Architecture-based Hybrid Approach to Verify Safety-critical Automotive System Functions by Combining Data-driven and Formal Methods

Adina Aniculaesei, Andreas Vorwald, Meng Zhang, Andreas Rausch
Institute for Software and Systems Engineering
Technische Universität Clausthal
Clausthal-Zellerfeld, Germany
{adina.aniculaesei, andreas.vorwald, meng.zhang, andreas.rausch}@tu-clausthal.de

***Abstract*—Safety-critical automotive functions are required to satisfy stringent safety requirements. To guarantee the safety of such functions, their conformance with industry approved standards as well as statutory regulations must be ensured. Testing is the main method for checking automotive system functions, yet testing is incomplete and cannot show correctness. Inherent uncertainties in the physical environment introduce non-determinism in testing, increasing the difficulty of replicating environmental stimuli relevant for edge cases, and thus, the effort invested in road tests to produce statistical significance. Formal verification techniques are able to show correctness and are recommended for functions with higher automotive safety integrity levels (ASIL), e.g. for ASIL D. However, formal verification has scalability issues in case of highly complex automotive systems and heterogeneous sensor data received as inputs. To address these challenges, this paper proposes a novel architecture-based approach, which combines data-driven methods with formal methods for the verification of safety-critical automotive functions, with consideration of the system decomposition within the functional system architecture. We illustrate the application of our concept on two industrial automotive functions, speed estimation and exhaust aftertreatment, and report on results and lessons learned.**

Keywords-safety-critical automotive functions; data-driven methods; formal verification methods; automotive software architectures; system verification test

I. INTRODUCTION

Safety-critical automotive functions are subject to stringent safety requirements, since any error in the function's behavior can cause a system failure, and thus in worst case, can endanger the safety of the vehicle's occupants and the integrity of the physical environment. In order to guarantee the safety of such functions, their conformance to the industry approved standards and norms as well as to the statutory regulations must be ensured.

Testing is the main method for the verification of automotive functions, as required by the automotive standard ISO 26262 (cf. [17], [18]). However, testing can show the presence of errors in a system, but is not adequate to show their absence (cf. [11]). Thus, testing is incomplete and cannot prove the correctness of the system under test with respect to the defined safety requirements. To be certified as safe with fulfillment of its designated ASIL, a safety-critical automotive function must pass specific tests, e.g. road tests required by ASIL C/D. The inherent uncertainties in the physical environment contribute to non-determinism in testing, making it more difficult to replicate inputs relevant for edge cases. This phenomenon increases the level of effort necessary to be invested in road tests in order to produce statistical significance.

In contrast to testing, formal verification methods can prove correctness (cf. [9]) and are recommended by the standard ISO 26262 for the verification of automotive functions with higher ASIL (cf. [18]). However, formal verification suffers from scalability issues in the case of highly complex automotive functions, which receive heterogeneous sensor data as input from their environment. During the vehicle's operation, an automotive function exchanges constantly data with the other hardware and software systems in the vehicle's technical environment, but may also receive input via external perception sensors from the vehicle's surrounding physical environment. For example, the data exchange between the function under investigation and the other vehicle systems relies on the communication bus system including its process of data quantization, which can lead to deviations from the original data. Furthermore, the quality of the sensor data received from the physical environment can vary time-dependently due to changes in the environmental conditions. A typical example here is the global positioning system (GPS) data quality, which depends on the vehicle's signal reception from the satellites of the global navigation satellite system (GNSS). The GNSS signal reception can be susceptible to errors due to the presence of obstacles between the vehicle and the GNSS satellites, e.g. large buildings in urban area or tunnels.

Research Focus. In order to address the challenges presented by testing and formal verification methods, this paper proposes a novel hybrid approach which combines data-driven and formal methods for the verification of safety-critical automotive functions. Our approach is embedded within the development process of safety-critical automotive functions and uses the development artefacts as inputs for the verification of these functions. Thus, the approach takes the functional architecture of the automotive function as a basis and splits up the modules, which are later analyzed with data-driven techniques from the modules investigated with help of formal methods. The data-driven techniques are applied on modules, which handle sensor inputs with uncertain data quality, in order to obtain a worst-case estimation of the module result based on assumptions about the sensor inputs and constraints with consideration of physics laws. The formal

978-0-7381-3356-0/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSA-C52384.2021.00036

methods are used to check system modules with deterministic computations against the defined system-level safety requirements. Through text substitution similar to Hoare calculus (cf. [16]), the safety requirement of the automotive function is gradually refined into sub-requirements for the worst-case estimation. The results of the worst-case estimation are checked against the derived safety requirement. If the derived sub-requirements are satisfied, then it can be said that the automotive function is conform with respect to the original system-level safety requirement. In order to evaluate the proposed concept, we have constructed two case studies about two industrial automotive functions, speed estimation and exhaust aftertreatment, and report on results and lessons learned in the following sections of this work.

II. Related Work

The interaction of automotive functions with their environment as well as the wear and tear suffered by the hardware components on which these functions are executed can cause faults in the automotive system and in time can lead to a degradation of its performance. Fault diagnosis can be used to preempt the effects which faults have on the system's performance, by identifying and compensating for fault conditions, regardless of the system's operational mode (cf. [24]). Various works have surveyed the field of fault diagnosis and fault-tolerant techniques. Although these surveys present slightly different taxonomies by which they classify the research done in this area, a few categories of fault diagnosis approaches stand out: signal-based (cf. [14], [19]), model-based (cf. [23], [24], [19]) and data-driven (cf. [23], [13], [19]) or knowledge-based (cf. [13]). Model-based fault diagnosis relies on models that describe a relationship between the measured inputs and outputs of the function under analysis. This is often modeled with equations which describe the physical phenomenon taken into consideration in the automotive function (cf. [24]). Signal-based methods extract features from measured signals and base their diagnostic decision on analysis of signal patterns and on prior knowledge of what signal patterns look like in healthy systems (cf. [14]). Instead of using models which characterize physical phenomena, data-driven fault diagnosis uses historic data to generate a model, that mathematically relates system inputs with the system outputs (cf. [24], [13]). Regardless of the method, fault diagnosis detect faults, which originate in the system's hardware and software, thus contributing to the improvement of the system's functional safety. However, diagnosis methods cannot detect errors, which occur due to the unforeseen environmental conditions and cannot ensure the correctness of the system with respect to the system-level requirements.

Formal verification can ensure correctness with respect to defined requirements. The verification approaches range from theorem proving to model checking. Theorem provers like Isabelle/HOL [4] can handle infinite state spaces, but are semi-automated, due to the user input needed to carry out the proof. Model checking tools are fully automated, but can work only on finite state models. Some model checkers can verify formal system models against their system-level requirements, e.g. Uppaal [3]. Other model checkers execute the source code of the system in order to verify it, e.g. PathFinder [27], requiring appropriate orchestration in order to execute all possible situations. Prism [22] and Storm [10] offer mechanisms to handle stochastic system inputs and non-deterministic system behavior. Although it has achieved important results throughout the years, highly complex industrial systems represent a challenge for formal verification due to the scalability issues.

III. Verification of Safety-critical Automotive Functions by a Combination of Data-driven and Formal Methods

An overview of our concept is depicted in Figure 3. The proposed approach for the verification of safety-critical automotive functions is integrated with the V-model, which is the standard system development process also used in the automotive industry. To illustrate our concept, a simple function, called FM, is used as an example. The function has two inputs in_1 and in_2, and computes one output parameter out. Based on the input in_1, the function under analysis estimates the parameter $x_{Estimated}$, which is then checked for plausibility. Based on the result of the plausibility check $x_{Plausible}$ and on the input in_2 the function FM computes the final result out. The architecture of the function under analysis depicts five modules M1 to M5, which are used for the realization of the function's computations. The five modules are connected to each other through their respective interfaces. The concept shows based on this example function the steps necessary for the verification of safety-critical automotive functions. These steps represent intuitively the respective development phases of the V-model.

Requirements Analysis. In this phase, the standards and norms as well as the statutory regulations with which the automotive function under analysis must comply are analyzed. The goal of this phase is to derive the system-level safety requirements as well as the assumptions and the physical constraints under which the function is supposed to operate. For the example function in Figure 3, the system-level safety requirement is formulated on the computation result out, as shown in (1):

$$0 \leq out \leq 10. \tag{1}$$

There are several assumptions placed on the inputs of the function under analysis, shown in (2) and (3):

$$P(-1 \leq in_1 \leq 10) = 0.75 \tag{2}$$
$$-2 \leq in_2 \leq 2, \tag{3}$$

meaning that the input in_1 is situated in the interval [-1, 10] with a probability of 0.75 and a uniform probability distribution, while the input parameter in_2 takes values in the interval [-2, 2]. Furthermore, there are lower and upper bounds defined for the estimated parameters $x_{Estimated}$ and $x_{Plausible}$:

$$x_{Min} = -20, x_{Max} = 20. \tag{4}$$

These bounds are understood as physical constraints enforced by the laws of physics or due to the chosen vehicle configuration.

Construction of the Abstract System Model. The verification process is not carried out directly on the implementation of the investigated automotive function. Instead, a functional abstraction of the function's implementation is created using mathematical functions, which can work on infinite domains. Stream processing functions, introduced by Broy [6], can receive as inputs and produce as outputs infinite data streams. For these reason we consider stream-processing functions to be an appropriate computational model for the description of the automotive function's abstraction. Each of the modules in the function under analysis as well as the entire function are modeled as stream processing functions as defined in [6]. Figure 1 gives a visual intuition of this notion, using the mathematical function $f(x) = x^2$ as an example.

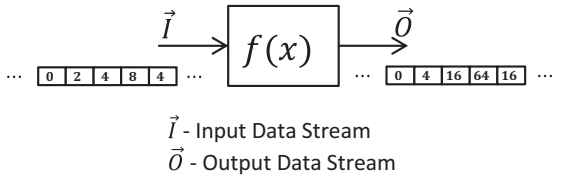

Figure 1. Visual intuition of stream processing functions. In this example, the function f(x) = x^2.

Time is a further issue addressed in the abstract system model of the function under analysis. Automotive functions are often composed of modules, which have different clock cycles. In order to simplify the verification of the function under analysis, its abstract model introduces a representation of time as a sequence of system states (see Figure 2). Such an approach guarantees that different clock periods of the modules in the function can be handled in a uniform manner.

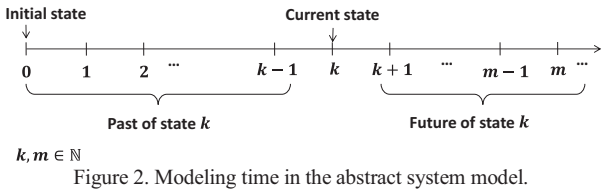

Figure 2. Modeling time in the abstract system model.

The final concern in the modeling process of the automotive function is finding a consistent systematic which serves as a guiding principle in this process. Our systematic consists of six parts, which is carried out for each of the function's modules as well as for the entire function under analysis: (1) data type definitions for function's inputs, (2) data type definitions for function's outputs, (3) data type definitions for function's intermediary results, (4) definitions of function's application parameters, (5) declaration or signature of the corresponding abstract mathematical function, and (6) definition of the abstract mathematical function. The defined data types built an abstract data type system, with operations, which abide by mathematical laws, independently of how the concrete data types are implemented in the automotive function and represented on the target system. Notice that the declaration and the definition of an abstract mathematical function are the same notions as in programming. Thus, the declaration of a function is in fact its signature, while the definition of the function constitutes its block in which the function's computation steps are described.

In order to demonstrate how our systematics works, we apply it on the automotive function FM, introduced as an example in Figure 3. The data type definitions for the inputs in_1 and in_2 and the output out are given in (5) – (7).

$$TIn_1 \stackrel{\text{def}}{=} \mathbb{R}, in_1 \in TIn_1 \quad (5)$$
$$TIn_2 \stackrel{\text{def}}{=} \mathbb{R}, in_2 \in TIn_2 \quad (6)$$
$$TOut \stackrel{\text{def}}{=} \mathbb{R}, out \in TOut. \quad (7)$$

The data types for the intermediary results $x_{Estimated}$ and $x_{Plausible}$ of the function FM are given in (8) and (9).

$$TEstimatedX \stackrel{\text{def}}{=} \mathbb{R}, x_{Estimated} \in TEstimatedX \quad (8)$$
$$TPlausibleX \stackrel{\text{def}}{=} \mathbb{R}, x_{Plausible} \in TPlausibleX. \quad (9)$$

Following the definition of the data types, (10) and (11) give the declaration or the signature of the corresponding mathematical function

$$FM: TIn_1 \times TIn_2 \rightarrow TOut \quad (10)$$
$$TOut\ FM(TIn_1\ in_1, TIn_2\ in_2), \quad (11)$$

while its definition is shown in (12) – (15):

$$FM(in_1, in_2) = out \quad (12)$$
$$out = M5(x_{Plausible}, in_2) \quad (13)$$
$$x_{Plausible} = \begin{cases} M2(x_{Estimated}), & x_{Min} < x_{Estimated} < x_{Max} \\ M3(x_{Estimated}), & x_{Estimated} \leq x_{Min} \\ M4(x_{Estimated}), & x_{Estimated} \geq x_{Max} \end{cases} \quad (14)$$
$$x_{Estimated} = M1(in_1). \quad (15)$$

Notice that the definition of the mathematical function FM is formulated through a \emph{walk back} from the function's output towards its inputs, which is similar to the notation in functional programming. The intermediary results described in the definition of the function FM are directly associated with the interfaces of the function's modules. Thus, the formal description of the abstract system model is intuitively linked to the architecture of the automotive function under analysis, which plays a central role later in the system verification test.

System Validation Test. The system validation test consists of road tests carried out with the test vehicle in the real physical environment. Besides testing the fully assembled test vehicle in the real environment, a further goal of the system validation test is to gather sensor data, which can be used to validate the assumptions derived during requirements analysis.

System Verification Test. The system verification test is realised through the hybrid approach proposed by this paper, which combines data-driven worst-case estimation with formal methods in order to verify the automotive function under analysis. The worst-case analysis is carried out on those

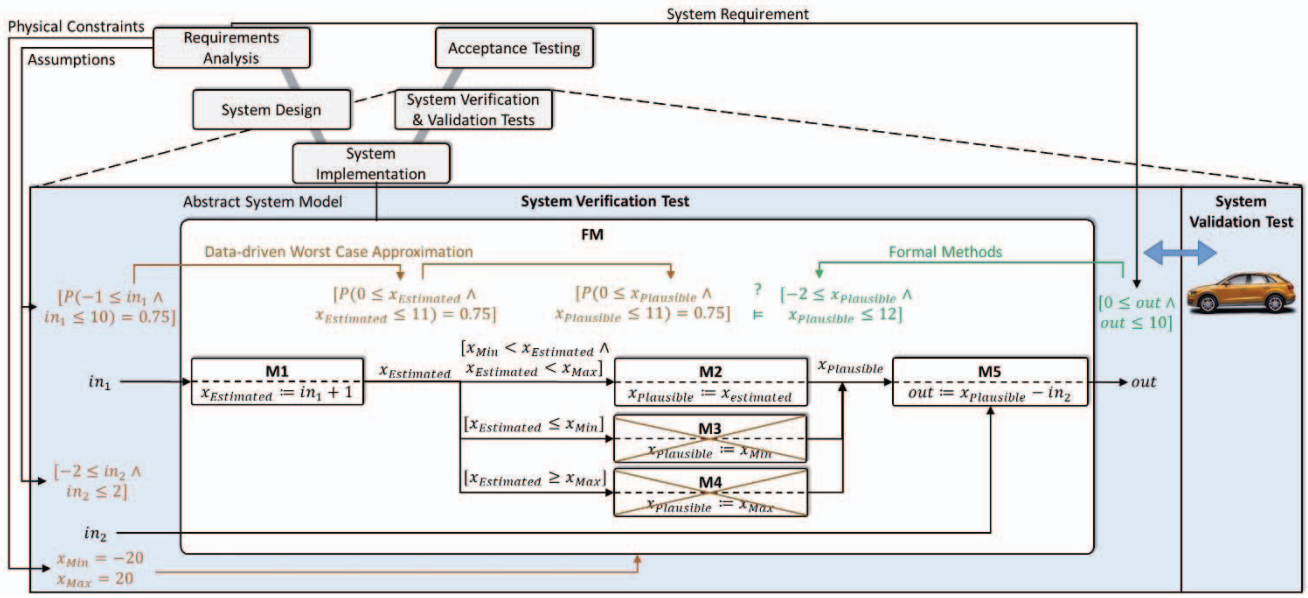

Figure 3. System development process enhanced with data-driven analysis and formal methods.

modules of the automotive function, which handle non-determinism and inputs with uncertain data quality, while formal verification methods are applied on the modules with deterministic behavior. Since the abstract model description is linked to the architecture of the function under analysis, the separation of the application of the data-driven approach and the formal methods approach at module level follows the modules' interfaces defined in the function's architecture. In the example function displayed in Figure 3 the data-driven analysis is carried out over the modules M1 to M4. Firstly, module M1 estimates the quantity $x_{Estimated}$ based on input in_1. As shown in (2), in_1 is situated in the interval [-1, 10] in 75% of the time with a uniform probability distribution. Applying the computation of module M1, it follows that the quantity $x_{Estimated}$ is in the interval [0, 11] with a probability of 0.75. Next, the estimated value is checked for plausibility against the physical constraints defined in the requirements analysis phase. Only the plausible values, which are strictly situated between the lower bound x_{Min} and the upper bound x_{Max} are taken into consideration, while all the other values are discarded. Thus, modules M3 and M4 are cut off from further computations of the function under analysis. Through the computation of the module M2, $x_{Plausible}$ is in the interval [0, 11] with a probability of 0.75.

The formal methods approach starts from the system requirement imposed on the output parameter out and is applied on the module M5. This is where the definition of the mathematical function FM from the function's output towards its inputs plays a decisive role. It allows the usage of text substitution in a similar way to the Hoare calculus [16], in order to derive the requirement for $x_{Plausible}$. Starting with the system-level requirement in (1), we apply text substitution and obtain the implication in (16):

$$0 \leq out \leq 10 \overset{M5}{\Rightarrow} 0 \leq x_{Plausible} - in_2 \leq 10 \overset{+in_2}{\Longrightarrow}$$

$$in_2 \leq x_{Plausible} \leq in_2 + 10 \overset{(3)}{\Rightarrow} -2 \leq x_{Plausible} \leq 12. \quad (16)$$

In this way, we have derived the requirement for $x_{Plausible}$, which can be checked against the worst-case estimation delivered by the data-driven approach. It is fairly easy to observe that for $x_{Plausible}$ the estimated interval is a subset of the required value interval: $[0, 10] \subset [-2, 12]$, meaning that the worst-case estimation for $x_{Plausible}$ satisfies its derived requirement. Due to the uncertainty in the estimation expressed through the probability distribution, the derived requirement, and by transitivity, the system-level requirement are satisfied with a confidence level of 75%.

Notice that the method of text substitution can be applied manually to fairly simple examples of software modules. However, in the automotive domain, system functions are often composed of a large number of modules, sometimes with several hierarchy levels in their architecture. Although the modules selected for the formal methods approach may have deterministic behavior, they can still be very complex so that it is not feasible anymore to apply formal methods manually. Instead, an automated verification method is employed. For this purpose, the corresponding mathematical functions are translated into the suitable modeling language of the verification tool of choice, thus building a stand-alone system model. The system-level requirements are formulated into the specification language that is accepted by the verification tool. The results of the data-driven estimation are taken as assumptions for the modules under formal verification. The goal of formal verification is to answer the question whether the system model satisfies the given system requirement under the estimated assumptions.

IV. Case Study 1: Automotive Function for Speed Estimation

The first case study is based on an industrial automotive function for speed estimation. From the perspective of

technical feasibility, precise measurements of a vehicle's speed are already possible with high-quality reference measurement systems, e.g. inertial measurement units (IMU) or differential GPS (D-GPS). However, the usage of such measurement systems is expensive for the automotive original equipment manufacturers (OEMs), and if mounted in series vehicles, cost-prohibitive for their end users. Therefore, in commercialized series vehicles the speed estimation is performed with an accepted tolerance level, without using a reference measurement system. Vehicle speed estimation is a safety-critical automotive function, since the estimated speed is used as input to other driving assistance systems relevant for the vehicle's safety, e.g. adaptive cruise control (ACC) or speedometer. Recent algorithms for the vehicle speed estimation strongly rely on vehicle's wheel speed, which can be measured with onboard sensors. However, for the calculation of the vehicle speed, the estimation algorithm also requires the tire circumference of the vehicle. The vehicle's tires have diverse deformations in different driving situations, due to the inherent elasticity of rubber. Furthermore, this deformation is influenced by other non-deterministic environmental factors, e.g. the temperature in the physical environment or the road conditions, making it impossible to obtain an exact value for the tire circumference of a driving vehicle. Thus, a precise approximation of the tire circumference becomes a fundamental point for the vehicle speed estimation. Automotive engineers have patented diverse approaches to estimate the vehicle's speed as precise as possible (cf. [20], [12], [26], [21]). In the course of this case study, an automotive OEM partner granted us access to their speed estimation function. The function was designed with the goal to fulfill relevant requirements defined in the European New Car Assessment Program (Euro NCAP). In this function, a redundant GPS system delivers a reference vehicle speed. This reference speed has potentially a large deviation to the ground truth speed, which can get exponentially higher due to strong temporal dependencies, especially in the cases with poor reception of GPS signal. For these reasons, the reference speed is not used directly for the speed display on the vehicle's instrument board. Instead, an estimation of a reference value for the tire circumference is carried out. The reference tire circumference is then integrated with an approximation of the tire circumference carried out by additional mechanisms, which rely on assumptions made about various sensor inputs that are used depending on different situations. Due to the non-disclosure agreement (NDA) signed with the OEM partner, further details about this function are excluded from this paper. The OEM partner provided us with the implementation of the speed estimation function itself and gave us access to a large amount of test data collected in field tests with the respective vehicle sensors used in the speed estimation function.

Requirements Analysis. Speedometers have been integrated in the car at the beginning of the 20th century. The role of the speedometer is to indicate the instantaneous speed of the car in miles per hour (mph), kilometers per hour (kmh), or both. The initial European regulation [25] has imposed the following requirement for the deviation between the visualized speed on the speedometer and the real ground truth speed: $0 \leq v_{display} - v_{real} \leq 0.1 * v_{real} + 4\ kmh$ under the assumption that $40 \leq v_{real} \leq 120\ kmh$, where $v_{display}$ is the speed displayed on the dashboard of the ego-vehicle, and v_{real} is the actual vehicle speed. A more restrictive regulation has been passed in the Euro NCAP program[1], which requires for new vehicles that the speed estimation function satisfies the requirement in (17):

$$0 \leq v_{display} - v_{real} \leq 5\ kmh, \tag{17}$$

under the assumption that

$$50 \leq v_{real} \leq 120\ kmh. \tag{18}$$

The lower bound of the deviation between $v_{display}$ and v_{real} is set to zero, especially due to the safety reasons. This means that the visualized vehicle speed should never undercut the lower bound, since the driver or the assistance system might decide a too conservative braking strategy, possibly leading to collision risk, once the displayed speed is lower than the ground truth. The Euro NCAP requirement in (17) is considered the system-level requirement of the speed estimation function. Further assumptions have been formulated with respect to the quality and the availability of the GPS signal and of the wheel speed data, the maximum tire deformation per second as well as the error deviation due to the quantization on the CAN bus. The physical constraints with respect to the estimated tire circumference have been derived from the configuration of the target vehicle. Concrete examples of assumptions and physical constraints are prohibited from publication by the signed NDA.

Construction of the Abstract System Model. The implementation of the speed estimation function has been realized in MATLAB/SIMULINK by the OEM partner. The functionality of the function has been implemented in five large modules which in turn contained 13 submodules situated on four hierarchy levels. The abstract system model of the speed estimation function is divided into four large modules, which are responsible for (1) estimation of tire circumference, (2) plausibility check of the tire circumference, (3) computation of the vehicle speed, and (4) post processing of the computed vehicle speed. The model has 14 input parameters, 24 application parameters and one output parameter. Further details about the abstract system model are excluded from this paper. However, a high-level overview of the speed estimation algorithm is given in Figure 4.

System Validation Test. The test data has been collected with the wheel speed and GPS sensors used by the speed estimation function in road tests carried out by the OEM partner. Based on an analysis of the test data, we were able to

[1] https://www.euroncap.com/en/for-engineers/protocols/safety-assist/

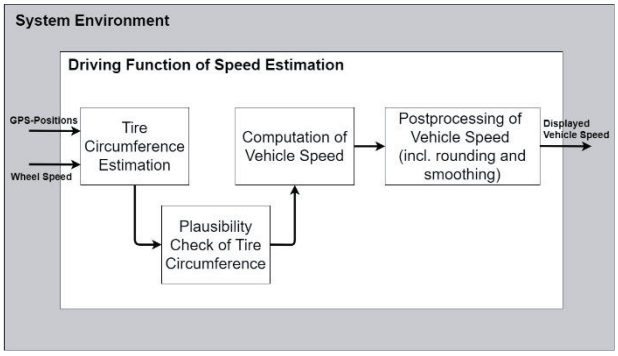

Figure 4. Abstract overview of the speed estimation algorithm (cf. [1]).

validate the assumptions and constraints formulated in the requirements analysis phase.

System Verification Test. The system verification test consists of the data-driven worst-case estimation of the tire circumference and the formal methods approach for the derivation of requirements for the estimated tire circumference using the system-level requirement. The worst-case estimation of the tire circumference is carried out based on the assumptions made in the requirements analysis phase. The requirement for the estimated tire circumference is derived using the text substitution method introduced in Section 3. The concept was applied iteratively on several versions of the MATLAB/SIMULINK model of the speed estimation function. The results of the system verification test have shown that a first version of the speed estimation function did not satisfy the Euro NCAP requirement. However, the verification test results obtained on its first MATLAB/SIMULINK model allowed us to make suggestions for improvements of the speed estimation function. These improvements focus on limiting the effect of non-deterministic factors, e.g. quality of GPS signal, on the estimation result by constraining the respective factor within a certain value range through assumptions made during the development process. The system engineers of the OEM project partner took these suggestions and, based on them, created a second version of the speed estimation function. The improved model was again verified with the concept proposed in Section 3, the results showing that the improved function satisfies the system-level requirement with a confidence level of 95%. The part of the abstract system model consisting of deterministic modules has also been checked with the model checker STORM against the system-level requirement. For this purpose, this part of the abstract model has been translated into the PRISM language and the system-level requirement has been formalized in probabilistic computational-tree logic (PCTL). Although PRISM is an expressive modeling language, it does not support all concepts used in software development, e.g. loops and division. In order to implement these components, a preprocessor is developed in Python, which reads the preformatted model with specific placeholders as an input file and, depending on the application parameters, generates a complete, verifiable PRISM model. Further constraints are made on the input parameters of the model as well as on their discretization step in order to make the PRISM model verifiable. The verification results showed that the constrained abstract system model satisfies the system-level requirement.

V. CASE STUDY 2: AUTOMOTIVE FUNCTION FOR EHAUST AFTERTREATMENT

The second case study of this paper is based on an automotive system function provided by our automotive OEM partner. This function is part of the overall exhaust aftertreatment system in diesel engines, which controls the returning of a portion of the exhaust gases back to the burning process, in order to reduce the overall exhaust production of the vehicle. An abstract overview of the automotive function located in the low-pressure part of the exhaust aftertreatment system is shown in Figure 5.

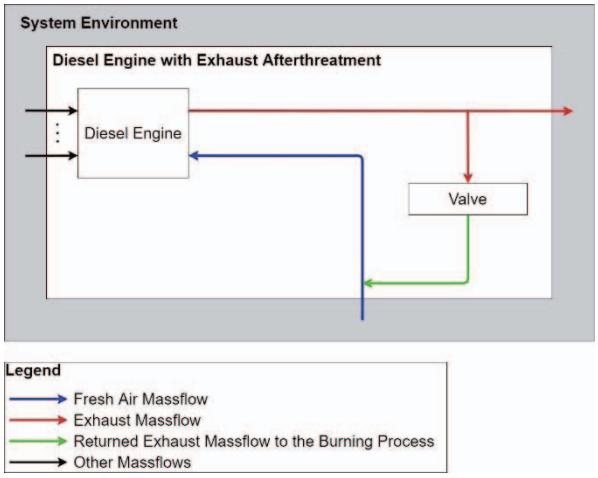

Figure 5. Abstract overview of the low pressure exhaust aftertreatment subsystem (cf. [28]).

In general, engines in older vehicles use fresh air to burn fuel in the engine and return all of the exhaust to the environment (see Figure 5). However, this is bound to violate laws and regulations regarding environment protection. In contrast, modern engines return a portion of exhaust to the burning process, so that the vehicle's overall production of exhaust gases is reduced. Despite a portion of the exhaust gases being returned to the burning process, the engine still has to deliver the expected power during driving and efficiently burn fuel. In order to keep the burning process efficient, a software-controlled valve is used to deliver an optimal ratio between fresh airflow and the returned exhaust gas flow.

Since we had no access to other technical components communicating with the function under investigation, the focus of this case study lied solely on the formal verification of the software used to control the valve. The software is an ASCET[2] model organized in five modules, each with several hierarchical layers, and using block diagrams and C code. The ASCET model is executed cyclically to convert a potentially infinite input stream to an output. The model calculates a set value as a pressure instance p_{set} based on an actual value

[2] https://www.etas.com/de/

p_{actual} and several other inputs for the valve, which regulates the returning of the mass flow of exhaust gas to the mass flow of air. Due to its direct impact on environmental health, this automotive function is considered as safety-critical.

Requirements Analysis. To start the verification process, the system-level requirement needs to be formalized. We received an informal requirement based on expert-knowledge of the OEM project partner. The informal requirement specifies, that *the offset between the actual value* p_{actual} *and the set value* p_{set} *shall not differ by more than* $\pm x$. Based on the information received from the experts, this is a safety-critical requirement since noncompliance with it reduces the percent of exhaust gas returned to the burning process. This leads on short term to violation of environment protection laws and on long term may cause imbalance to natural ecosystems and severe health issues for humans. The informal safety requirement is then analyzed and refined into a semiformal requirement, which requests that the distance between p_{actual} and p_{set} shall not be greater than $x \in \mathbb{R}_+$. The mathematical expression for the distance between p_{actual} and p_{set} is $|p_{actual} - p_{set}|$. A visual intuition of this semiformal requirement is shown in Figure .

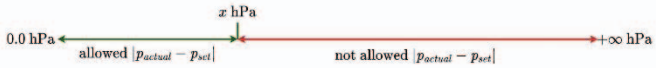

Figure 6. Requirement from the viewpoint of the distance between $\boldsymbol{p_{actual}}$ and $\boldsymbol{p_{set}}$ (cf. [28]).

Since the components of the system are running with equal clock periods, time is considered to be discrete, which makes the formal verification easier. Linear temporal logic (LTL) is a formal logic which uses a discrete representation of time (cf. [2]). The LTL requirement of the exhaust aftertreatment function is:

$$G\neg(|p_{actual} - p_{set}| > x). \tag{19}$$

Construction of the Abstract System Model. We created a mathematical abstraction M_S of the whole ASCET model. This mathematical model is built by applying the rules proposed in Section 3. Several decisions were taken during the construction of the abstract model in order to reduce its complexity. For example, several software design conventions used by the OEM partner were left out, e.g. replacing inputs passed by array elements with scalar variables. The data type domains have been defined based on the system documentation and expert knowledge, which also includes specific data quantization factors. Once the abstract system model was finished, we noticed that the formal requirement in (19) refers only to one submodule M_S', which has just one hierarchy level in the abstract model, and with 14 input arguments and one output argument in total. Therefore, we simplified further our abstract model, overapproximating the results of the other four submodules, which served as inputs to module M_S', to reduce the overall space for formal verification.

System Validation Test. In this case study, we had no access to test data collected via vehicle sensors as in the first case study. Instead, the OEM partner provided us with several characteristic curves for the valve pressure.

System Verification Test. For a fully automated formal verification, we used KIND2 [8] which is a modern SMT-based model checker using the synchronous programming language LUSTRE [15] to specify the system and COCOSPEC-contracts [7] to express formal requirements. Due to LUSTRE's functional way to describe systems, the system model M_S' could be easily specified without losing the connection to the underlying abstract mathematical description. The formal requirement in Equation (19) was transformed into a guarantee-specification by leaving out the G (*Globally)* operator. The data type definitions of the mathematical abstraction provided us with the input domains for assumptions, expressed in LUSTRE language via `assumes`-instructions.

Even with the overapproximation of the other four submodules, KIND2 was not able to find a solution, due to the non-linear arithmetic in the module M_S'. To make the system verifiable, we cut out the part with non-linear arithmetic using the rules introduced in Section 3. During this process of model simplification, we analyzed a limiter with dynamic limits right before the output, which can be mathematically described as follows:

$$limiter(x, max, min) = Max(Min(x, max), min), \tag{20}$$

with $x, min, \max \in \mathbb{R}$.

The analysis of the limiter function is depicted in Table 1. This table presents all possible use cases, enumerating the considered inputs and the total result of the composition of the Min- and Max-operator. In four of six cases the outcome will be min. Further calculations of min do not result in non-linear arithmetic, therefore we verify this dataflow and consider it as M_S'', which has just two inputs parameters and one output. Additionally, we added functions for the quantization of inputs and outputs based on the mathematical description and functions that model the sensor errors within specific value ranges.

TABLE 1. POSSIBLE DATAFLOWS ON THE LIMITER (CF. [28])

#	Case	$\boldsymbol{Max(Min(x, max), min)}$
1	$x < max < min$	min
2	$x < min < max$	min
3	$max < x < min$	min
4	$max < min < x$	min
5	$min < x < max$	x
6	$min < max < x$	max

The result of the formal verification of using KIND2 is shown in Table 2. We had four different characteristic curves, with different application parameters, one of which was provided by the OEM partner. We used three setup configurations for our evaluation: (1) no quantization errors and no measurement errors, (2) only quantization errors, and (3) quantization errors and measurement errors. The time KIND2 needs for delivering a result is also presented. Only the fourth characteristic curve was verified successfully under consideration of quantization errors and measurement errors (see experiment 12 in Table 2). The time it takes KIND2 to

deliver the verification result is quite low, with the experiment 12 as the most complex task needing 29.696 s to finish the verification.

TABLE 2: RESULTS OF THE FORMAL VERIFICATION OF M_S'' USING KIND2.

Exp.	Setup	Quanti-zation	Meas. Error	Result	Duration
1	1	No	No	Falsifiable	0.144
2	2	No	No	Correct	0.225
3	3	No	No	Correct	0.268
4	4	No	No	Correct	0.479
5	1	Yes	No	Falsifiable	0.176
6	2	Yes	No	Falsifiable	0.297
7	3	Yes	No	Correct	17.776
8	4	Yes	No	Correct	19.301
9	1	Yes	Yes	Falsifiable	0.237
10	2	Yes	Yes	Falsifiable	1.883
11	3	Yes	Yes	Falsifiable	1.184
12	4	Yes	Yes	Correct	29.696

VI. DISCUSSION AND LESSONS LEARNED

Due to the complexity of the abstract model of the speed estimation function, it was not possible to cover the original value range for all input parameters during automated verification. Instead, the lower and upper bounds defined for the parameter ranges originally through application parameters were adapted, and thus, more restricted versions of the model have been created, albeit without losing the characteristics of the abstract model. These constrained versions of the speed estimation's abstract model were successfully verified. The preprocessor allows setting up different application parameters and discretization ratios so that verifiable, albeit constrained, versions of the abstract model were obtained. Even if the restricted versions of the abstract model were successfully verified, the automated verification took up a lot of resources, e.g. one restricted version of the abstract model took 6898 s CPU time and 33.33 GB RAM space to verify successfully the system-level requirement of the speed estimation function. This is because the stochastic inputs and non-determinism were considered important and were retained even in the constrained version of the speed estimation abstract model. Although the initial abstract model of the speed estimation function could not be checked via automated verification in a single iteration due to its complexity, the verification results obtained on the constrained PRISM models are still applicable on the unconstrained model. A constrained PRISM model can be automatically generated with the Python preprocessor, on the basis of a *model input configuration*. The model input configuration is derived from the constraints imposed on the input parameters of the unconstrained abstract system model. The model input configurations of the constrained models can be defined in such a way that the value ranges of the input parameters in the constrained models cover the value range of the respective input parameters in the unconstrained PRISM model. Should all the constrained models be successfully verified against the system-level requirement, then it can be said that the unconstrained abstract model also satisfies this requirement.

Proving correctness of functions in safety-critical domains like automotive is a hard task due to the undecidable verification problem, especially in the case of fully automated formal verification, e.g. model checking. Automated verification tools have in general difficulties in handling non-linear arithmetic, which is often present in automotive software. Even if verification is possible, the environment model of the function under investigation must be taken into account. Often, probabilistic inputs and non-determinism are additional factors inherent to the function's environment which increase the size of the state space and impact the decision on which verification tool to use. Broman et al. [5] introduce an approach to help system and verification engineers choose the appropriate verification tool based on the considered viewpoints of the system under investigation.

The choice of the verification tool depends strongly on the viewpoints from which the system is analyzed, e.g. the necessity to model the stochastic inputs and the non-deterministic behavior of the speed estimation function led to the choice of the probabilistic model checker STORM.

The proposed hybrid approach is driven by the functional architecture of the system under analysis. Depending on the types of modules in the system, the verification engineers must find a balanced solution between the data-driven and formal methods. Data-driven methods allow the elimination of nondeterministic factors through worst-case estimation as deterministic. Nevertheless, data-driven methods work on the presumption that enough input data is available, that is the assumptions made with respect to the probability distributions of the input data are fulfilled. In case these assumptions are not satisfied, because e.g. the necessary data cannot be measured, then the issue posed for the formal verification by the nondeterministic factors remains. In this case, the investigation scope of the abstract model's behavior can be reduced through overapproximation. Within the reduced investigation scope, the formal verification provides the mathematical proof to the question whether the abstract model satisfies the system-level requirement. The proof of correctness can be further used by the OEM's system engineers to construct a safety argument for the purpose of the automotive system certification and approval process.

In both case studies, the hybrid verification approach provided additional information to the systems engineers of the OEM partner with respect to the inner workings of the automotive function under analysis and to possible design improvements. In the speed estimation function, the application of the proposed approach on different version of the MATLAB/SIMULINK allowed us to make suggestions for improvements in the design of the speed estimation function. Based on these suggestions, the OEM was able to make the appropriate settings in the vehicle so that the speed estimation function satisfies the Euro NCAP requirements. The insights gained with the help of the help of the hybrid verification approach allowed us to develop an alternative concept for the vehicle speed estimation to benchmark the verified approach in this case study. Our alternative concept has been published in a previous paper [1]. The exhaust aftertreatment function

has become increasingly complex over the years through a large number of development iterations. This complexity hindered the OEM's system engineers to obtain an informed overview of this automotive function. Our approach helped the system engineers to gain a better understanding of the inner workings of the exhaust aftertreatment function. Moreover, this approach allowed in both case studies the finding of appropriate test cases and application parameters which can be further used in the development and improvement of the automotive function.

VII. Conclusion and Future Work

This work proposed a hybrid approach, oriented along the system development process and driven by the functional system architecture, which brings together data-based estimation techniques to retain statistical significance and formal methods to ensure correctness of the function under investigation with respect to system-level requirements. It uses worst case approximations from the side of the system input parameters and Hoare-calculus like substitutions and automated formal verification from the side of the system-level requirements. The system is split at the right place in such a way that the verification is made simpler without losing expressiveness.

For further work, we are looking at expanding the verification in the second case study. We have verified one dataflow completely, leaving out two data flows to be checked in the future. Additionally, these data flows can be broken down into more data flows, which can be verified separately with different verification tools. An additional future work is the automatization of the worst-case approximation and the text substitutions, in order to avoid mistakes. Both part can be implemented in a toolchain, in order to keep the system model as consistent as possible to the original system and guide verification engineers through our approach. The assembled mathematical abstraction of the system can be represented in a well-defined computer readable manner, so that it can be used as an intermediate description for a verification tool. This intermediate description can then be automatically translated with an adequate compiler into the system description language of the desired verification tool. This allows verification engineers to maintain the abstract system model consistent to the original system and to switch easily between verification tools, if the selected one is not able to deliver a result within a finite time window.

[1] Aniculaesei A, Zhang M, Rausch A. Data-driven Approach for Accurate Estimation and Validation of the Ego-Vehicle Speed. In: Knieke C, Mansouri M, Telleschi G, editors. ICONS 2020: The Fifteenth International Conference on Systems: IARIA, 2020, pp. 72–77.

[2] Baier C, Katoen JP. Principles of Model Checking. Cambridge, Massachusetts: The MIT Press, 2008.

[3] Bengtsson J, Larsen KG, Larsson F, Pettersson P, Yi W. Uppaal in 1995. In: Margaria-Steffen T, Steffen B, editors. Tools and algorithms for the construction and analysis of systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 431–434.

[4] Blanchette JC, Bulwahn L, Nipkow T. Automatic Proof and Disproof in Isabelle/HOL. In: Tinelli C, Sofronie-Stokkermans V, editors. Frontiers of combining systems. Heidelberg: Springer, 2011, pp. 12–27.

[5] Broman D, Lee EA, Tripakis S, Törngren M. Viewpoints, formalisms, languages, and tools for cyber-physical systems. In: Hardebolle C, Syriani E, Sprinkle J, Mészáros T, editors. Proceedings of the 6th International Workshop on Multi-Paradigm Modeling (MPM'12). New York, NY, USA: ACM, 2012?, pp. 49–54.

[6] Broy M. Functional Specification of Time Sensitive Communicating Systems. ACM Trans. Softw. Eng. Methodol. 1993;2(1):1–43.

[7] Champion A, Gurfinkel A, Kahsai T, Tinelli C. CoCoSpec: A Mode-Aware Contract Language for Reactive Systems. In: Nicola R de, Kühn E, editors. Software Engineering and Formal Methods. Cham: Springer International Publishing, 2016, pp. 347–366.

[8] Champion A, Mebsout A, Sticksel C, Tinelli C. The Kind 2 Model Checker. In: Chaudhuri S, Farzan A, editors. Computer Aided Verification. Cham: Springer International Publishing, 2016, pp. 510–517.

[9] Clarke EM, Henzinger TA, Veith H. Introduction to Model Checking. In: Clarke EM, Henzinger TA, Veith H, Bloem R, editors. Handbook of Model Checking. Cham: Springer International Publishing, 2018.

[10] Dehnert C, Junges S, Katoen J-P, Volk M. A storm is Coming: A Modern Probabilistic Model Checker. Available at: https://arxiv.org/pdf/1702.04311.

[11] Dijkstra EW. The humble programmer. Communications of ACM 1972;15(10):859–66.

[12] Dittrich H, Gärtner V, Rinck R, Wehren V. Method for determining a vehicle reference speed(DE10254628A1), 2004.

[13] Gao Z, Cecati C, Ding S. A Survey of Fault Diagnosis and Fault-Tolerant Techniques Part II: Fault Diagnosis with Knowledge-Based and Hybrid/Active Approaches. IEEE Transactions on Industrial Electronics 2015:1.

[14] Gao Z, Cecati C, Ding SX. A Survey of Fault Diagnosis and Fault-Tolerant Techniques - -Part I: Fault Diagnosis With Model-Based and Signal-Based Approaches. IEEE Transactions on Industrial Electronics 2015;62(6):3757 - 67.

[15] Halbwachs N, Caspi P, Raymond P, Pilaud D. The synchronous data flow programming language LUSTRE. In: Proceedings of the IEEE, 1991, pp. 1305–1320.

[16] Hoare CAR. An axiomatic basis for computer programming. Communications of the ACM 1969;12(10):576–80.

[17] International Organization for Standardization. ISO 26262-10:2011: Road vehicles - Functional safety: Part 10: Guideline to ISO 26262. Geneva, Switzerland: ISO, 2011.

[18] International Organization for Standardization. ISO 26262-6:2011: Road vehicles - Functional safety: Part 6: Product development at the software level. Geneva, Switzerland: ISO, 2011.

[19] Kia SH, Henao H, Capolino G-A. Survey of real-time fault diagnosis techniques for electromechanical systems. In: Staff I, editor. 2017 IEEE Workshop on Electrical Machines Design, Control and Diagnosis (WEMDCD). Piscataway: IEEE, 2017, pp. 290–297.

[20] Kost F. Speed Estimation Process(EP0495030A1), 1994.

[21] Kruse ST, Wagstaff D, Palmer J. Systems and methods for managing speed thresholds for vehicles(US10683017B1), 2020.

[22] Kwiatkowska M, Norman G, Parker D. PRISM: Probabilistic Symbolic Model Checker. In: Field AJ, editor. Computer performance evaluation. Berlin and London: Springer, 2002, pp. 200–204.

[23] Mohammadpour J, Franchek M, Grigoriadis K. A survey on diagnostic methods for automotive engines. International Journal of Engine Research 2012;13(1):41–64.
[24] Mouzakitis A. Classification of Fault Diagnosis Methods for Control Systems. Measurement and Control 2013;46(10):303–8.
[25] Richtlinie 75/443/EWG des Rates vom 26. Juni 1975 zur Angleichung der Rechtsvorschriften der Mitgliedstaaten über den Rückwärtsgang und das Geschwindigkeitsmeßgerät in Kraftfahrzeugen, 1975.
[26] Schmitt DP, Hasegawa A, Lachmayr S. Systems and methods for determining a speed limit violation(US9536426B2), 2017.
[27] Visser W, Havelund K, Brat G, Park S, Lerda F. Model Checking Programs. Automated Software Engineering 2003;10(2):203–32.
[28] Vorwald A. Formale Verifikation von Reaktiven System am Beispiel einer Fahrfunktion. M.Sc. Clausthal-Zellerfeld, Germany, 2020.