Qualitative Monitors based on the Connected Dependability Cage Approach

Felix Helsch*, Iqra Aslam†, Abhishek Buragohain‡, and Andreas Rausch§
Institute for Software and Systems Engineering
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
Email: { felix.helsch*, iqra.aslam†, abhishek.buragohain‡, andreas.rausch§ } @tu-clausthal.de

***Abstract*—Many Autonomous Systems (ASs) have been widely applied in safety-critical applications like driverless taxis and financial credit assessment. Due to the integration of machine learning techniques for functions like environmental perception, ASs are nowadays a hybrid construction combined with classical engineered and Artificial Intelligence (AI-) based subsystems. Such a construction of the hybrid AI-based AS makes it impossible for engineers to guarantee the dependability requirements during development, since the system cannot be completely tested and formally verified. Addressing these dependability issues of the hybrid AI-based AS, this paper provides a transparent overview of our Dependability Cage approach on different levels of abstraction. In particular onboard continuous monitoring is combined with remote technical supervision for human intervention in our approach for Runtime System Observation and Resilience System Stabilization, forming a *Connected* Dependability Cage. The Qualitative Monitor for observing the system's functional correctness at runtime is chosen as an implementation example and is evaluated in the concrete use case of the research project "VanAssist" which focuses on using AVs for package delivery in urban areas.**

Keywords—Dependable Autonomous System; Connected Dependability Cage; Runtime System Observation; Resilience System Stabilization; Qualitative Monitoring

I. INTRODUCTION

Autonomous Systems (ASs) have recently achieved success in many application domains, including automated vehicles (AVs), smart home systems, and autonomous financial agents. They are getting increasingly useful and beneficial for us. As a side effect, we as the users rely on the services of such systems increasingly, even in safety-critical applications such as driverless taxis and financial credit assessment [1] [2].

Many recent improvements in the performance of AS are made by using machine learning techniques [3]. Such a system design makes AS nowadays become hybrid Artificial Intelligence (AI-) based systems, consisting of classical engineered subsystems and machine-learned subsystems based on Artificial Intelligence (AI) techniques. Automated vehicles as an example, the AV's perception is mainly realized based on AI and the feedback control is designed as classical engineered subsystem. Both system parts are integrated on an AV to perform the expected safety-critical tasks.

A. Motivation

Considering the engineering perspective, there are various differences between the classical and the AI-based systems [4]. The classical engineering process for safety critical applications starts with a (semi-)formal requirements specification that must be complete and correct. While this idealized process is rarely realized to its full extend, the requirements specification is later used as main input for the system's testing and verification. For the development of an AI-based system, a huge data collection is used instead of the requirements specification. Different from the completeness and correctness of the requirements specification, the data collection is incomplete and may even contain a small percentage of incorrect data samples.

Nevertheless, these AI-based systems are widely applied for the fulfillment of safety-critical tasks. Product liability regulations impose high standards on manufacturers regarding the safe operation of such systems [5]. Against such a background, established engineering methods are no longer adequate to guarantee the dependability requirements (safety, security and privacy) in a cost-efficient way due to significant limitations. For instance, they are not able to handle the specific aspects of AI-based systems, as discussed in [1]. Thus, engineers cannot completely test and verify AS during development to fully guarantee the dependability requirements.

In the development of AI-based systems like the AV's perception, engineers use labeled training data and machine learning techniques to train an interpretation function. For illustration, a simplified perception task that classifies the traffic signs to the corresponding semantic classes is shown in Figure 1. For this purpose, a machine learned interpretation function needs to be trained using the training data to map a finite set of input data (e.g., traffic sign images) to the correct output information (e.g., traffic sign classes). Machine learning abstracts from the given training data examples and produces the machine learned function that is able to process an infinite number of different images. Thus, the resulting machine learned interpretation function can map any kind of image taken by the camera in a real environment to one of the known traffic sign classes. Considering the AS's operation from a safety perspective, an essential question is whether the produced output information of the machine learned interpretation function $if_{ml}(x)$, processing the current input data x, is sufficiently reliable to be safe.

Different from the AI-based systems, classical engineered systems are developed by using a (semi-)formal requirements

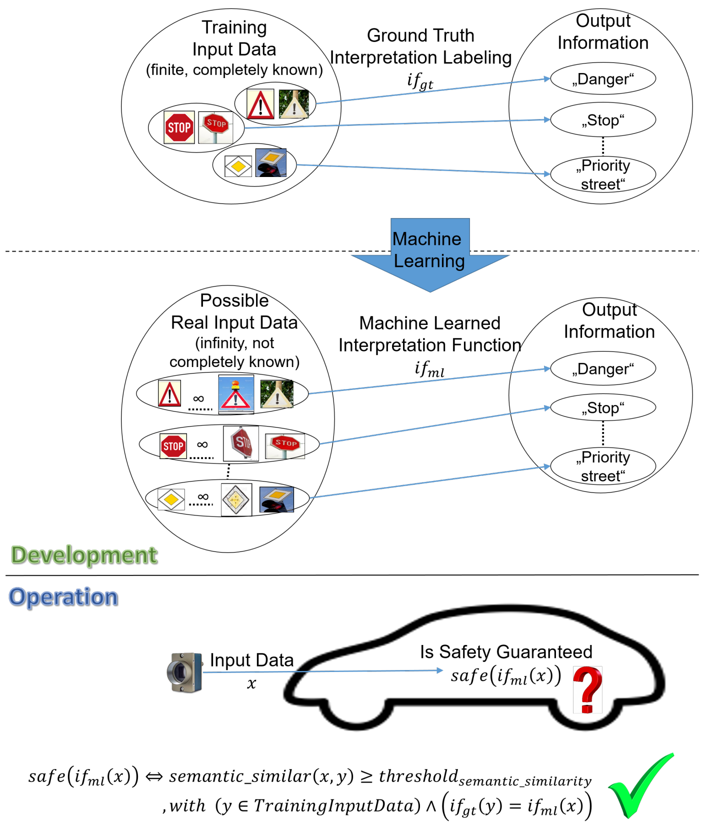


Figure 1. Check of Dependability Requirements for AI-based Systems. [6]

specification to describe the systems' behaviors, as introduced before. In this case, the behaviors must conform to the specification and thus it can be verified if the system meets the requirements. Such engineered behaviors are desirable for most types of systems like information and cyber-physical systems. However, the classical engineering approach is not applicable for AI-based ASs due to the use of a huge data collection instead of the (semi-)formal requirements specification. Even if a requirements specification is available, AI-based systems are not only expected to adhere to their specification but also solve problems more effectively by acquiring new skills with their online learning capabilities. For this purpose, ASs need to learn from the experienced situations and accordingly adapt themselves. In addition, such an adaptation and learning capability is especially essential for ASs designed for highly complex tasks, since the problem domains of the automated driving cannot be fully specified during development due to their incredible high complexities. An analog example in our real life to get a rough estimation of the high complexities is that a human driver needs many years of driving experience to be able to assess potentially dangerous traffic situations.

In contrast to traditional systems, adaptive and learning systems also entail risks and challenges. While the classical system behavior is predictable and comprehensible, the behavior of adaptive and learning systems can possibly deviate from the behavior specified during the system development. For example, the Twitter bot TayAndYou was launched by Microsoft in 2016 as an experiment to communicate with people [7]. It was supposed to learn from the conversations with Twitter users and adapt itself accordingly. But the bot only learned to curse and scold other people. Such behavior were neither planned nor expected by the system designers.

Based on the fundamental concepts of adaptive and learning systems, it has to be accepted that, we cannot completely specify and predict the behavior of such kind of systems. Due to incompleteness of the specification and uncertainty of the operational environment, as discussed above, it is not possible to test, verify or validate these systems exhaustively during development. Thus, we need to identify new ways to monitor adaptive and learning systems, and develop standard procedures to verify their behaviors' correctness. To sum up, the core challenge is: How can we guarantee safe and secure behavior for all parts of an AI-based AS (e.g., complex functions, machine learned functions, sensors and actuators, and the whole system), if the system operates in an unknown environment and do behavioral changes due to online learning have an impact on dependability requirements during operation?

B. Previous and Related Work

In order to tackle the challenges of engineering dependable hybrid AI-based ASs the Dependability Cage concept was proposed [8] [9] [10] [11] [12]. Dependability Cages are derived by engineers from existing development artifacts. The derived Dependability Cages are then used both during the ASs development and operation to check the fulfillment of dependability requirements. This approach aims to give the users a transparent view of the confidence level.

The Quantitative Monitor as an essential part in the Dependability Cage concept was proposed in [6]. As illustrated in Figure 1, the Quantitative Monitor intuitively indicates whether the AS is currently processing actual real input data x, which are from a reliability perspective similar enough ($semantic\_similar(x, y) \geq threshold_{semantic\_similarity}$) to the (ground truth) training input data (y) used for machine learning techniques, so that the produced actual output information of the machine learned interpretation function $if_{ml}(x)$ can be assumed to be correct and safe. If this is not the case, the AI-based AS is possibly in an unsafe state.

In this case, providing a measure for the semantic similarity of input data that serves as an argument for the output information reliability is a challenge. Novelty detection to automatically identify new relevant test data differing from the available training data becomes an interesting approach to realize such a semantic similarity measure. One promising novelty detection approach using AI technique called autoencoder have been proposed [13].

The basic principle of autoencoder-based novelty detection was introduced by Japkowicz et al. [14]. In their approach, the autoencoder is trained to minimize the error between an input image and a reconstructed input image. Firstly, known images were used to train the autoencoder. After training, the autoencoder was fed with new images. If the difference between original and reconstructed image was higher than a given threshold value, the new image was classified as novel [15] [16].

However, a big structural difference in input data does not necessarily correlate with a different output information class. For instance, as illustrated in Figure 2, completely different traffic situations (output information class) frequently have similar images (input data). While the AV is free to pass the zebra crossing in Figure 2a, in Figure 2b it instead has to stop and let the pedestrians pass the crossing.

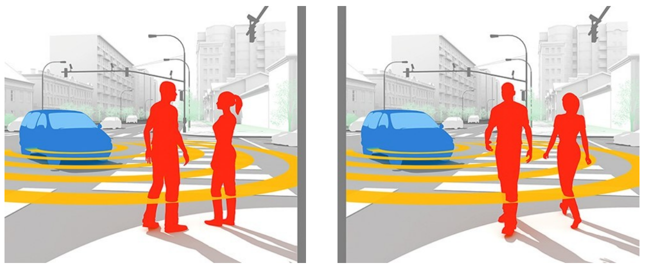

(a) AV can pass (b) AV has to wait

Figure 2. Similar camera images of pedestrians represent very different traffic situations [6] [17].

In the previous sections, different challenges with respect to the engineering of hybrid AI-based ASs were identified. A high level concept for these challenges, which we also applied to our Dependability Cage was set up in [8]: The identified challenges pointed out two types of risks that have to be considered through all development phases of the AS: (1) external risks due to the uncertainties in the system's real operation environment, and (2) internal risks caused by the system's changing behavior. In the aforementioned Dependability Cage concept, two categories of Dependability Cages were defined to safeguard against these risks: (a) Dependability Cages developed for the system and (b) Dependability Cages for the system's environment. In order to use these Dependability Cages, a distinction is made between several types of behaviors of the system and its environment both at development time and at operation time.

At development time a given AS and its environment are further differentiated into its engineered behavior and its tested behavior (cf. Figure 3 left).

Similarly to the development time, at operation time a differentiation is made between the real behavior and the observed behavior of a given AS and its environment (cf. Figure 3 right). The real behavior means the behavior at operation time which may differ from the engineered behavior. It contains an uncountable number of situations caused by different influencing factors. For an AS, the real behavior is based on the system adaptation to the constantly changing operational environment. For the system's environment, the real behavior is determined by the environmental uncertainty due to unforeseeable situations that may occur during operation. The observed behavior is a subset of the real behavior and

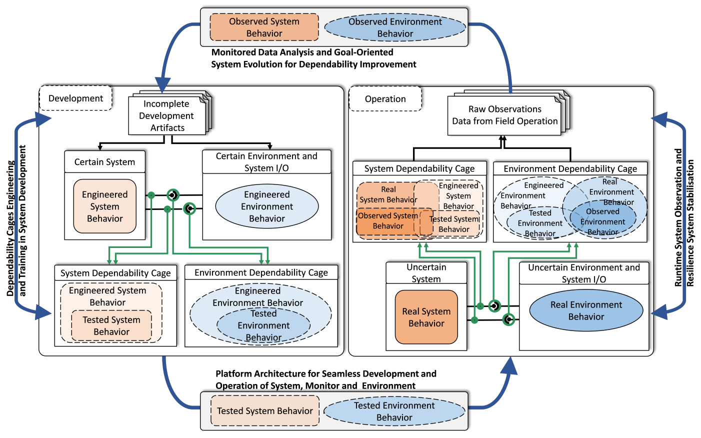

Observed System Behavior
Observed Environment Behavior
Monitored Data Analysis and Goal-Oriented System Evolution for Dependability Improvement
Development
Incomplete Development Artifacts
Operation
Raw Observations Data from Field Operation
Certain System
Engineered System Behavior
Certain Environment and System I/O
Engineered Environment Behavior
System Dependability Cage
Engineered System Behavior
Tested System Behavior
Environment Dependability Cage
Engineered Environment Behavior
Tested Environment Behavior
System Dependability Cage
Real System Behavior
Engineered System Behavior
Observed System Behavior
Tested System Behavior
Environment Dependability Cage
Engineered Environment Behavior
Real Environment Behavior
Tested Environment Behavior
Observed Environment Behavior
Uncertain System
Real System Behavior
Uncertain Environment and System I/O
Real Environment Behavior
Dependability Cages Engineering and Training in System Development
Runtime System Observation and Resilience System Stabilisation
Platform Architecture for Seamless Development and Operation of System, Monitor and Environment
Tested System Behavior
Tested Environment Behavior

Figure 3. Dependability Cages: Overall approach. [8]

represents the behavior monitored at operation time through the Dependability Cages.

Once the tests at development time are completed, the tested behavior is transferred into operation time via a platform envisioned for this purpose (cf. Figure 3 bottom). In turn, the observed behavior is channeled back to development time to augment the development artifacts and contribute the AS's evolution, and consequently, the improvement of the system's dependability through further training of the Dependability Cages (cf. Figure 3 top). Thus, the Dependability Cage approach consists of four major parts:

- Dependability Cages Engineering and Training in System Development
- Runtime System Observation and Resilience System Stabilization
- Monitored Data Analysis and Goal-Oriented System Evolution for Dependability Improvement
- Platform Architecture for Seamless Development and Operation of System, Monitor and Environment

Due to the scope of this paper, we will discuss only the part of Runtime System Observation and Resilience System Stabilization.

C. Section Overview

This paper is organized as follows: In the following sections we will describe our connected dependability cage concept on different levels of abstraction. Section II defines the high level components of our Dependability Cage which we use for Runtime System Observation and Resilience System Stabilization and especially also introduces the remote operator for the *Connected* Dependabilty Cage concept. Following up on that, Section III introduces a more concrete architecture with lower level components for the VanAssist Project. And finally in Section IV we demonstrate our Connected Dependabilty Cage concept on the use case of the VanAssist project and concretize the realised implementation.

II. The Core: Runtime System Observation and Resilience System Stabilization

As introduced before, one of the major parts in the Dependability Cage concept is the Runtime System Observation and Resilience System Stabilization. The core element of this part is an onboard continuous monitoring framework, as shown in Figure 4. For reasons of simplicity, we depict the architecture for automated driving systems by following the input-processing-output pattern along a well-known high-level reference architecture established previously [18] [19] [20]. This reference architecture consists of three parts: (1) environment- and self-perception, (2) situation comprehension and action decision, and (3) trajectories planning and vehicle control.

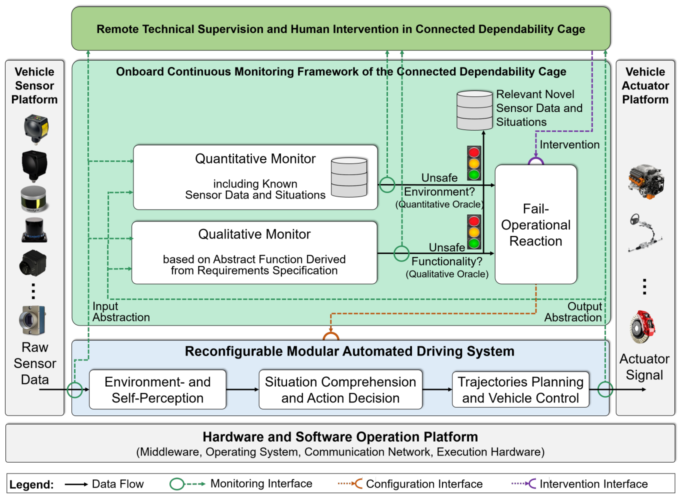

Remote Technical Supervision and Human Intervention in Connected Dependability Cage
Vehicle Sensor Platform
Onboard Continuous Monitoring Framework of the Connected Dependability Cage
Vehicle Actuator Platform
Relevant Novel Sensor Data and Situations
Intervention
Quantitative Monitor
including Known Sensor Data and Situations
Unsafe Environment? (Quantitative Oracle)
Fail-Operational Reaction
Qualitative Monitor
based on Abstract Function Derived from Requirements Specification
Unsafe Functionality? (Qualitative Oracle)
Input Abstraction
Output Abstraction
Raw Sensor Data
Actuator Signal
Reconfigurable Modular Automated Driving System
Environment- and Self-Perception
Situation Comprehension and Action Decision
Trajectories Planning and Vehicle Control
Hardware and Software Operation Platform
(Middleware, Operating System, Communication Network, Execution Hardware)
Legend: Data Flow Monitoring Interface Configuration Interface Intervention Interface

Figure 4. Runtime System Observation and Resilience System Stabilization in the Connected Dependability Cage approach based on [6].

The monitoring framework focuses on two issues: (a) does the system show correct behavior in terms of dependability requirements (Qualitative Monitor) and (b) does the system operate in a situation and environment that has been trained or tested during development (Quantitative Monitor)?

Both monitors require consistent and abstract data access to the system under consideration, with the help of the input and output abstraction components. They depict the interface between the AS and the two monitors. Both, the input and output abstraction components use defined interfaces to access the AS's data and transform it into abstract representations. Abstract representation types and values are derived from the requirements specification and dependability criteria of the AS.

The Qualitative Monitor evaluates the correctness and safety of the system's behavior under the assumption that (a) the system operates in a situation and environment that conforms to the requirements specification, and (b) the system consists of an abstract behavior function and a conformity oracle. The abstract behavior function calculates a set of correct and safe actions in real-time for the system in the current abstract situation. The conformity oracle compares the output abstraction with the set of correct and safe abstract actions from the abstract behavior function. For applications of the Qualitative Monitor, we refer to the work of Grieser et al. [21] and Mauritz et al. [9] [11].

The Quantitative Monitor observes the encountered abstract situations. For each situation, the monitor evaluates in real time if the encountered abstract situation is already known from development. A knowledge base provides information about tested situations on an abstract level. A canonical representation of abstract situations is used in this study. These canonical abstract situations are considered to be unique situations. For further details about the Quantitative Monitor, we refer to the work of Rausch et al. [6].

If one of both monitors either identifies an incorrect and unsafe system behavior or a novel situation (outliers), safety measures must be initialized to guarantee dependability requirements. For this purpose, a Fail-Operational Reaction component is designed in the monitoring framework to immediately transfer the failed system into a safe state with acceptance of appropriate risks, e.g., following the approach of a graceful degradation like in [22].

In addition to the initialized safety measure, the system data are automatically logged, which will be used as artifacts during the system's further development, aiming to analyze and eliminate previously unknown faulty system behaviors and thus increase the system's testing scope. Relying on such a process flow of runtime system observation and a continuous iterative development process, a so-called resilience system stabilization is realized.

In the reality, it is utopian that the onbaord continuous monitoring framework based on the technical system can handle all critical cases of the automated driving system. Thus, a redundant monitoring element, a so-called Remote Technical Supervision and Human Intervention is proposed in the Dependability Cage concept from the safety perspective. The Remote Technical Supervision and Human Intervention plays the role of a complementary consultant of the onboard continuous monitoring framework, so that a remote human supervisor can lively access current situations of the AVs during their operations. In the case that the current problem cannot be solved locally by the fail-operation activity of the onboard monitoring framework, a human intervention of the remote supervisor can be performed, e.g., by manually choosing an appropriate fail-operational reaction. Such a monitoring and supervision concept with redundancy massively increases the AV's safety. Moreover, the Remote Technical Supervision and Human Intervention also enables to remove the required onboard safety driver from the AVs, following the draft bill for the german traffic rule law [23]. Thus, the Remote Technical Supervision and Human Intervention work together with the onboard continuous monitoring framework and constitute an cooperative solution of a so-called *Connected* Dependability Cage concept. Since the Remote Technical Supervision and Human Intervention is not located on the AV, interfaces to a communication infrastructure are designed, so that we will now speak about a Connected Dependability Cage.

III. Instantiation of the Connected Dependability Cage for real World Applications of Automated Vehicles

The Connected Dependability Cage approach as a generalized concept addresses the safety domains of different automated mobility solutions. In reality, different domains have their constraints and boundary requirements like hardware setups, regulations, and physical performance limits. To evaluate the approach's feasibility, we applied the concept of Runtime System Observation and Resilience System Stabilization in the overall approach by instantiating a more concrete architecture. The instantiated architecture is designed considering the use case of the research project VanAssist [24].

A. Use-Case Definition

The project VanAssist aims to develop an integrated vehicle and system technology that enables largely emission-free and automated delivery of goods in urban centers. This project focuses on how automated transporters for delivery of goods can helps to optimise the daily jobs of postmen by creating optimized routes which are not possible in the classical manner. For further information about the project VanAssist, we want to refer to [24] and the VanAssist website [25]. Since misbehavior of the automated transporter can lead to hazardous situations, such a system is considered a safety critical system. In such a safety critical system, an overall safety architecture is required to guarantee that a so-called minimal risk condition is reached in case a hazardous situation happens [26]. However, in such states the monitored system tends to not be able to continue its mission and might cause traffic jams for instance. To avoid such undesired behavior in the project, the previously mentioned Remote Technical Supervision and Human Intervention is required, which is implemented as a Remote Command Control Center to acquire and supervise the AV's context information and realize the human intervention if necessary. Thus the Remote Operator shall always get enough information about the AV's context, which is mainly provided by cameras. Within the use case, we addressed the following safety requirements:

1) The AV shall never drive against or over obstacles
2) Camera data of the AV shall never be invalid

In the project, a goods delivery transporter – the Platform for Future Urban Mobility and Transport (PLUTO) – with an automated driving system was developed as an evaluation platform by the Automotive Research Centre Niedersachsen in Germany [27]. The PLUTO serves as an evaluation platform in the project VanAssist and is configured as an automated goods delivery transporter. For environmental perception four fish-eye cameras and eight high-performance LiDARs are used to create a surround view with 360° degrees.

B. Instance Architecture of Dependability Cage in the Use Case

Since the aforementioned safety requirements address the functional correctness of the automated system, the Quantitative Monitor concept (cf. Section II) with another safety goal was not applied in the project. Instead, the Qualitative Monitor and the Fail-Operational Reaction in the onboard runtime monitoring framework and the remote technical supervision were derived as the main aspects of the instantiated architecture.

The instantiation of the Connected Dependability Cage is driven by the use case defined above and thus addresses automated vehicles using LiDARs and cameras to perceive local environmental information and is based on the work of Raulf et al. [28]. The instance of the Connected Dependability Cage is depicted in Figure 5.

The instantiated system architecture consists of three subsystems: (a) A Reconfigurable Modular Automated Driving System that performs the driving task and provides reconfiguration interfaces, (b) a Dependability Cage that executes the onboard Qualitative Monitoring and the Fail-Operational Reaction at runtime, and (c) a Remote Command Control Center with HMI for the offboard supervision and human intervention by a remote Teleoperator. In addition, the Remote Command Control Center also provides interfaces to external entities to constitute a more complicated networked architecture consisting of multiple Connected Dependability Cages.

As illustrated in Figure 5, the automated driving system is the target system under runtime observation by the Connected Dependability Cage and Remote Command Control Center, which can be seen as a cooperative human-machine monitoring system for the safeguard of the automated driving system. Once the automated driving system detects a hazardous situation, the monitoring system would trigger an appropriate reaction and perform a reconfiguration via the interface provided by the driving system. Depending on concrete cases of the detected hazardous situations, the responsibilities between the Dependability Cage for onboard runtime monitoring and the Remote Command Control Center would be dynamically changed. Both can also be understood as redundant monitoring systems (respectively onboard and offboard) that aim to minimize the safety risk as far as possible.

In the following, we will focus on the internals of the instance architecture from Figure 5. Due to the scope of this work, we will only focus on the internals of (b). Within (b) the Qualitative Monitor consists of three components: ”Safe Zone”, ”LiDAR Detector” and ”Camera Validator”. Whereas the Fail-Operational Reaction consists of one component: the ”Mode Control”.

The component ”Safe Zone” is designed to determine areas, in which obstacles shall not occur based on the physical characteristics like the dynamic driving behavior and stopping distance based on the steering angle and current speed of the monitored system. Thus the *Safe Zone* remarks hazard areas in which obstacles shall not occur when performing the dynamic driving task. This component is an generalisation of the work of Grieser et al. [21] to determine a monitoring area and can consist of subzones which can correlate with driving modes.

The component ”LiDAR Detector” focuses on the fulfillment of safety requirements relevant to the LiDAR-sensor. It checks if obstacles are in the *Safe Zone* or in specific subzones based on LiDAR point cloud(s). If an obstacle is identified,

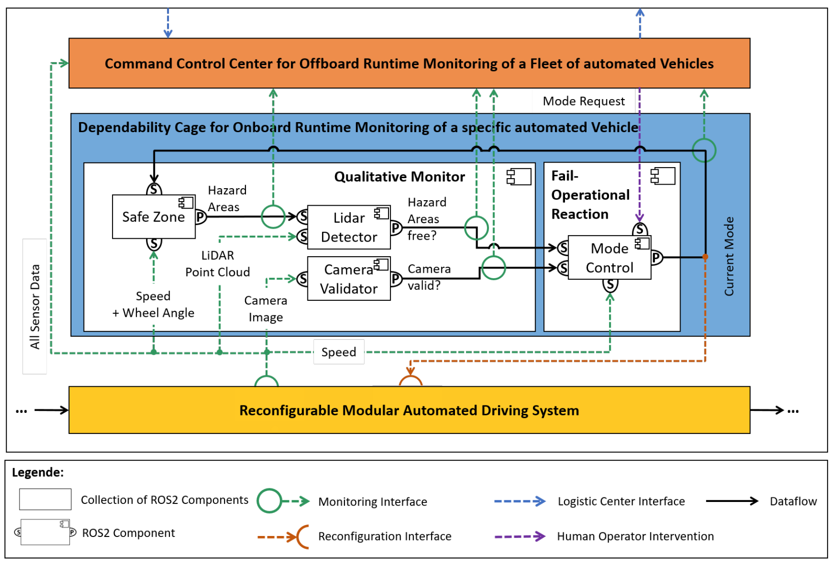

Command Control Center for Offboard Runtime Monitoring of a Fleet of automated Vehicles
Mode Request
Dependability Cage for Onboard Runtime Monitoring of a specific automated Vehicle
Qualitative Monitor
Fail-Operational Reaction
Safe Zone
Hazard Areas
Lidar Detector
Hazard Areas free?
Camera Validator
Camera valid?
Mode Control
LiDAR Point Cloud
Speed + Wheel Angle
Camera Image
Current Mode
All Sensor Data
Speed
Reconfigurable Modular Automated Driving System
Legende:
Collection of ROS2 Components
ROS2 Component
Monitoring Interface
Reconfiguration Interface
Logistic Center Interface
Human Operator Intervention
Dataflow

Figure 5. Instance of the Connected Dependability Cage based on [28] for automated vehicles using LiDARs and Cameras to perceive local environmental information.

a trigger is caused, which is consumed by components of the Fail-Operational Reaction. To safeguard the automated driving system and the Remote Operator from unusable camera data, the ”Camera Validator” checks if camera data is valid and causes triggers for a Fail-Operational Reaction in the case of invalid data.

The ”Mode Control” was derived as a component of the Fail-Operational Reaction and is responsible for determining an appropriate driving mode based on the current situation. The driving mode can be either a nominal, a degradated or a safe mode and directly correlates with instances responsible for overall system safety. The selection is based on triggers caused by the Qualitative Monitor and other necessary information correlating with driving modes.

IV. Detailed concepts in the instance Architecture

In the previous section, we introduced a safety architecture called Connected Dependability Cage for automated vehicles which uses LiDAR and camera sensors for the environmental perception. One main part, the Qualitative Monitor, continuously monitors the selected Automated Driving System. In this section, we will focus on the Qualitative Monitor implementation and will explain the motivation and concepts of the components within this safety architecture(see Figure 5). Additionally, we will provide a proof of concept in the scope of the two safety requirements presented in the previous section using PLUTO [27] as a demonstration platform. In the following, we will call the PLUTO AV.

To fulfill the safety requirements, the Qualitative Monitor shall (a) detect hazardous situations in the AV’s surroundings and (b) detect invalid camera data produced by the AV’s cameras. Once a violation is detected a fail operational reaction is triggered. Following the safety architecture of the Connected Dependability Cage in Figure 5, the implementation of the Qualitative Monitor M_{quali} can be represented in a functional description as follows:

$$M_{quali}(v, \alpha_{steering}, P_{li}, I_{cam}) = f(Z_{safe}, D_{li}, V_{cam}) \quad (1)$$

$$D_{li}(Z_{safe}, P_{li}) = f(z_{state}) \quad (2)$$

$$V_{cam}(I_{cam}) = f(c_{state}) \quad (3)$$

$$Z_{safe}(v, \alpha_{steering}) = f(Z_{clear}, Z_{focus}) \quad (4)$$

v	current AV speed
$\alpha_{steering}$	steering angle of the outer front wheel
P_{li}	LiDAR point cloud
I_{camera}	camera image
z_{state}	Zone state (free/blocked)
c_{state}	Camera state (valid/invalid)
Z_{clear}	Clear Zone (inner zone)
Z_{focus}	Focus Zone (outer zone)

Taking the safety requirements, the AV’s hardware setup and physical attributes into account, we took the work of Grieser et al. [21] as a starting point and extended it, resulting in the development of a ”Safe Zone” component Z_{safe}, a ”LiDAR Detector” D_{li}, a ”Camera Validator” V_{camera} and a ”Mode Control” component.

A. *Safe Zone*

The purpose of the ”Safe Zone” Component is to calculate a hazard zone around the AV, as shown in Figure 6 and Figure 7. In addition to the main hazard zone, we also defined an outer hazard zone, which has larger offsets to all sides. This was used to demonstrate the concept of graceful degradation like in [22]. To differentiate between these two zones, we introduced the terms *Clear Zone*, for the main hazard zone and *Focus Zone*, for the outer hazard zone. The most important information required to calculate these hazard zones are the vehicle’s trajectory, its direction of driving and the stopping distance.

Since the approach for calculating the stopping distance is the same as in the previous work of Grieser et al. [21], we only briefly recapitulate this part in the first subsection. In the later subsections the generalized zone shape and the derivation of the corresponding parameters are described in more detail.

Stopping Distance

The calculation of the stopping distance of the AV is based on Ackermann steering geometry. For the basic calculation of the stopping distance, we are just listing the formulas 5 through 11 and for a more detailed description of this part of the approach, we refer to Grieser et al. [21].

Stopping distance s_{stop} (and intermediate results) for straight driving:

$$s_{stop} = s_{reaction} + s_{brake} \tag{5}$$

$$s_{reaction} = v \cdot t_{reaction} \tag{6}$$

$$s_{brake} = \frac{v^2}{2 \cdot a_{brake}} \tag{7}$$

$s_{reaction}$	distance travelled during the AV's reaction time
s_{brake}	distance travelled during the AV's braking time
$t_{reaction}$	estimated worst case reaction time of the AV
v	current speed of the AV
a_{brake}	braking deceleration of the AV

Stopping angle α_{stop} (and intermediate results) for curved driving:

$$\alpha_{stop} = s_{stop}/r_{mean} \tag{8}$$

$$r_{mean} = (r_o + r_i)/2 \tag{9}$$

$$r_o = d_{axle}/sin(\alpha_{steering}) \tag{10}$$

$$r_i = \sqrt{r_o^2 - d_{axle}^2} - d_{wheel} \tag{11}$$

r_{mean}	mean value of outer and inner radius
r_o	distance from the center of steering M to the outer front wheel
r_i	distance from the center of steering M to the inner back wheel
$\alpha_{steering}$	steering angle of the outer front wheel
d_{axle}	distance from the center of the back axle to the center of the front axle
d_{wheel}	distance from wheel center to opposing wheel center

Zone Shape

To accommodate the perception system being able to detect objects all around the AV, we needed to generalize the concept of the hazard zone described in the previous work. While one end of the zone is still placed at the end of the stopping path, the other end is now placed at the axle opposite to the driving direction (compare Figure 6 and Figure 7).

For driving straight the resulting zone shape is still a rectangle; on the other hand for driving curved the shape was generalized to a circle segment. In the following subsections the derivation of the zone shapes is described in more detail for the different cases.

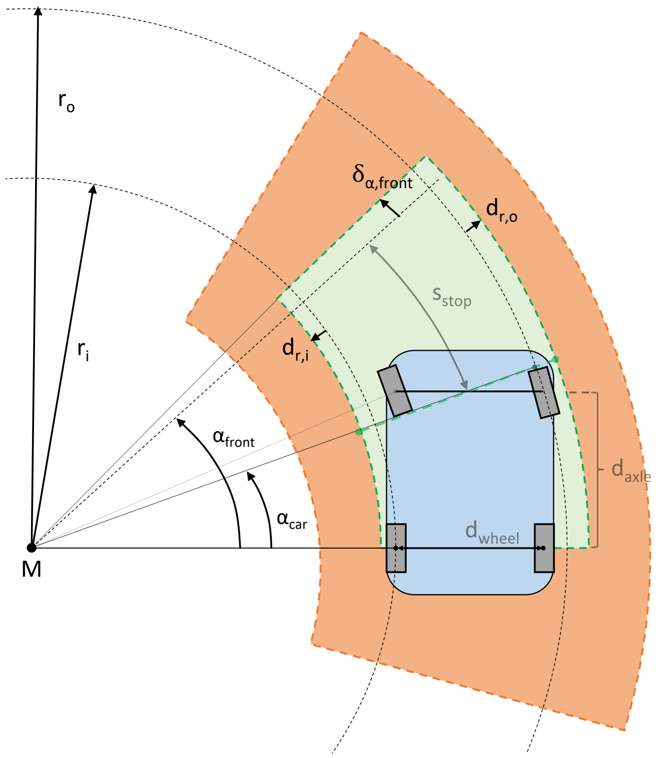


Figure 6. Forward Driving: Determination of Safe Zone with Clear Zone (green) and Focus Zone (orange).

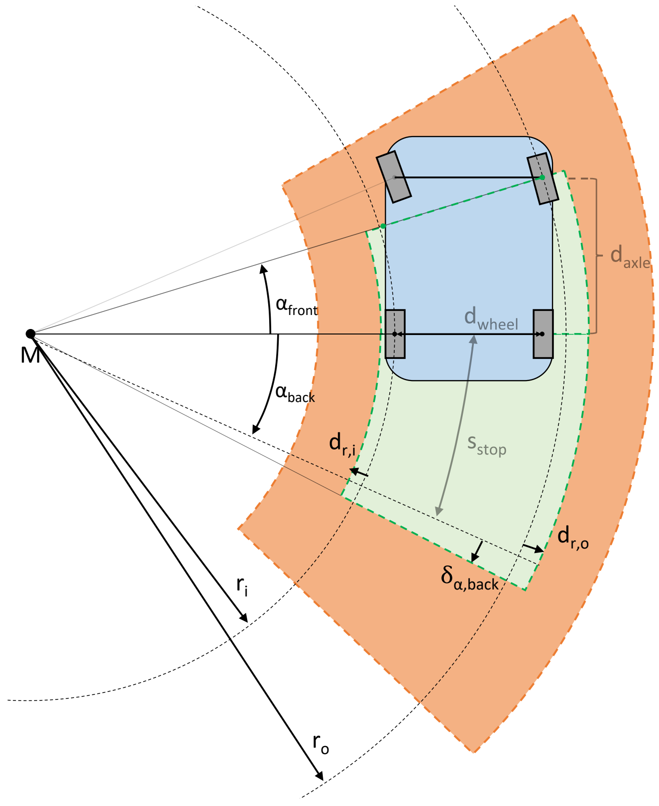


Figure 7. Backward Driving: Determination of Safe Zone with Clear Zone (green) and Focus Zone (orange).

Driving Straight

In the case of driving straight, the width of the zone is given by the wheel distance d_{wheel} plus a safety offset (on both sides). Whereas the length of the zone now is the sum of the axle distance d_{axle} and stopping distance s_{stop}, again plus a safety offset.

Driving Curved

Overall the zone for driving curved is described by four parameters: The inner radius r_i, and the outer radius r_o. The angle α_{back}, which starts the circle segment at the back side of the AV and the angle α_{front}, which ends the circle segment at the front side of the AV.

Outer radius r_o and inner radius r_i are always calculated by formulas 10 and 11, which are already used for the basic stopping distance calculation. The derivation of the other parameters is described in the corresponding subsections for the different cases.

Driving Curved, Forward

For the case of driving forward (see Figure 6) the starting point of the back angle is placed on the rear axle, so that:

$$\alpha_{back} = 0 \tag{12}$$

The starting point for the stopping distance is placed in the center of the front axle, so that the total front angle α_{front} is the sum of the stopping angle α_{stop} and the angle α_{car} to the front of the car:

$$\begin{aligned} \alpha_{front} = &\ \alpha_{car} + \alpha_{stop} \\ with &\ \ \alpha_{car} = atan(r_i + d_{wheel}/2) \end{aligned} \tag{13}$$

Driving Curved, Backward

For driving backwards (see Figure 7) the back angle is directly equal to the stopping angle, since the start of the stopping circle segment is directly aligned with the back axle:

$$\alpha_{back} = -\alpha_{stop} \tag{14}$$

And the end point of the front angle is placed at the center of the outer front wheel, which results in the front angle being equal to the steering angle:

$$\alpha_{front} = \alpha_{steering} \tag{15}$$

Driving Curved, Safety Offsets

To obtain the final set of zone parameters, we add a safety distance in all four directions in the following manner:

$$\alpha'_{front} = \alpha_{front} + \delta_{\alpha_{front}} \tag{16}$$

$$\alpha'_{back} = \alpha_{back} + \delta_{\alpha_{back}} \tag{17}$$

$$r'_i = r_i + d_{r_i} \tag{18}$$

$$r'_o = r_o + d_{r_o} \tag{19}$$

Where for example the angle $\delta_{\alpha_{front}}$ is the safety offset for α_{front} and d_{r_i} is the safety offset for r_i(analogously for the other parameters). These safety offsets are implementation specific application parameters.

B. LiDAR Detector

The ”LiDAR Detector” component determines whether there are any obstacles in the *Clear Zone* or *Focus Zone* based on the Point Cloud from the eight Ibeo solid-state 3D laser scanners. For this purpose, the method of Grieser et al. [21] to determine whether there are LiDAR points in the monitoring area is extended.

Compared to the model vehicle, we faced several additional challenges for the LiDAR of the AV. The fundamental difference, is that the AV uses a 3D LiDAR setup, whereas the model vehicle LiDAR only measures two dimensional in a horizontal plane.

Consequently the ground is included in the LiDAR data and also obstacles which are higher then the AV, but still in LiDAR range. To exclude these areas, we implemented a z-cutoff, at a certain offset from the ground and a certain offset above the vehicle. In the resulting data, only z levels which are relevant for the *Safe Zone* were included, since we only want to determine if an object is in the vertical area of the zone and not where it is. This allowed us to subsequently ignore the height in the processed data and simplify it from 3D to 2D again.

Another challenge was, that the AV's LiDAR setup provides a lot more data with higher amount of details. This also means, that the data includes more points, which belong to small particles or diffusely reflected laser beams instead of actual obstacles. In the following, we will refer to these points as ”ghost points”. To filter out ghost points, we implemented a clustering algorithm, which clusters neighbouring points together, if their distance is smaller then our empirically determined cluster distance. Using this clustering algorithm enabled us to successfully reduce false emergency stops due to ghost points.

The result of the clustering algorithm is used in the ”LiDAR Detector” in the following way: To check the state z_{state} of the zone, we determine the largest number of LiDAR points $p_{cluster}$, which are inside the rectangle/circle segment and belong to the same cluster. If this number is below the defined threshold value $t_{cluster}$, then the zone is considered as free, otherwise as blocked:

$$z_{state} = \begin{cases} free & p_{cluster} < t_{cluster} \\ blocked & p_{cluster} >= t_{cluster} \end{cases} \tag{20}$$

This test is carried out for Clear Zone as well as Focus Zone and the result is forwarded to the Mode Control.

C. Camera Validator

The ”Camera Validator” component determines the validity of the raw data from the onboard camera sensor by checking whether the camera is covered or not. An algorithm is used to check the validity of the camera data. This algorithm determines the sharpness of the incoming stream of camera images, which detects that the camera is covered when the level of sharpness value falls below an empirically determined threshold value.

D. Mode Control

The ”Mode Control” is a component of the Fail-Operational Reaction. It decides the current driving mode. It could be either a nominal, degraded, or safe mode, based on the ”LiDAR Detector’s” output, ”Camera Detector’s” output and the current speed of the AV. If the Qualitative Monitor components detect any problems, the ”Mode Control” changes the mode from the nominal driving mode to the fail-safe mode (emergency stop). Once a fail-safe mode happens, the remote operator has the authority to select an appropriate driving mode, e.g., a degradaded mode like limited automated driving.

The logic behind the ”Mode Control” is modelled as a synchronous hierarchical automata in SCADE, enabling verification of the mode logic.

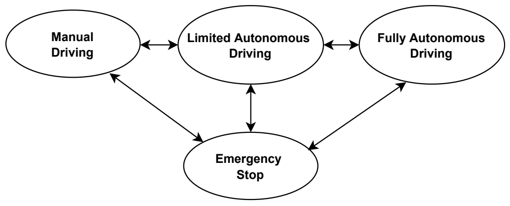


Figure 8. Simplified mode diagram of the component ”Mode Control”.

A simplified version of the mode diagram of the ”Mode Control” can be seen in Figure 8. Due to the scope of this work, we will not dive deeper into the details of this component.

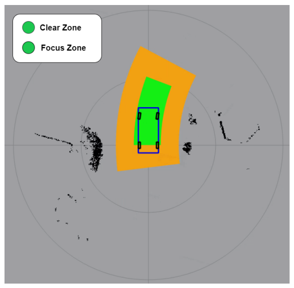
E. Implementation Middleware

Each component in the architecture, as seen in Figure 5 is implemented using the decentralized middleware ROS2 [29]. ROS2 is based on the publish-subscribe communication paradigm and provides the capability of self-adaptation and component reconfiguration. Since the AV is designed as a distributed system deployed on different ECUs, it motivated us to use ROS2. An additional point in favor of ROS2 for a safety critical system is, that it provides real-time capabilities.

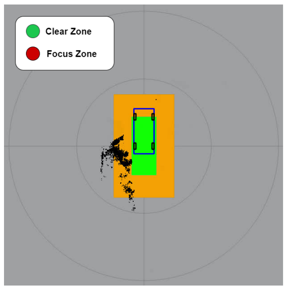
F. Testing of the Qualitative Monitoring Architecture

The Qualitative Monitor is tested in two steps. First, it was conducted in a controlled environment where a track similar to the AV’s test track was setup in our mobility lab using a 1:8 scaled ADAS model vehicle (similar to [21]).

As a second step, we reparameterized the *Focus Zone* and *Clear Zone* in the ”Safe Zone” component for the test on the AV, since the dynamics and hardware setup vary from the model vehicle. The track considered for the test is located at the Campus Nord of TU-Braunschweig (Bienroder Weg 95, 38106 Braunschweig – for more information refer to [24] [27] and the VanAssist website [25]). Multiple test cases were derived and tested, such as driving straight, driving backward, driving at different angles, and driving towards static objects and walls. Two examples from our numerous test cases can be seen in Figure 9a and Figure 9b. In the images the LiDAR Point Cloud is visualized by black points and the *Focus Zone* (orange), and *Clear Zone* (green) can be seen around the AV, which is denoted by a blue box at the center. The indicators on the top left corner signal if the zones are free or blocked, respectively with green or red color.



(a) Safe Zone free



(b) Safe Zone blocked

Figure 9. Dependability Cage Test.

In the first example (see Figure 9a), we drove the AV within a curved section of the track, where no obstacles were detected by the ”LiDAR Detector” within the *Focus Zone* and *Clear Zone* calculated in the ”Safe Zone” component. As a result the ”Mode Control” does not intervene and the AV stays in the nominal driving mode.

In the second example (see Figure 9b), The AV drove on a straight section of the test track surrounded by obstacles. These obstacles are detected as LiDAR points (black dots) inside the *Focus Zone*, but the *Clear Zone* is still free as shown in Figure 9b. Since the amount of clustered LiDAR points detected by the ”LiDAR Detector” component within the *Focus Zone* is above the threshold, it resulted in the ”Mode Control” triggering a fail-safe mode (emergency stop), if the nominal driving mode uses the *Focus Zone* for fail-safe checks.

V. Conclusion and Future Work

In this paper, a short overview of the Connected Dependability Cage approach is provided. As a core part, the meta-concept of Runtime System Observation and Resilience System Stabilization relying on the onboard continuous monitoring framework and the remote technical supervision allowing human supervision and intervention is introduced in detail. In addition, a concrete system architecture of the Runtime System Observation and Resilience System Stabilization is instantiated considering the derived safety requirements based on a real-world application of the AV for package delivery in urban areas. In the instantiated architecture, the Qualitative Monitor and Fail-Operational Reaction are deployed to guarantee the functional correctness of the automated driving system under runtime observation. The instantiated architecture was implemented and tested on an automated demonstration vehicle (PLUTO) in the project VanAssist. Detailed insights on the implementation were included in the paper, too.

On the way towards dependable automated driving, significant challenges that we have identified in the Connected Dependability Cage approach, are remaining. The Connected Dependability Cage allows an asymmetric assignment of AVs to the remote operators in the Command Control Center. It means that a single remote operator may supervise multiple AVs in normal operation. But the person would not be able to remotely solve the safety issues of AVs in problem cases simultaneously. Thus, an intelligent coordination and assignment between remote operators and the AV having safety issues needs to be investigated in the Connected Dependability Cage approach.

In addition, AVs under SAE Level 3+ are fail-operational [26]. For this purpose, the previously proposed concept of graceful degradation proposed by [22] for the fail-operational reaction in the instance architecture still needs to be implemented and tested. Additionally, the automated driving function shall stay within the specified Operational Design Domain (ODD) during normal operation [26], another way to further develop the Connected Dependability Cage would be the ODD monitoring.

As introduced at the beginning of this paper, ASs are nowadays AI-based. Therefore the system strongly relies on the training data without explicit requirements specifications. Thus, the Qualitative Monitor would reach its limitation to work against the safety issues due to unknown situations, which cannot be explicitly described by the requirements specifications. In this case, the Quantitative Monitor in the Connected Dependability Cage approach like presented in [6] would be a meaningful solution and needs to be implemented and evaluated in the future.

Acknowledgments

This work results from the joint project ”VanAssist - Interactive, intelligent system for autonomous telemonitored vans in parcel logistics” and has been funded by the Federal Ministry of Transport and Digital Infrastructure based on a resolution of the German Bundestag.

References

[1] M. Anderson, James *et al.*, “Autonomous systems: Issues for defence policymakers,” Headquarters Supreme Allied Commander, Tech. Rep., 2015.

[2] J. Youtie, A. L. Porter, P. Shapira, S. Woo, and Y. Huang, “Autonomous systems: A bibliometric and patent analysis,” Exptertenkommission Forschung und Innovation, Tech. Rep., 2017.

[3] V. C. Müller, Ed., *Fundamental Issues of Artificial Intelligence*, ser. Synthese Library. Cham: Springer International Publishing, 2016.

[4] J. Rushby, *Quality measures and assurance for AI software*, 1988, vol. 18.

[5] D. Harel, A. Marron, and J. Sifakis, “Autonomics: In search of a foundation for next-generation autonomous systems,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 117, no. 30, pp. 17 491–17 498, 2020.

[6] A. Rausch, A. M. Sedeh, and M. Zhang, “Autoencoder-based semantic novelty detection: Towards dependable ai-based systems,” *Applied Sciences*, vol. 11, no. 21, p. 9881, 2021.

[7] T. Sickert. (2016, March) From hipster-girl to hitler-bot. Spiegel Netzwelt. [Online]. Available: https://www.spiegel.de/netzwelt/web/microsoft-twitter-bot-tay-vom-hipstermaedchen-zum-hitlerbot-a-1084038.html (retrieved: 2022.03.10).

[8] A. Aniculaesei, J. Grieser, A. Rausch, K. Rehfeldt, and T. Warnecke, “Towards a holistic software systems engineering approach for dependable autonomous systems,” in *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, R. Stolle, S. Scholz, and M. Broy, Eds. New York, NY, USA: ACM, 2018, pp. 23–30.

[9] M. Mauritz, F. Howar, and A. Rausch, “Assuring the safety of advanced driver assistance systems through a combination of simulation and runtime monitoring,” in *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications*, ser. Lecture Notes in Computer Science, T. Margaria and B. Steffen, Eds. Cham: Springer International Publishing, 2016, vol. 9953, pp. 672–687.

[10] M. Mauritz, A. Rausch, and I. Schaefer, “Dependable adas by combining design time testing and runtime monitoring,” in *FORMS/FORMAT 2014 - 10th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems*, 2014.

[11] M. Mauritz, “Engineering of safe autonomous vehicles through seamless integration of system development and system operation,” Ph.D. dissertation, Universitätsbibliothek der TU Clausthal, 2020.

[12] M. Mauritz, F. Howar, and A. Rausch, “From simulation to operation: Using design time artifacts to ensure the safety of advanced driving assistance systems at runtime,” in *MASE@MoDELS*, 2015.

[13] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, “A review of novelty detection,” *Signal Processing*, vol. 99, pp. 215–249, 2014.

[14] N. Japkowicz, C. Myers, and M. Gluck, “A novelty detection approach to classification,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, ser. IJCAI’95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 518–523.

[15] C. Richter and N. Roy, “Safe visual navigation via deep learning and novelty detection,” in *Robotics: Science and Systems*, 2017.

[16] A. Alexander *et al.*, “Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing.” IEEE, 2018, pp. 568–575.

[17] R. Brooks, “The big problem with self-driving cars is people,” *IEEE spectrum: technology, engineering, and science News*, vol. 27, 2017.

[18] S. Behere and M. Törngren, “A functional architecture for autonomous driving,” in *Proceedings of the First International Workshop on Automotive Software Architecture*, P. Kruchten, Y. Dajsuren, H. Altinger, and M. Staron, Eds. New York, NY, USA: ACM, 2015, pp. 3–10.

[19] S. Behere and M. Törngren, “A functional reference architecture for autonomous driving,” *Information and Software Technology*, vol. 73, pp. 136–150, 2016.

[20] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, *Autonomes Fahren*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[21] J. Grieser, M. Zhang, T. Warnecke, and A. Rausch, “Assuring the safety of end-to-end learning-based autonomous driving through runtime monitoring,” in *2020 23rd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2020, pp. 476–483.

[22] A. Aniculaesei, J. Griesner, A. Rausch, K. Rehfeldt, and T. Warnecke, “Graceful degradation of decision and control responsibility for autonomous systems based on dependability cages,” in *5th International Symposium on Future Active Safety Technology toward Zero*, Blacksburg, Virginia, USA, 2019.

[23] German Government. Draft law amending the road traffic act and the compulsory insurance act - autonomous driving act. Federal Ministry for Digital and Transport. [Online]. Available: https://www.bmvi.de/SharedDocs/DE/Anlage/Gesetze/Gesetze-19/gesetz-aenderung-strassenverkehrsgesetz-pflichtversicherungsgesetz-autonomes-fahren.pdf?__blob=publicationFile (retrieved: 2022.03.10).

[24] G. Seber *et al.* Final report VanAssist. [Online]. Available: https://www.vanassist.de/ergebnisse/ (retrieved: 2022.03.10).

[25] VanAssist website. [Online]. Available: https://www.vanassist.de (retrieved: 2022.03.10).

[26] S. International, “SAE J3016 - surface vehicle recommended practice - taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,” 2018.

[27] T. Hegerhorst *et al.*, “VanAssist - integrated safety concept for automated vans in parcel logistics,” in *ACIMobility Summit*, Braunschweig, Germany, september 2021.

[28] C. Raulf *et al.*, “Dynamically configurable vehicle concepts for passenger transport,” in *13. Wissenschaftsforum Mobilität ”Transforming Mobility – What Next”*, Duisburg, Germany, 2021.

[29] ROS2 website. [Online]. Available: https://docs.ros.org/en/foxy/index.html (retrieved: 2022.03.10).