Improving the Development Process for Automotive Diagnostics

Ingolf Krüger, Massimiliano Menarini and Filippo Seracini
Department of Computer Science and Engineering
UC San Diego
La Jolla, California, USA
{ikrueger, mmenarini, fseracini}@ucsd.edu

Maximilian Fuchs[1], Jens Kohl[2]
[1]Diagnostic Department, [2]Strategy R&D
BMW Group
Munich, Germany
{maximilian.fuchs, jens.kohl}@bmw.com

***Abstract*— We present a vision for the evolution of automotive diagnostics as car electronics are increasingly implemented as distributed systems. We first analyze the state of the art in the automotive development process, and we identify how distributed functionalities challenge this process and lead to higher development and field maintenance costs. To address these challenges, we modify the development process currently used by automotive manufacturers and propose the use of a model-based approach. Finally, we outline future research directions for car diagnostics.**

Keywords- Model-based diagnostics, automotive diagnostics, automotive development process.

I. INTRODUCTION

Modern cars are complex, distributed, software-intensive systems. A luxury vehicle can have up to 100 electronic control units (ECUs) connected to multiple networks [1]. Most of the new functionalities provided to users depend heavily on software and arise from the interaction of multiple ECUs. The range of functionalities in cars varies substantially from safety-relevant, real time critical systems, such as airbag or driving assistance, to soft real time systems, such as infotainment.

The two key players in the automotive industry are the car manufacturers, also known as Original Equipment Manufacturers (OEMs) and a multi-tiered ecosystem of suppliers. The interaction and the coordination of this ecosystem of suppliers is a major challenge each OEM must actively manage for the success of the entire development process. In a typical automotive development process, the OEM designs the overall vehicle, and suppliers produce the components (ECUs) based on high level specifications from the OEM. The OEM then integrates all components into the car. This organizational structure adds a layer of complexity on top of an already difficult integration task, especially for the distributed functions. Since different components can be developed by different suppliers (which also own the Intellectual Property (IP) of their internals), the OEM does not have complete knowledge of their internal implementation details. In an effort to mitigate some of this complexity, new middleware and reusable services are being developed. For example, AUTOSAR [2] is an initiative aiming at creating a common infrastructure and a set of common services that foster reuse of software across different OEMs and suppliers.

Customers consider safety as one of the most important aspects when choosing a new car [3]. An effective diagnostic system is important to guaranteeing car safety [4]. First and foremost, diagnostics must ensure that all potentially critical failures do not endanger passengers and the environment. In many countries, OEMs are also required by law to provide a diagnostic system. For example, California's On Board Diagnostics (OBD) II [5] rules require each car to continuously monitor its emission-relevant functions while driving. Another reason to provide an effective diagnostic system is that it enables garages to efficiently identify which parts to replace.

In this paper, we discuss the current development process for diagnostics in a car, and we propose improvements that address technical issues. We are aware that there is a variety of non-technical issues that influence the development of a vehicle; we discuss them in [4].

Currently, OEMs integrate the diagnostic systems developed by suppliers for the various ECUs during the system integration phase. Since this process was created to integrate local functions, the paradigm change towards distributed functions poses challenges to this integration process.

Diagnostics observes a system for behavior deviating from its specifications. In case a deviation from the specifications is found, a diagnostic system stores information on the affected component in form of Diagnostic Trouble Codes (DTCs). A supplier developing a component must provide the diagnostics for the component according to the OEM's specification. Information about the fault and the ECU that generated it must be provided to support repairs in garages.

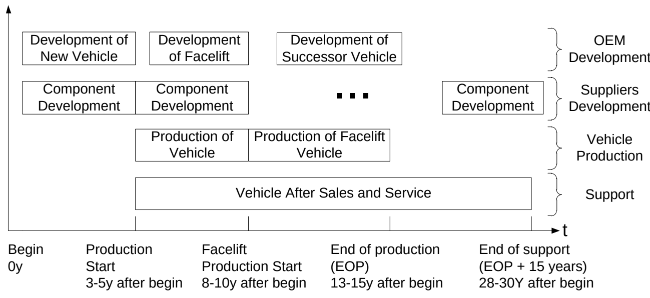
In this paper, we use the definition of error, fault, and failure as given in [6]. A failure is an event that occurs when the service delivered by a system deviates from the correct one. An error is part of the system state that is liable to lead to subsequent failure. A fault is the adjudged or hypothesized cause of an error. Therefore, a fault causes an error, and an error can lead to a failure.

Software functions are used to support many safety-relevant features of a car. Since failures of these functions can have critical effects, their faulty behavior must be detected, and any dangerous effect must be prevented or at least mitigated. A major challenge for both OEMs and suppliers is how diagnosis should deal with distributed, cross-component functions. These functions consist of several sub-functions deployed across car components that interact via messages. With distributed functions, transitive faults may occur. A *transitive fault* propagates across sub-functions and can affect the whole car. It is very difficult for OEMs and suppliers to identify transitive faults if the OEMs must reproduce the problem on their test environment. A fault not identified during development is called *unknown fault*. Responding to unknown faults is extremely difficult for diagnostic systems; therefore, the goal of a development process for diagnostics is to minimize the chances of having unknown faults.

Model-based diagnostics is a possible solution to address diagnosis of transitive or unknown faults. With this technique, a model of the specified behavior is compared with the observed behavior of the system. Diagnostic models require only an abstract representation of the various components; therefore, they are a viable avenue when the IP is split across multiple organizations. Still, reasoning about the fault's root cause is hard. In fact, there can be interactions and functional dependencies that exist in the components but are not reflected in the models. Therefore, even using model-based diagnostics an iterative process between OEMs and suppliers is still required.

Outline — The rest of the paper is structured as follows. First, we discuss the current automotive development process. Then, we present an evolution of the diagnostic development process that leverages model-based diagnostics. Finally, we identify a future direction for car diagnostics and draw our conclusions.

II. CAR DEVELOPMENT LIFE CYCLE



Development of New Vehicle
Development of Facelift
Development of Successor Vehicle
OEM Development
Component Development
Component Development
...
Component Development
Suppliers Development
Production of Vehicle
Production of Facelift Vehicle
Vehicle Production
Vehicle After Sales and Service
Support
t
Begin 0y
Production Start 3-5y after begin
Facelift Production Start 8-10y after begin
End of production (EOP) 13-15y after begin
End of support (EOP + 15 years) 28-30Y after begin

Figure 1. A typical car development life cycle. (Based on [7])

From the OEM's point of view, a typical car model's life cycle can be divided into 3 phases: development of a new vehicle, facelift development, and after sales. During each phase, the OEM and suppliers follow a development process abiding to two ISO standards: [8] for software components, and [9] for safety-relevant systems.

Development of a New Vehicle. Producing a new car model is a very long term commitment for an OEM. A car life cycle typically spans over 30 years and comprises multiple phases and players (Figure 1). The project for a new car model is initiated by the OEM with the "Development of New Vehicle" phase. Here, the OEM designs the new car in all its domains (e.g. powertrain, chassis) and defines all the specifications for the different components that will be developed by its network of suppliers. "Component Development" phases are performed by suppliers while the OEM is performing the "Development of New Vehicle". These phases require a strict collaboration between the OEM and suppliers. Typically after 3-5 years, the car is ready for the market and the "Production of Vehicle" starts. The new car is sent to assembly lines and car sale begins.

Facelift Development. Shortly after the car arrives on the market, the "Development of Facelift" starts. The car design is updated with new features based on feedback from customers and garages. Internal components are also updated, hence requiring other "Component Development" phases. After 3-4 years, the facelift development is completed. The updated car begins production -"Production of Facelift Vehicle"- and the previous version is discontinued. Typically, OEMs only have one facelift per car model. One or two years after the "Facelift Development" phase terminates, the design for the successor of this model starts with a new "Development of New Vehicle" phase ("Development of Successor Vehicle" in Figure 1).

After sales. In parallel with the start of car production, the "Vehicle after Sales and Service" phase starts. This is the longest phase of vehicle life cycle, and it typically lasts for 15 years after the end of facelift production. In this phase, feedback from customers and garages is collected and used for improving several aspects of the car. The OEM and suppliers may go through multiple "Component Development" cycles in case urgent updates (i.e. recalls) are required.

III. CURRENT DEVELOPMENT PROCESS

Current practices for developing automotive diagnostics are based on a stream of documents, exchanged between OEMs and suppliers, which aim to create an integrated view of car diagnostics. Across different phases of the development process, OEMs provide suppliers with diagnostic requirements and details about possible global states of the car. In return, suppliers provide documentation on the faults covered by the supplier's diagnostics and the error codes (DTCs) that can be reported. This documentation covers the meaning and the cause of each DTC together with possible countermeasures the component can adopt to mitigate faults. OEMs define requirements for the general diagnostics. OEMs also define the information suppliers must report and which car status information is available to each component.

Figure 2 outlines an abstracted development process for a car, which would be enacted during the two development phases of the life cycle depicted in Figure 1. The process in Figure 2 follows the V-Model. The general development phases are stated inside the V picture; the numbered callout boxes describe the activities currently performed in each development phase to produce the diagnostic system.

Project Definition and Implementation. The first step in developing a car is the "System Requirements Analysis". In this phase, an OEM specifies the global requirements for the vehicle together with the high level requirements for diagnostics (box 1). The next step is “System Functional Specification/Design”. Here, the OEM divides car functionalities into a network of functions assigned to the various components and design integration tests. “Software Requirement Analysis” is the step in which an OEM works with suppliers to specify the requirements and acceptance test for the software. The suppliers can then start the “Software Design” phase, where the software is designed together with the unit tests. In this phase, suppliers also analyze and design the local diagnosis for their components (box 2). At the bottom of the V, OEMs and the suppliers individually develop models and code.

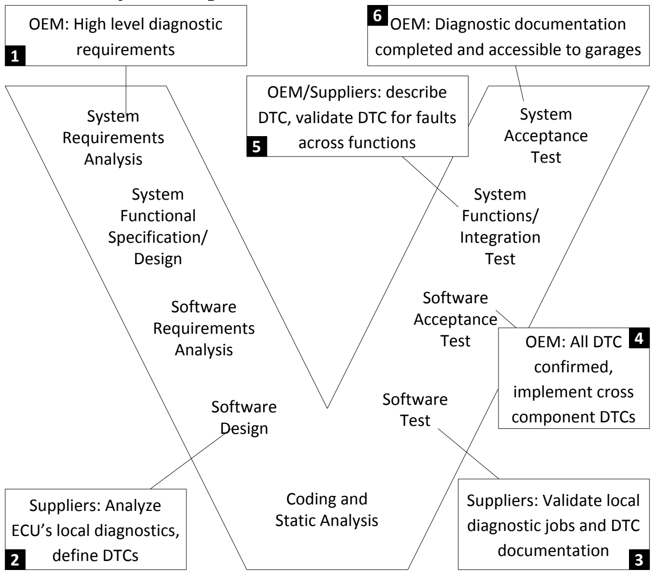

OEM: High level diagnostic requirements
1
6
OEM: Diagnostic documentation completed and accessible to garages
System Requirements Analysis
OEM/Suppliers: describe DTC, validate DTC for faults across functions
5
System Acceptance Test
System Functional Specification/ Design
System Functions/ Integration Test
Software Requirements Analysis
Software Acceptance Test
OEM: All DTC confirmed, implement cross component DTCs
4
Software Design
Software Test
Coding and Static Analysis
Suppliers: Analyze ECU’s local diagnostics, define DTCs
2
Suppliers: Validate local diagnostic jobs and DTC documentation
3

Figure 2. Exemplary development process for automotive diagnostics.

Project Test and Integration. Going up the right leg of the V, the validation phases begin. In these phases, suppliers and OEMs validate their implementations by testing that all requirements are fulfilled. The first test phase is the “Software Test”. In this phase, suppliers also validate their local diagnostics and the DTC documentation (box 3). In the “Software Acceptance Test”, the OEM verifies the diagnostic implementation of the suppliers’ components and confirms the proper usage of each DTC. The OEM also addresses cross component failures and defines the proper DTCs for them. In the next phase, “System Functions and Integration Test”, the OEM and suppliers collaborate to validate all DTCs for faults of cross-component functions and ensure that the description of each DTC is correct. In the final phase, “System Acceptance Test”, the OEM validates the car’s completion and that each ECU fulfills its requirements. The diagnostic documentation must be completed and made available to garages before the car is sold on the market.

The development process depicted in Figure 2 was devised for systems where each function was localized in independent components. Recently, automotive electronic architecture has evolved towards functions distributed across components. This evolution poses challenges to diagnostics implementation under the current development process.

Figure 2 shows that the current process does not design diagnostics into the system until the very end of the design phase. Furthermore, this is done locally by the suppliers of each ECU. Therefore, incompatibilities between diagnostic systems created by different suppliers are discovered only when OEMs and suppliers integrate diagnostic functions (during the “System Function and Integration Test”), rather late in the development process. Moreover, there is no formal diagnostics model exchanged between parties – instead, OEMs and suppliers informally share knowledge via documents, tables etc. Therefore, it is hard to validate the design and anticipate problems before integrating the various ECUs in the later phases of the development. In particular, there is no global view of the interactions between various ECUs produced by different suppliers, so diagnostics focus mainly on each ECU in isolation and on its internal functionalities.

The lack of an integrated diagnostic model spawns additional problems. First, it limits the reuse of diagnostics across car models and generations. Because there is no formal diagnostics model, diagnostic functions are coded over and over by hand, even if the requirements that originate them are recurring. Recently, frameworks such as AUTOSAR have been introduced to try to mitigate this problem. However, given their level of abstraction, such frameworks can alleviate the diagnostic code reuse problem only for a subset of functions. Redeveloping diagnostic functions, obviously, leads to higher costs, which come from the need to redevelop functions, maintainability issues (because of the different code bases to maintain across models and generations), and verification and validation costs.

IV. IMPROVING THE DEVELOPMENT PROCESS

Based on the experience we gathered by analyzing a real car development process of an OEM, we identify areas that can be improved. The first is the communication between OEMs and suppliers in the development phases. In the current state of the art, communication during the design and development phases mostly relies on requirements and high level designs in the form of textual specifications, tables describing the communication protocols, and message formats. While these documents include diagnostic requirements, there is no formal model of the interactions between components or the diagnostic information exchanged between them. This *status quo* poses major challenges to OEMs and suppliers alike. Neither party can validate their designs or anticipate problems, which are instead discovered and solved in the integration phase. It is well-known that the later a problem is discovered in the process, the more expensive and complex it is to address it.

A second area of improvement is in the detection and mitigation of unexpected (possibly safety critical) function interactions. Faults can be localized in a single ECU or can involve the interaction of multiple ECUs. We divide faults into two groups: *known* and *unknown* faults. Known faults have been identified during the development phase and the diagnostic system of the car can detect them. Unknown faults have not been identified during development, and currently only model-based diagnostics techniques can detect them. Model-based techniques detect unknown faults by observing at runtime that the behavior of a function deviates from the modeled one. The most difficult faults to identify are those with multiple ECUs involved. Currently, the only solution to cope with unknown faults is for OEMs to try to reproduce the problem after learning of its existence from customer complaints or reports from garages.

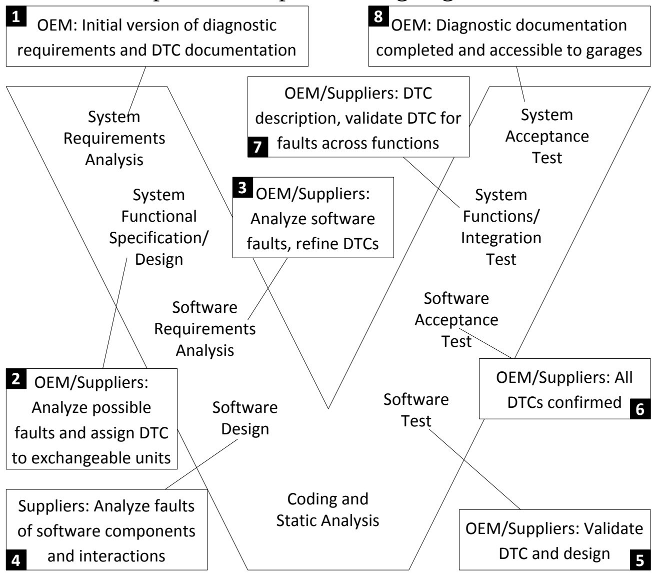

1 OEM: Initial version of diagnostic requirements and DTC documentation
8 OEM: Diagnostic documentation completed and accessible to garages
OEM/Suppliers: DTC description, validate DTC for faults across functions 7
System Requirements Analysis
System Acceptance Test
System Functional Specification/ Design
3 OEM/Suppliers: Analyze software faults, refine DTCs
System Functions/ Integration Test
Software Requirements Analysis
Software Acceptance Test
2 OEM/Suppliers: Analyze possible faults and assign DTC to exchangeable units
Software Design
Software Test
OEM/Suppliers: All DTCs confirmed 6
Suppliers: Analyze faults of software components and interactions 4
Coding and Static Analysis
OEM/Suppliers: Validate DTC and design 5

Figure 3. Updated development process for car diagnostics.

From these challenges arise an opportunity to dramatically improve the provisioning and exploitation of diagnostic vehicle information. A common thread is the lack of principled composition and analysis of the interactions between sets of ECUs.

Figure 3 depicts an improved development process that addresses the shortcomings of the current process described in Figure 2. Space constraints prevent us from presenting and discussing here a full case study. One of the authors developed a complete diagnostic system for a car window control following the outlined improved process and presented it in Chapter 6 of [10]. The key improvement is the introduction of diagnostic concerns at the very beginning of the process – in the "System Functional Specification and Design" phase, OEMs and suppliers collaborate on analyzing possible faults and assign DTCs to components (see box 2). In this phase, OEMs can engage with suppliers in identifying faults occurring across components and, therefore, can diminish the risk of having unknown faults later in the development process. Also, the "Software Requirement Analysis" step requires OEMs and suppliers to analyze software faults and refine DTCs (box 3). During the "Software Design" phase, suppliers must take into account not only faults local to their components but also faults that could arise from interactions with other components. This new process improves upon the state of the art by starting a productive interaction on failure management between OEMs and suppliers before the real components are developed. Diagnostics become a major part of the design and architecture of the vehicle. This reduces the risks of finding incompatibilities in later integration phases.

The proposed process improves upon the current one by also increasing the agility of the interactions between OEMs and suppliers. Continuous communication between the parties since the early stages of the process enables shorter iterations when suppliers validate their diagnostic systems with OEMs. These shorter iterations support changing requirements and potentially reduce time-to-market.

For this process to be applicable, suppliers and OEMs must have a common language to describe possible diagnostic problems. Since the internals of most of the car components are IP of the various suppliers, achieving a shared knowledge of diagnostics is a major challenge. A viable solution to overcome this challenge must abstract from internal details of the components. We propose the use of model-based diagnostics as a key enabler for the improved development process.

V. MODEL-BASED DIAGNOSTICS

Given our analysis of the current situation, a substantial improvement of the diagnostic development process can result from the support of modeling techniques. Model-based diagnostics would allow diagnostic models to be reused across vehicles and generations.

Almost all diagnostic methodologies used within the automotive domain are based on two different paradigms: expert-based and model-based diagnostics. Diagnostic expert systems codify an expert's knowledge of a system in form of rules in a knowledge base. An inference engine diagnoses a system by combining the codified rules together with observations taken at runtime. One of the first and most well-known approaches based upon this paradigm is MYCIN [11]. However, expert-based diagnostic systems have challenges diagnosing unknown failures because the system can only identify faults that have been codified in the rules.

In contrast to expert systems, model-based systems formally codify the system's behavior as models. The behavior model can either be coded with (residual) equations (Fault Detection and Isolation (FDI), [12]), predicate logic formulae [13] or by a discrete set of states and events (Discrete Event Systems (DES) diagnosis, [14]).

At runtime model-based diagnostics examines the system's behavior by taking observations and comparing the actual with the specified behavior. In case of a discrepancy, the system must reason to determine the root cause. Model-based diagnostics separates the system model from the inference engine, thus facilitating reuse. Different reasoning engines can be used with model-based techniques. For example, qualitative reasoning (using ranges of values or fuzzy logic) supports concepts such as "input value too high". Hence, model-based reasoning can detect unknown faults.

While the model-based diagnostic techniques reviewed in this section are promising approaches for the automotive domain, the proposed automotive process requires a tailored model-based technique that focuses on abstract interaction models. Such models must be able to capture the interactions across different components without the need to model internal details of the participating components. For example, a description of the interaction for locking a car should describe only the communication protocol between the key fob, the alarm system, the actuators that release the door locks, and the light control system. A more sophisticated locking system could also communicate with the seat adjustment engines to move the driver seat into its preferred position, the radio to change presets, and so on. Previous work on architecture definition languages (ADL) for describing systems' interactions, could contribute to the evolution of car diagnostics. For instance, [15] presents a service ADL that supports failure management in cars. The ability to abstract from components' implementation details is a key enabler to cope with the typical automotive organizational structure where IP is to be protected across component's boundaries. Moreover, focusing on abstract components' interaction protocols increases the reusability of failure models across product lines and generations.

VI. FUTURE OF CAR DIAGNOSTICS

The process and model-based approach we propose in this paper are important steps in the direction of improving car diagnostics. However, we envision that, with the growing complexity of automotive software, the need for more powerful and efficient diagnostic systems will be even greater. For example, with the advent of drive-by-wire, where the mechanical link between the steering wheel and the car's wheels is removed, we believe that fault management and regulatory requirements for formal diagnostic models will become commonplace.

For automotive systems to evolve as we envision, researchers must support industry in its effort to create this new generation of vehicles. Important research questions that must be answered span multiple domains, such as formal specification of behavior, faults, and remedies; maintenance and reuse of diagnostic systems; runtime reconfiguration of diagnostic systems; and abstraction techniques for creating and refining components' models.

A line of work worth exploring is the use of service-oriented architectures for on- and off-board diagnostics (cf. [10]). Loose-coupled services have the potential for increasing the reuse and reduce complexity of diagnostic services. Services could potentially support dynamically adding and removing diagnostic monitors at runtime, thus, saving storage and computation power (relatively scarce and expensive resource on board of cars). In this scenario, garages could pinpoint faults by increasing the on-board diagnostic capabilities only in the areas affected by the faults.

VII. CONCLUSIONS

The increase in software-enabled functionalities of modern cars caused a paradigm shift in the architecture of automotive systems. It evolved from a set of independent functions localized in specific components to a network of functions distributed across many. In this paper, we observed that the development process currently used in the automotive industry is not optimized for developing diagnostics of distributed functions. We proposed an improved development process for car diagnostics that better supports the architecture of modern cars. This process promises to reduce the time to market and the development costs of new vehicles. However, given the organizational boundaries and IP issues between car companies and component suppliers, the process must be supported by a model-driven approach to diagnostics. The paper proposes modeling techniques that abstract components' implementations. This eliminates IP issues that could impede the adoption of the proposed diagnostic process and foster reuse of diagnostic models across product lines.

ACKNOWLEDGMENT

Our work was partially supported by NSF grants CCF-0702791, CNS-0963702, and the California Institute for Communication and Information Technology. We thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] K. V. Prasad, M. Broy, and I. Krüger, "Scanning Advances in Aerospace & Automobile Software Technology," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 510–514, 2010.

[2] "AUTOSAR Automotive Open System Architecture." [Online]. Available: http://www.autosar.org/. [Accessed: 23-Jan-2012].

[3] B. Betts, "Recall and reboot," *Engineering & Technology*, vol. 5, no. 4, pp. 54–55, Mar. 2010.

[4] I. Krüger, "Opinion: An Outlook on Automotive Software," *IEEE Embedded Systems Letters*, vol. 2, no. 1, pp. 14–15, Mar. 2010.

[5] California Air Resources Board (CARB), *Malfunction and Diagnostic System Requirements -- 1994 and Subsequent Model-Year Passenger Cars, Light-Duty Trucks, and Medium-Duty Vehicles and Engines (OBD II)*. Code 13, Section 1968.1: California Code of Regulations (CCR), 1994.

[6] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

[7] J. Schäuffele and T. Zurawka, *Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen*. Vieweg +Teubner, 2010.

[8] ISO - International Organization for Standardization, *Standard for Systems and Software Engineering - Software Life Cycle Processes*. ISO/IEC 12207:2008, 2008.

[9] ISO - International Organization for Standardization, *Road Vehicles - Functional Safety*. ISO 26262 (Part 1-9), 2011.

[10] J. Kohl, "Efficient diagnosis of distributed functions of automotive control units," Ph.D. Thesis, TU Munich, 2012.

[11] B. G. Buchanan and E. H. Shortliffe, *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, 1984.

[12] R. Isermann, "Model-based fault-detection and diagnosis – status and applications," *Annual Reviews in Control*, vol. 29, no. 1, pp. 71–85, 2005.

[13] J. de Kleer and J. Kurien, "Fundamentals of model-based diagnosis," in *Fault Detection, Supervision and Safety of Technical Processes*, Washington, D.C., USA, 2003, vol. 1, pp. 25–36.

[14] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, Sep. 1995.

[15] V. Ermagan, C. Farcas, E. Farcas, I. H. Krüger, and M. Menarini, "A Service-Oriented Approach to Failure Management," in *Tagungsband des Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme IV*, 2008.