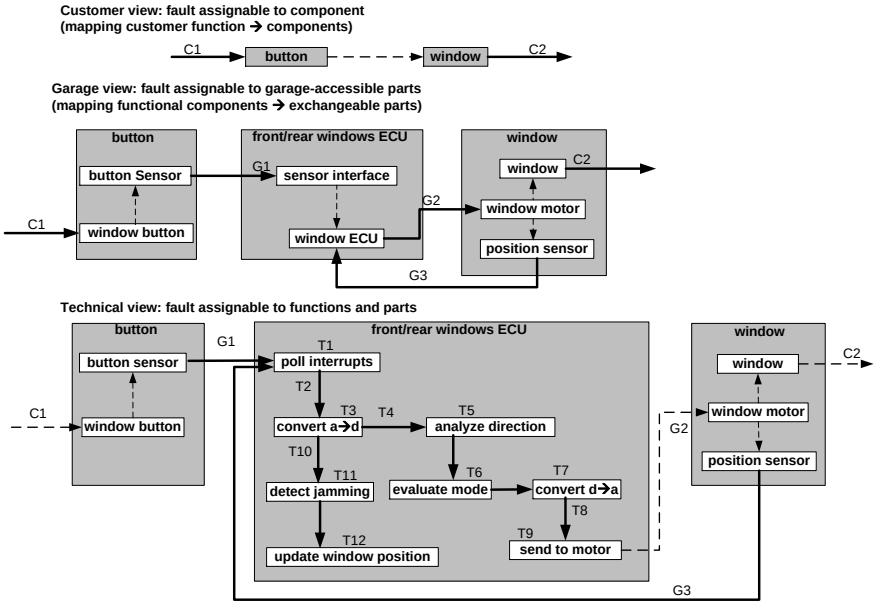

Figure 3: Selection of observable events of a window controller

ECU and window can be measured by the garage (G2). Since the automatic closing function can jam obstacles, jammings have to be detected and countered with an immediate reversing of the window's movement direction. Most cars today detect jammings of the window by analysing two Hall-effect sensors aligned orthogonally to each other next to the window motor with which they move along. After each movement of the window motor, the ECU polls the Hall-effect sensors (G3) to determine the current position and to detect a jamming, if one occurs. The Hall-effect sensors use the physical Hall effect to transduce the rotations of a motor by returning an output voltage. The rotations per minute (rpm) of the window motor can be calculated by measuring the output voltage's pulse amplitude. Aligning two Hall-effect sensors orthogonally to each other additionally enables to detect the motor's rotation direction. Hence, the window position can be updated. For a more detailed overview of Hall-effect sensors refer to (Ramsden, E., 2006). The technical view on the window controller is more detailed than the above. The first function that is performed in the ECU is a polling of input signals from the window button unit (T1). If an analogue signal from a window button is detected, the signal is converted into a discrete value (T2, T3) and given as parameter to the "evaluate direction and moving mode" function which checks the desired direction (T5). Since the signals are continuous, a timer is initialised to evaluate if the customer wanted a stepwise movement of the window or an automatic (T6), separated by pressing the button for a short or a long-time. After movement mode and direction have been determined, the command has to be transferred to the window and therefore first converted to an analogue signal (T7), then sent as a command (T8, T9) over an analogue wire to the motor (G2). Since the jammings have to be detected, the sensor signals are sent over (G3) to the ECU to determine the new window position. The sensor signals are polled (T1), converted to discrete values (T3) and given to the "detect jamming" function (T11). The discretised sensor signals indicate the direction into which the window moved and thus also the window's new position. If after an attempted movement of the window, the window's new position equals the old one, but is neither upper nor lower border of the window frame, a jamming of the window has occurred. In this case the window is reversed (observable in terms of C2), otherwise the window position is updated (T12). Since the automatic closing is safety-relevant, failures of functions using the sensors lead to a deactivation of the window by the window controller.

4 OBTAINING FAULT MODELS

The FMEA is a standard quality method in the automotive domain performed to analyse possible failures of a component and their effects on the system as a whole and its environment. Analysing failures of functional interactions is especially important for distributed functions, but, as mentioned in Sec. 2, problematic due to the restricted view into the components. We tackle this problem by performing an FMEA on the functions and their interactions on different abstraction levels and analyse how the effects of faults can be associated with our abstraction layers. This helps associate the functions, their interactions and faults with the respective layers as given by Fig. 1. For each fault we analyse how it can be detected by our roles starting from customer down to garage and development. For the roles garage and development, we add measures to prevent the fault becoming a failure (pre-emptive diagnosis on the ECU) and repair measures (garage level) for a system remedy. Starting from the customer, we initiate our search for faults that manifest themselves

using equivalent observations. If these faults have their individual repair measures associated to them, involving different technical parts, the faults have to be distinguished by additional observations on deeper technical levels to avoid costly "try-and-error" repairs. We are guided by the realistic assumption that due to the garage's focus on replaceable units, safety-irrelevant faults with equal observations and repairable parts do not need to be discriminable by the diagnosis.

In the next step, we then map some of the elements of the FMEA which in the following are marked in bold to logical variables as well as to our abstraction layers, where they are relevant. Note that this is currently a manual process and, in some sense, constitutes the transition from an informal description to a formal diagnosis model given by logical statements. In our case, however, the mapping from FMEA elements to logical variables and statements is almost canonical, hence, in what follows, we do not detail on this step. From a logical point of view, we can then treat each layer as a set of logical statements involving these variables.

System: name of the component, e.g., window controller
Function: name of the analysed function. Can be a customer function, functional component, software component or software unit from Fig. 1, e.g., automatic closing of window
Failure mode: classification of failure or its exact description, e.g., overheating of window motor.
Fault reason: root cause described with a formula such as $F_i \rightarrow B_i$ with variable F_i assigned true if its related fault is present and B_i assigned true if the behaviour corresponding to B_i can be observed, e.g., a defect window ECU leading to an observable permanent defect of the window.
Failure effect: observable behaviour of the system or function that can be linked to the failure effect, e.g., a specific customer complaint linked to variable CC_i which is set to true if the complaint was reported to the garage, a set of variables DTC_i with each variable DTC_i assigned true if the DTC related to DTC_i is set by the ECU. An example for a failure effect is the customer complaint "window is not moving generally".
Detection: measure to detect the fault. Can be via variable T_i with T_i assigned true if the abstract condition related to T_i was observed by an on-board-monitor, variable G_i assigned true, if the off-board test function's condition corresponding to G_i is observed, etc. An example for a detection is the garage test function "check wire to window motor for electric flow" which holds true if electronic flow can be measured by the garage.
Occurrence: variable CM_i is assigned true if its corresponding measure was initiated after fault was detected, e.g., setting of a DTC and deactivation of window closing function if jammings cannot be detected.

With the outlined mapping of the FMEA to variables, we now construct the following logical statements for the respective abstraction layers, practically encoding the fault models of the respective layers:
Fault model as seen from customer view:
$\neg O_i \rightarrow \bigvee_i CC_i$
(Avižienis, A. *et al.*, 2004) define a system failure abstractly as an event that occurs when the delivered service of a system deviates from correct service. A failure of a customer function leads to an unexpected output which is noted by assigning true to variable $\neg O_i$ corresponding to the output. The unexpected behaviour leads to customer complaint(s) and a repair of the car in the garage.
Fault model as seen from garage view:
After a customer reports a complaint to the garage, the garage starts reasoning about the present faults in the car. If a fault is assumed to be present, we assign true to the related variable F_i. The garage starts the repair with analysing and verifying the customer complaints. Since some failures are well known and their fault reasons deducible just by analysing the customer's complaints, we also have $\bigvee_i CC_i$ with variable CC_i assigned true if the corresponding complaint was reported to the garage (e.g., jamming of a window or warning lights): $\bigvee_i CC_i \rightarrow \bigvee_i F_i$.
If the variables F_i have different replaceable units, their related fault causes have to be further isolated. Therefore the garage either needs additional off-board measurements from the garage testing system with variable G_i assigned true if the related measurement is observed in the process or, especially if the faults are in relation with an ECU, the set of DTCs with variable DTC_i assigned true if the related DTC was set:

$$\bigvee_i CC_i \rightarrow \bigvee_i F_i, \quad \bigvee_i G_i \rightarrow \bigvee_i F_i,$$
$$\bigvee_i DTC_i \rightarrow \bigvee_i F_i$$

Note that as mechanical faults cannot be detected directly by the ECU, no corresponding DTC is stored.
Fault model as seen from technical view:
$\bigwedge_i T_i \rightarrow DTC_i$.
The ECU analyses several measurements with related variable T_i assigned true if the measurement was observed and sets a DTC resulting in assigning true to the respective variable DTC_i. Within the ECU, diagnosis functions analyze the DTCs and reason for a faulty function or component. The goal of a DTC or DTC pattern is to distinctively identify a fault:
$\bigwedge_i DTC_i \rightarrow F_i$.
Since faults of safety-relevant functions can have severe effects, they have to be prevented or mitigated before leading to failures which is called pre-emptive diagnosis. The preventive diagnosis on-board the ECU initiates counter measures (occurrences in FMEA) related to variable CM_i assigned true if the counter measure was initiated: $F_i \rightarrow \bigwedge_i CM_i$.
The initiation of counter measures can be an inhibition of functions visible to the customer and thus leading to a customer complaint related to CC_i which is then set to true: $CM_i \rightarrow CC_i$.
For instance, the jamming detection function of the window is safety-relevant as its failures lead to the jamming of objects. If the power window controller's diagnosis detects those faults, the window moving functions are disabled leading to a customer complaint that the window is not working generally.

In consequence, we obtain a formal fault model describing the relations between faults and their observable behaviour in terms of statements in canonical variables. Obviously, these statements can then be transformed into a clause normal form, and be deployed on a testing system for the garages and those

from the development layer on an ECU, thus solving a propositional satisfiability problem. Each solution obtained is then a potential diagnosis candidate, i.e., an explanation of the fault's cause(s).

5 HIERARCHICAL FAULT MODELLING

In this section we show the diagnosis of real-life faults affecting the function "automatic closing of window" from our case study. The diagnosis is performed on the three different levels of abstraction defined earlier. The fault model then consists of logical statements, modelling the specified behaviour of the system as seen from the perspective of the introduced roles. The development view from the fault model is deployed on the ECU and the garage view as a testing system, which supports the garage in the repairs. At runtime, the observable behaviour as shown in Fig. 3, is added to the fault model thus forming the mentioned satisfiability problem, which can then be efficiently solved using a so-called propositional SAT-solver. Let us examine faults affecting the window controller's function "automatic closing of the window", and which lead to the observable fault effect "window does not move generally". The reason for this effect is one of the following faults shown in table 1. Each of the variables $F_1, ..., F_7$ encodes a different fault responsible for this problem.

Fault	Description
F_1	wire from button sensor to ECU defect
F_2	wire from ECU to window motor defect
F_3	wrong polarization of Hall-effect sensors
F_4	button defect
F_5	button sensor defect
F_6	ECU defect
F_7	window motor defect

Table 1: Faults of case study

Fault model as seen from customer view:
$\neg C2 \rightarrow CC_1$
The customer requested an automatic closing of the window, but the observable behaviour of the window was not as expected. The customer reports to the garage the complaint encoded as CC_1, i.e., "window does not move generally". Note that the following redundancy showing up in the formulation stems from the automatic conversion of FMEA artefacts into the logical statements. Although they may potentially dampen performance, they are not a logical problem per-se.
Fault model as seen from garage view:
$(\neg C2 \rightarrow CC_1) \bigwedge (CC_1 \rightarrow F_1 \vee \ldots \vee F_7) \bigwedge$
$(\neg C2 \rightarrow F_1 \vee \ldots \vee F_7) \bigwedge (\neg G1 \rightarrow F_1 \vee F_4 \vee F_5) \bigwedge$
$(\neg G2 \rightarrow F_2 \vee F_6) \bigwedge (\neg T3 \rightarrow F_3 \vee F_7)$
The garage analyses (and verifies) the customer complaint and knows that if the window does not move generally, at least one of the faults $F_1, ..., F_7$ is usually present.

With the garage test functions "check cable from button to ECU for electric flow", "check cable from window ECU to window for electric flow" and "check cable from Hall-effect sensors to ECU for electric flow", the garage is able to collect observations G_1, G_2, G_3 where each variable is true if electrical flow can be measured. Table 2 shows the suggested repair measures for the faults.

Fault present	Suggested repair measure
F_1	replace wire from button to ECU
F_2	replace wire from ECU to motor
F_3	repolarise ECU's Hall-effect sensor input or replace ECU
F_4	replace button
F_5	replace button
F_6	flash or replace ECU
F_7	replace window motor

Table 2: Suggested repair measures for faults

Faults F_2 and F_6 as well as Faults F_3 and F_7 are faults which have equivalent observations, but with repair measures for different repairable parts. Hence, these faults have to be distinguished on a deeper technical level to avoid expensive repeated repairs, when only viewed on the same abstract level. Therefore the garage needs support by the diagnosis on the ECU. Faults F_1, F_4, F_5 can neither be distinguished by the ECU nor the garage meaning that the garage has to try both different repair measures.
Fault model as seen from technical view:
$(T_1 \rightarrow DTC_1) \bigwedge (T_2 \rightarrow DTC_2) \bigwedge (T_3 \rightarrow DTC_3) \bigwedge$
$(DTC_3 \rightarrow F_3) \bigwedge (F_3 \rightarrow CM_1)$.
The ECU performs observations with monitors examining the discretised Hall-effect sensor signal which is the variable $D11$ from fig. 3. The monitors and its related variables are "discretised Hall-effect sensor signal did not change after movement of window" and T_1, "no discretised Hall-effect sensor signal available after window movement"' and T_2, as well as "discretised Hall-effect sensor signal indicates window movement in wrong direction" and T_3. If one of the measurements holds, the ECU sets a corresponding DTC to support the garage.

Fault F_3 is an example for a fault mitigation. If the sensor signal indicates a movement contrary to the commenced, jammings cannot be detected. As the jamming detection function is safety-relevant, the ECU has to initiate the counter measure "deactivate window permanently" related to variable CM_1. In this case the window controller deactivates the window permanently to prevent critical failures. In case of the other faults, the window motor cannot be moved.

Including the DTC set enables the garage to distinguish the faults F_2, F_6 and F_3, F_7 and initiate the repair measures from Table 2.

$$\neg G2 \wedge DTC_1 \rightarrow F_2$$
$$\neg G2 \wedge \neg DTC_1 \rightarrow F_6$$
$$\neg G3 \wedge DTC_3 \rightarrow F_3$$
$$\neg G3 \wedge DTC_2 \rightarrow F_7$$

With Fault F_6 being a fault of the ECU itself, no relevant DTCs are present.

6 SUMMARY AND OUTLOOK

We presented a methodology for diagnosing automotive systems. The methodology reflects and supports

on-board as well as off-board diagnosis, the latter being performed by the garage. Our methodology centres around the respective roles involved in the diagnostic process and reflects these by allowing diagnosis on different levels of abstraction to cover all occurring faults within the domain. What is more, in Sec. 5 we showed that our methodology seamlessly integrates two of the most widely used diagnosis techniques in the automotive domain, namely the expert-based diagnosis which we employed for diagnosing the discretised Hall-effect sensor signals and the model-based diagnosis, which we employed for diagnosing the semi-mechanical components window and button. Furthermore, using the different layers of abstraction, we showed how to make the high complexity of the diagnosis task manageable in practise.

The proposed methodology is currently under evaluation for the off-board case, i.e., garage diagnosis, where it is possible to run, e.g., a solver on a dedicated tester which is equipped with sufficient resources to undertake this task as outlined in this paper. It is our hope that results from this evaluation allow us to determine and isolate some of the constraints that need to be addressed before we can also use our approach to on-board diagnosis. Depending on the on-board system, i.e., whether or not it is safety-critical, there are stringent constraints for on-board software. Safety-critical systems, for example, must not dynamically allocate memory (cf. MISRA Coding Guidelines, rule 118 (MISRA, 2004)). Hence when diagnosing such components, it is not yet clear how and where the diagnosis engine should be executed, and what memory requirements have to be met.

REFERENCES

(Avižienis, A. *et al.*, 2004) Avižienis, A., Laprie, J. C., Randell, B., and Landwehr, C. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

(Bauer, 2007) Andreas Bauer. *Model-based runtime analysis of distributed reactive systems*. PhD thesis, Technische Universität München, 2007.

(Broy, M., 2006) Broy, M. Challenges in Automotive Software Engineering. In *International Conference on Software Engineering (ICSE 2006)*, pages 33–42, Shanghai, China, May 2006. ACM.

(Buchanan, B. and Shortliffe, E., 1984) Buchanan, B. and Shortliffe, E., editors. *Rule-Based Expert Systems—The MYCIN Experiments of the Stanford Heuristic Programming Project*, volume 1 of *The Addison-Wesley Series in Artificial Intelligence*. Addison-Wesley, 1984.

(Ioannou, P. and Chien, C., 1993) Ioannou, P. and Chien, C. Autonomous Intelligent Cruise Control. *IEEE Transactions on Vehicular Technology*, 42(4):657 – 672, 1993.

(Isermann, R., 2004) Isermann, R. Model-based fault detection and diagnosis: status and applications. In *Proceedings of the 16th IFAC Symposium on Automatic Control in Aerospace*, St. Petersburg, Russia, June 2004.

(Kleer, de J. and Williams, B. C., 1987) Kleer, de J. and Williams, B. C. Diagnosing multiple faults. *AI*, 32(1):97–130, 1987.

(MISRA, 2004) Motor Industry Software Reliability Association MISRA. MISRA-C:2004 – Guidelines for the use of the C language in critical systems, 2004.

(Nolte, T. *et al.*, 2005) Nolte, T., Hansson, H., and Bello, L. L. Automotive Communications - Past, Current and Future. In *10th IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2005)*, volume 1, 2005.

(Nyberg, M. *et al.*, 2001) Nyberg, M., Stutte, T., and Wilhelmi, V. Model based diagnosis of the air path of an automotive Diesel engine. In *IFAC Workshop: Advances in Automotive Control*, 2001.

(Picardi, C. *et al.*, 2002) Picardi, C., Bray, R., Cascio, F., Console, L., Dague, P., Dressler, O., Millet, D., Rehfus, B., Struss, P., and Vallée, C. IDD: Integrating diagnosis in the design of automotive systems. In *Proceedings of the Fifteenth European Conference on Artificial Intelligence (ECAI 2002)*, pages 628 – 632, Lyon, France, July 21 – 26 2002.

(Prestl, W. *et al.*, 2000) Prestl, W., Sauer, T., Steinle, J., and Tschernoster, O. The BMW active cruise control ACC. *Society of Automotive Engineers (SAE) Transactions*, 109(7):119–125, 2000.

(Pretschner, A. *et al.*, 2007) Pretschner, A., Broy, M., Krüger, I., and Stauner, T. Software engineering for automotive systems: A roadmap. In *Future of Software Engineering (FOSE'07)*, pages 55 – 71. IEEE Computer Society, Mai 2007.

(Ramsden, E., 2006) Ramsden, E. *Hall-Effect Sensors—Theory and Application*. Elsevier, 2. edition, 2006.

(Reiter, R., 1987) Reiter, R. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57 – 95, 1987.

(Sampath, M. *et al.*, 1995) Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, September 1995.

(Stamatis, D. H., 2003) Stamatis, D. H. *Failure Mode and Effect Analysis: FMEA from theory to execution*. American Society for Quality (ASQ), 2. edition, 2003.

(Struss, 2006) P. Struss. A model-based methodology for the integration of diagnosis and fault analysis during the entire life cycle. In *Proc. of SAFEPROCESS 2006*. Elsevier, 2006.

(Thiel, S. and Hein, A., 2002) Thiel, S. and Hein, A. Modeling and using product line variability in automotive systems. *IEEE Software*, 19:66–72, 2002.

(Tripakis, S., 2009) Tripakis, S. A combined on-line/off-line framework for black-box fault diagnosis. In *9th International Workshop on Runtime Verification (RV 2009)*, pages 152 – 169, Grenoble, France, 2009. Springer.