A Formally Verified Fail-Operational Safety Concept for Automated Driving

ARTICLE INFO
Article ID: 12-05-01-0002
© 2022 SAE International
doi:10.4271/12-05-01-0002

A Formally Verified Fail-Operational Safety Concept for Automated Driving

Yuting Fu,[1] Andrei Terechko,[1] Jan Friso Groote,[2] and Arash Khabbaz Saberi[2]

[1]*NXP Semiconductors, The Netherlands*
[2]*Eindhoven University of Technology, The Netherlands*

Abstract

Modern Automated Driving (AD) systems rely on safety measures to handle faults and to bring the vehicle to a safe state. To eradicate lethal road accidents, car manufacturers are constantly introducing new perception as well as control systems. Contemporary automotive design and safety engineering best practices are suitable for analyzing system components in isolation, whereas today's highly complex and interdependent AD systems require a novel approach to ensure resilience to multiple-point failures. We present a holistic and cost-effective safety concept unifying advanced safety measures for handling multiple-point faults. Our proposed approach enables designers to focus on more pressing issues such as handling fault-free hazardous behavior associated with system performance limitations. To verify our approach, we developed an executable model of the safety concept in the formal specification language mCRL2. The model behavior is governed by a four-mode degradation policy-controlling distributed processors, redundant communication networks, and virtual machines (VMs). To keep the vehicle as safe and cost effective as possible, our degradation policy can reduce driving comfort or AD system's availability using additional low-cost driving channels. We formalized five safety requirements in the modal μ-calculus and proved them against our mCRL2 model, which is intractable to accomplish exhaustively using traditional road tests or simulation techniques. In conclusion, our formally proven safety concept defines a holistic and cost-effective design pattern for AD systems.

This article is part of a Special Issue on Autonomy and Connectivity at the Edge—Autonomous Racing.

History

Received:	07 Dec 2020
Revised:	02 Jul 2021
Accepted:	20 Dec 2021
e-Available:	17 Jan 2022

Keywords

Automated driving, Safety concept, Safety measures, Degraded mode, Redundancy, Multiple-point fault, Situational awareness, Driving decision-making, Formal verification, Modeling

Citation

Fu, Y., Terechko, A., Groote, J.F., and Saberi, A.K., "A Formally Verified Fail-Operational Safety Concept for Automated Driving," *SAE Int. J. of CAV* 5(1):7–21, 2022, doi:10.4271/12-05-01-0002.

ISSN: 2574-0741
e-ISSN: 2574-075X

1. Introduction

The safety of autonomous vehicle passengers and other road users depends on Automated Driving (AD) systems. To increase traffic safety, governments and the automotive industry formulated Vision Zero [3] with the goal of completely eradicating road fatalities in the future. To design an AD system, automotive safety engineering involves a reliable product lifecycle management, safety analysis techniques, and design patterns. State-of-the-art safety standards and guidelines (ISO 26262 [1], ISO/PAS 21448 SOTIF [2], SaFAD [13], RSS [15], IEEE P2846 [16], etc.) are developed to reduce risks due to system malfunction, fault-free hazardous behavior, and wrong driving decisions. Our work contributes by providing a holistic fail-operational design pattern and exploring the promising practice of formal verification applied to a novel highly resilient degradation policy.

The International Organization for Standardization (ISO) 26262 standard [1] focuses on system malfunctioning due to random hardware faults and systematic hardware and software faults. According to this standard, the required integrity of AD elements is indicated by the Automotive Safety Integrity Levels (ASILs). Elements with higher ASILs are more costly to develop. Commercially viable AD systems favor cost-effective solutions with fewer high ASIL elements. ASIL decomposition [1] process can help reduce the required ASIL, given that there is sufficient architectural, hardware, or software independence [4].

Traditional automotive systems required only fail-safe mechanisms; however, AD applications (especially automation level 3+ [5]) require fail-operational capabilities. Developing fail-operational AD systems requires complex safety analysis such as dependent failure analysis and ensuring resilience to multiple-point faults. In this article, we analyze the handling of multiple independent faults, while keeping the system operational. Our proposed solution provides fail-operational capabilities while limiting the number of high ASIL elements to optimize the cost [13]. Upon detection of a fault in the nominal subsystem, our concept uses a degradation policy switching to a redundant healthy subsystem to continue AD.

Furthermore, the AD applications need situational awareness in the absence of faults to monitor its own performance limitations as specified in the Safety Of The Intended Functionality (SOTIF) [2]. A common safety measure against performance limitations is to follow a degradation policy depending on the Operational Design Domain (ODD) [13, 28]. An ODD specifies conditions under which the AD application is designed to correctly control the vehicle such as the road type, speed range, and weather and light conditions [29]. Note that AD components may have different performance in different ODDs (e.g., components that are based on machine learning) and may not always be safe to use.

Flawless decision-making for vehicle control becomes crucial at high speed and acceleration because there is no time slack to compensate for inaccurate actuations later. In an emergency scenario, such as an evasive maneuver or pre-crash, it is not sufficient to rely on low-coverage testing and validation techniques. Noteworthy, in the post-crash corner case, the vehicle may have suffered multiple faults due to a collision. Yet the AD system needs to reliably identify working subsystems and use them to carry out a minimum risk maneuver, such as a safe stop.

Formal verification techniques [24] are only "recommended" and not "highly recommended" by ISO 26262, which rather focuses on traditional methods, such as failure analysis techniques, testing, and validation. However, the traditional techniques are not exhaustive and not time effective [14], resulting in late identification of safety issues in the development, in the field, or even their complete oversight. In contrast, formal model checking reduces costly specification errors by exhaustively proving safety properties [17]. Furthermore, ISO 26262 part 6 does highly recommend semi-formal notations involving system modeling and gives guidelines on model-based development in Annex B. It is also noteworthy that an Institute of Electrical and Electronics Engineers (IEEE) P2846 industrial working group [16] is currently defining a formal model for automated decision-making using mathematical formulas of vehicle kinematics.

We propose a safety concept based on a Distributed Safety Mechanism (DSM) for AD with various degraded modes capable of handling multiple-point faults. To verify our approach, we model the DSM behavior using the formal specification language mCRL2 [25]. The safety requirements of the DSM model are defined, formalized, and formally verified.

This article makes the following contributions:

1. A DSM-based, cost-effective fail-operational safety concept (including behavioral specification) to handle multiple-point faults and performance limitations using degraded modes and allocation of fault models and monitoring techniques to different safety layers of the DSM.
2. A formal specification of the DSM model using the mCRL2 language and formally verified safety requirements for the DSM degraded modes.

The rest of the article is organized as follows. In Section 2 we give an overview of related work. Then we describe the DSM architecture in Section 3. We discuss the allocation of fault models and monitoring techniques to our DSM safety layers in Section 4. We elaborate on the fault handling and degraded modes behavior in Section 5. The DSM model and the formal verification process are presented in Section 6. Finally, we conclude in Section 7.

2. Related Work

In this section we present an overview of related AD systems' safety architectures and the application of formal modeling in the automotive domain that are most relevant to our work.

2.1. Safety Architectures for Automated Driving

A DSM using hypervisors and different middleware software stacks was proposed in [4]. The safety mechanism was implemented on a hardware-in-the-loop setup with an AD simulator. The proposed DSM in [4] is evaluated by proof-of-concept experiments on the hardware-in-the-loop simulation to be able to detect single-point faults in the AD system and safely stop the vehicle when necessary. Our work inherits the layered architecture from [4] and the employment of software middleware [20] and hypervisors. However, in contrast to [4] which focused on a proof-of-concept implementation of the DSM architecture, our work refines the safety concept to enable more degraded modes. Furthermore, we model and formally verify the logic of the DSM fault-handling behavior.

A scalable architecture for an AD system is proposed in [7]. In this architecture there is one primary ASIL channel that provides advanced autonomous driving functionalities and comfort for the passengers. On the same platform, there is a second ASIL channel with less advanced functionalities. But it has, for example, safer trajectory plans and is, therefore, safer than the primary channel. A selector is implemented to switch between the two ASIL channels. Besides, a fail-degraded channel is implemented on a separate platform with access to redundant braking and steering. Scalability and integration are the main focus of this design. Compared to [7] that relies on redundant actuators for degraded operation, our DSM relies on redundant communication networks, middleware stacks, hypervisors, and various monitoring techniques to enable degraded modes of AD system operation.

A fail-operational architecture, which remains operational in the presence of faults [9, 10], is proposed in [8]. It has two dual-channel domain controllers. Each of the four channels consists of a chain of sensor systems, AI-based data processing and control, and actuator systems. Note that only the data processing and control system is independent for each channel, all the other subsystems are shared among the channels. The secondary channel in our DSM is more isolated in this sense because it has a different AD system functionality implementation and its data comes from dedicated safety sensors. In [8], each channel has multiple monitors and they all report to a global safety/fault manager in the same channel. This centralized channel safety manager is responsible for fault containment and response functions in the channel. Similar systems are the centralized health monitor implemented in the Baidu Apollo AD framework [23] and a centralized AD system Mode Manager described in [13]. On the contrary, our DSM supports distributed monitors in all safety layers. The fault detection is reported not to a centralized monitor but to a diagnostic message topic provided by the middleware software. Another interesting point [8] made is that time required to find the minimal risk safe stop is indeterminant and could take several minutes. Our DSM copes with this challenge by providing further degradation even during certain degraded modes of the AD system.

Several system-level safety systems for truck platooning architectures applying the safety executive pattern [11] were discussed in [9] and [10]. The described architectures implement two heterogeneous channels running on different platforms. Voting mechanisms arbitrate between the two channels. This heterogeneous duplex pattern [11] is fail-operational in the case of a single failure in one of the channels [9]. In the DSM we also implement a heterogeneous safety channel next to the nominal channel and extend fault handling to multiple-point faults.

In [12] a monitor/actuator architecture for autonomous vehicles, also known as the doer/checker architecture, is described. This architecture depends on an ideal monitor to detect a failure in the primary system. Then a simple and high-integrity failover system can bring the vehicle to a safe state.

The multiple degraded modes of operation enabled by our DSM are similar to the failover mission described in [12]. The advantage of the DSM is that by making use of the layered architecture and different monitoring techniques, we can still guarantee fail-operation even if the monitor becomes faulty.

A functional generic architecture for AD systems is proposed in [13]. There are one or multiple AD system Mode Manager modules that switch the AD system from nominal to degraded operation based on information received from multiple monitors. While [13] describes its framework at a fairly high level, we present our DSM with both sufficient low-level hardware details and high-level safety mechanism behavior and allocation of fault models and mitigation techniques to the DSM architecture.

2.2. Modeling and Formal Methods for Automated Driving

The Responsibility-Sensitive Safety (RSS) [15] and the related IEEE P2846 working group [16] define a mathematical model for safe driving decisions. The model enforces longitudinal and lateral distance between the vehicles, the right of way, and evasive maneuvers. It defines a “Safe State” designed to prevent the autonomous vehicle from being the cause of a road accident from the legal perspective. Note that RSS assumes that the AD system never malfunctions and has no performance limitations, for example, the perception subsystem perfectly detects all road users. Furthermore, the RSS model is technology neutral, omitting safety-critical effects in the System-on-Chip (SoC), networks, and software. Our safety concept incorporates RSS as a crucial component for driving decision-making and addresses RSS limitations with system monitoring and fault-handling techniques.

In [18] a simplified fail-operational model of a brake-by-wire system is given using SysML [19] diagrams, and the transitions are analyzed using activity diagrams. There is arbitration logic to activate the fallback channel in case of a failure in the nominal channel. Faults in one channel are assumed to be independent from faults in the other channel. There is one emergency operation mode and a driver takeover mode in case of a failure. Both channels in [18] are fail-silent, which means they shall become silent and produce no output at all in case of a failure without interfering with any components, such that failure propagation is prevented at the component level. Our safety concept assumes only one safety layer to be fail-silent and relies on Virtual Machines (VM) to prevent fault propagation [6]. Because each VM is an isolated environment. The VM can be paused or shut off once it is identified to be faulty. Also more degraded modes are implemented in our DSM to handle multiple-point faults. [32] models an example AD system that is small yet sufficiently complex using a novel Domain-Specific Language. [32] illustrates the modeling approach that abstracts away from the actual system details while keeping system properties and restrictions that have the potential to be formally verified. Our work not only models our safety concept for an AD system but also formally verified it. Fault Tree Analysis (FTA) is used in [18] to deduce a fault tree failure model based on the state machine and the activity diagrams of the system. The result is a safety framework compliant with ISO 26262 [1]. Although no formal methods are applied in [18], the necessity of proving the correctness of the arbitration logic and deadlock-free property of the system is emphasized, and formal verification is proposed as the solution to prevent failure in the logic of the state machine controlling fail-operational driving.

In AUTOSAR (AUTomotive Open System Architecture), an open and standardized automotive software architecture, there is an Electronic Control Unit (ECU) State Manager module responsible for the determination of the initialization and de-initialization state of AUTOSAR applications, such as INIT, OFF, ON, and RESET status [33]. In our work we also use a state machine to model the safety concept. However, our state machine transitions are not only triggered by the change of ECU initialization status but also the diagnostic data of the AD function behaviors. In addition, corresponding safety mechanisms are triggered along the state machine transitions as well.

A framework to capture the relation between AD system design and functional safety is proposed in [14]. While acknowledging the limitation of formal methods such as scalability and the reality gap between the model and the actual implementation, [14] discussed the strengths and benefits of formal verification in the automotive domain. In particular, they suggest applying formal methods to address hazardous situations, which are not covered by ISO 26262 and SOTIF.

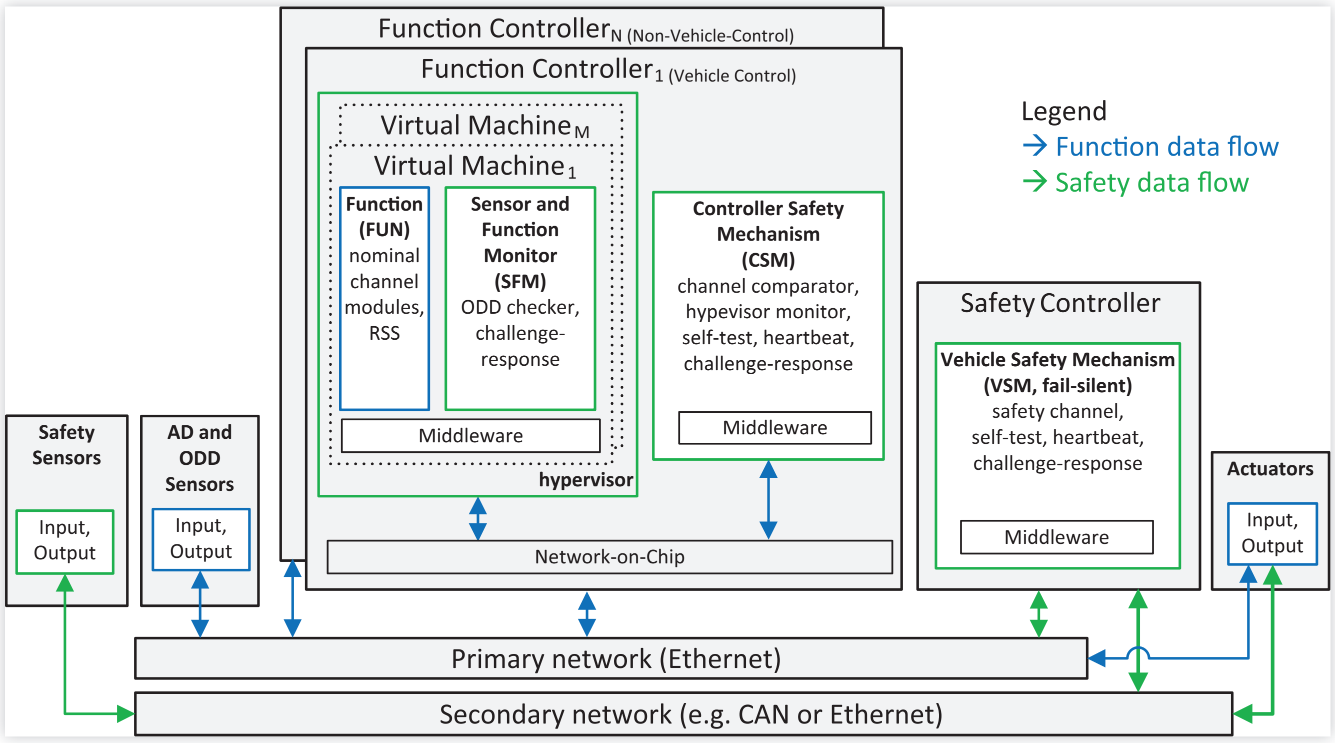
3. Architecture of the Safety Mechanism

In the following subsections, we detail the DSM architecture and component functionalities in our safety concept.

3.1. Functionalities of the Distributed Safety Mechanism

Figure 1 illustrates the high-level architecture of our DSM. Note that we do not limit the architecture to any specific

FIGURE 1 The architecture of an AD system with the distributed safety mechanism.


Function Controller N (Non-Vehicle-Control)
Function Controller 1 (Vehicle Control)
Virtual Machine M
Virtual Machine 1
Function (FUN) nominal channel modules, RSS
Sensor and Function Monitor (SFM) ODD checker, challenge-response
Middleware
hypervisor
Controller Safety Mechanism (CSM) channel comparator, hypevisor monitor, self-test, heartbeat, challenge-response
Middleware
Network-on-Chip
Legend
→ Function data flow
→ Safety data flow
Safety Controller
Vehicle Safety Mechanism (VSM, fail-silent) safety channel, self-test, heartbeat, challenge-response
Middleware
Safety Sensors
Input, Output
AD and ODD Sensors
Input, Output
Actuators
Input, Output
Primary network (Ethernet)
Secondary network (e.g. CAN or Ethernet)

implementation but only give implementation examples for illustrative purposes. The component is described below.

a. AD and ODD Sensors: There are AD sensors and ODD sensors in the AD system. The AD sensors provide data via the primary network to particular AD function modules on the Function (FUN) layer. For example, cameras and radar sensors provide data to the perception function module, while the global navigation satellite system feeds the localization module. At the same time the AD sensor data are monitored by the Sensor and Function Monitor (SFM) layer. The intention of the ODD sensors is to provide input via the primary network to an ODD checker integrated into the SFM layer.
b. Safety Sensors: There are hot standby sensors sending data for safe maneuvers via the secondary network to the Vehicle Safety Mechanism (VSM) layer in the safety controller.
c. FUN Layer: In the FUN layer the AD function modules, such as localization, perception, path planning, etc. are running. The FUN layer has access only to the primary network channel. One or multiple modules with related functionalities run in the same VM. Note that safety-related decision-making algorithms, such as RSS [15], are typically integrated into the path planning module, as it is the case in Baidu Apollo platform [23].
d. Middleware: Middleware is software that enables the various components of a distributed system to communicate and manage data reliably [12]. Communication reliability and visibility are guaranteed by middleware Quality of Service policies, such as real-time deadline, priority, and reliability [22]. Note that the middleware communication protocol has to be free of a single point of failure; see, for example, the Data Distribution Service standard [22]. In our DSM, we assume information is shared on top of the reliable middleware that uses the publish-subscribe communication pattern [20].
e. SFM Layer: Each VM has an SFM layer. The SFM monitors the status of the FUN layer in the same VM. Furthermore, it detects faults in the corresponding AD sensor inputs and outputs, such as timing jitter and out-of-range message data. The SFM layer also acts as an ODD checker. Based on the ODD sensor outputs, it detects when the driving situation changed [29] and triggers necessary AD system mode transitions to avoid unsafe usage of components with limited performance, such as neural networks in the AD system perception subsystem.
f. Controller Safety Mechanism (CSM) Layer: Each SoC in Figure 1 is annotated as Function Controller and has a dedicated CSM layer. The SoC with vehicle control functionalities is termed the Vehicle Control (VC) function controller. The others are called Non-Vehicle-Control (NVC) function controllers. NVC CSMs only monitor their local SFM layers, hardware, and hypervisors; the VC CSM monitors all the VC and NVC function controllers and the VSM layer. The CSM layer has access to the primary network channel and can send control commands to the vehicle actuators. Note that the CSM layer can compare channels' outputs to identify disagreement between the nominal and safety channels.
g. VSM Layer: The VSM layer runs on a separate safety controller. It monitors the VC CSM, the safety sensor data, and the two off-chip networks. The VSM layer can maneuver the vehicle via the secondary network using the safety sensor data. In addition, we assume the VSM layer has access to the power management ICs.
h. Network-on-Chip (NoC). The NoC is a low latency communication interconnect for hardware Intellectual Property modules on an SoC.
i. Primary Network: This is the high-performance network channel for nominal control of the AD system.
j. Secondary Network: This hot standby of the primary network is a heterogeneous high-performance redundant communication channel for safety control of the vehicle.

3.2. Cost-Effective Fail-Operational Architecture

In the scope of this article, we are not able to provide empirical evidence regarding cost-effectiveness of our proposed approach. However, we are able to qualitatively compare our approach to other well-known patterns. The Triple Modular Redundancy (TMR) design pattern [11] is commonly used in aerospace systems, such as the Boeing 777 airplane flight control system [31]. TMR consists of three costly redundant channels and a reliable voter. This pattern provides high degrees of fault resilience and can be fail-operational. However, TMR is considered too expensive for AD systems. Thus dual-channel instead of TMR are commonly used in AD systems, as described in [7, 8, 11]. To further enhance the AD system safety and redundancy, our proposed pattern adds simpler and cheaper emergency and safety channels to the nominal channel. Based on the system health status and environmental awareness, the DSM employs a degradation policy to switch between the channels, which is described in Section 5. The simplest emergency channel does not use sensors at all and merely stops the vehicle. Compared to the nominal channel our safety channel relies on simpler algorithms and fewer safety sensors to maneuver the vehicle, which reduce the development costs for introducing redundancy. Furthermore, this pattern does not assume additional hardware for functional redundancy, which helps reduce production costs. Also, by distributing the safety responsibility to the VSM layer, the function controllers require lower integrity, which contributes to reducing associated costs.

4. Fault Detection and Diagnosis

Thanks to its distributed nature, the DSM can effectively detect and diagnose various fault models by monitoring SoCs, networks, and distributed software of the AD system. In this section, we discuss the responsibilities of the DSM's layers in monitoring fault models. Table 1 presents examples of fault models accompanied with monitoring techniques that the DSM layers can use to detect system malfunctioning. The system components column covers the whole vehicle infrastructure for AD, including sensors, actuators, processors, and networks. Examples of fault models and performance limitations are classified according to the DFA (Dependent Failure Analysis) coupling factor classes from [1]. Performance limitations from [2] provoke hazards when the fault-free AD system cannot safely handle a road situation, for example, because of machine learning implementation restrictions. Note that the presented fault model is not comprehensive, but rather exemplifies the areas of responsibility of the DSM layers. Furthermore, the monitoring techniques column includes examples of how the system can detect or mitigate corresponding fault models. Several monitoring techniques, such as heartbeats and message rate monitoring, depend on the middleware software for distributed coordination of the DSM and AD functions. Finally, the responsible DSM layer is specified along with an indication of the minimum required safety integrity level.

The primary responsibility of the SFM layer is to monitor the AD and ODD sensors and AD functions, such as localization, perception, and path planning. Modern smart sensors integrate complex processing and networking onboard. Therefore, their failure modes span from communication issues (e.g., message publish rate jitter) to software issues (e.g., wrong ego-vehicle localization) to performance limitations of the algorithms (e.g., ghost and missed objects [30]). ECUs, SoCs, and VMs running AD functions will exhibit similar fault models. Examples of monitoring and mitigations techniques include message rate monitoring, data fusion of multimodal sensors, and comparison against an independent safety channel. Furthermore, the SFM layer can perform fast sophisticated diagnosis of the health state of the AD functionality thanks to its proximity to the FUN layers running in the same powerful VM. For example, it can track timing and priority of the published packets, examine packet content using in-range and out-of-range checks, and inspect drivers, the OS kernel, etc.

The CSM layer is responsible for monitoring faults in the function controllers (SoCs) and VSM. Each SoC and the low-level software (e.g., hypervisor) can suffer from various failure modes: on-chip memory corruption, stuck-at faults in the processor core and NoCs, hypervisor isolation failure, firmware deadlock, or priority inversion. These fault models often occur in shared SoC resources and affect multiple components at the higher level of the AD system functionality. Therefore, handling of these fault models requires a higher integrity level than that of the SFM layer. Besides mentioned in Table 1 monitoring techniques the CSM can deploy on-chip hardware monitors for clock jitter, IO errors, control flow monitoring, and lock-step processing and data transfer.

In our DSM the VC CSM monitors all the other NVC function controllers using a challenge-response protocol [21]. The challenge-response protocol covers memory- and processor-related fault models on top of traditional liveliness heartbeats with a watchdog. Furthermore, the CSM also serves as a fall-over mechanism when the VSM layer on the safety controller fails. To detect a fail-silent failure of the VSM, the CSM can simply monitor the VSM heartbeats using a watchdog. The CSM can safely maneuver the vehicle only by cooperating with all the fully operational FUN layers implementing the AD functionality. Without the fully operational FUN layers, the CSM can only send basic emergency stop [13] commands to the vehicle actuators via an emergency channel. Besides virtualization and isolation, the implementation of hypervisors on the function controllers allows the CSM layer to quickly pause the VMs if a fault or a hazardous situation is detected.

The overall vehicle safety is managed by the high-integrity VSM layer. Besides monitoring its own safety controller similar to the CSM, it is responsible for checking the health of the VC CSM, actuators, data, and power networks. Steering, braking, and acceleration actuators share their (health) status similar to smart sensors. This status is continuously analyzed by the VSM. Furthermore, the VSM actively monitors the data networks by observing packet transmissions and querying network gateways and switches. Besides checking for unmet real-time deadlines, the VSM can analyze message priorities and content. Finally, the VSM can control the power distribution network and power management ICs to power off the VC CSM, sensors, and networks, when necessary.

It is noteworthy that besides checking the AD system functionality, the DSM monitors its own components for malfunctioning and has several fail-over degraded modes to cope with faults in all its layers. The multiple SFM and CSM layers are allowed to fail arbitrarily and, hence, have only low and medium safety integrity requirements, which help reduce the overall development cost, silicon area, and power consumption. However, the VSM layer with modest computation and networking requirements does need a high integrity level to reliably manage other safety layers in the AD system.

5. Degraded Modes of Operation

AD availability [13] can be defined as the ratio of the safe AD time without disengagements to the total requested AD time. The AD availability can be increased by extending supported ODDs, by decreasing fault probabilities, and, after a fault occurred, by graceful AD service level degradations. Whenever a severe fault is detected, our safety concept changes the automated operation mode to sustain the AD availability instead

TABLE 1 Allocation of example fault models and monitoring techniques to the DSM layers.

System component	Example fault model and performance limitations	Failure type	Example monitoring or mitigation techniques	DSM layer responsibility	DSM layer integrity
Sensors	Message rate jitter	Communication	Message rate monitoring	SFM, safety of AD functions	Low
	Wrong ego-vehicle localization	Component failure	Fusion with heterogeneous sensors, comparison with safety channel		
	Missed or ghost object, or incorrect location of a detected object	Component failure	Fusion with heterogeneous sensors, comparison with safety channel, ODD checker		
AD functions (localization, perception, prediction, planning, control)	Incorrect world model perception	Component failure	Comparison with safety channel		
	Incorrect ego-vehicle localization after sensor fusion	Component failure	Comparison with safety channel		
	Path planning incurs collision	Component failure	Comparison with safety channel		
	Incorrect prediction for non-ego vehicle	Component failure	Comparison with safety channel		
	Vehicle actuator command drives vehicle off-road	Component failure	Comparison with safety channel		
	Vehicle actuator command results in loss of vehicle control	Component failure	High-integrity (software) design		
	Application deadlock or crash	Component failure	Heartbeats, watchdogs		
Function controllers (SoCs)	Memory corruption	Shared resource	Error-correction and detection codes, memory BIST	CSM, safety of the function controller hardware and platform software (hypervisor, OS, firmware)	Medium
	Processor core failure	Shared resource	Software self-test, logic BIST		
	OS kernel panic, driver crash, hard fault	Shared resource	Heartbeats, watchdogs		
	Firmware crash or deadlock	Shared resource	Diagnostic messages with watchdog		
	Out-of-memory error or memory leak	Shared resource	Diagnostic messages with watchdog		
Hypervisor	CPU scheduler is stuck	Shared resource	Heartbeats, watchdogs		
	Wrong VM scheduling	Shared resource	Diagnostic messages with watchdog		
	Isolation failure (e.g., TLB failure)	Component failure	Software self-test, logic BIST		
DSM SFM	Arbitrary fault (timing, message content)	Component failure	Diagnostic messages with watchdog		
DSM VSM	Silence or wrong diagnostic response	Component failure	Diagnostic messages with watchdog		
Safety controller (SoC)	Memory corruption	Shared resource	Error correction and detection codes, memory BIST	VSM, vehicle safety, including monitoring of data and power networks integrity	High
	Processor core failure	Shared resource	Software self-tests, logic BIST		
	On-chip IP failure	Component failure	On-chip safety monitors		
	On-chip network failure	Component failure	On-chip safety monitors		
	SoC IO interface failure	Component failure	On-chip safety monitors		
	Software fault models	Component failure	Control flow checks		
Actuators	No status reading	Component failure	Heartbeats, watchdogs		
	Wrong status reading (e.g., wheel speed)	Component failure	High-integrity sensor design		
Data networks	Corrupted packet	Communication	Heartbeats, watchdogs		
	Short-circuit, open wire, stuck-at fault	Communication	Heartbeats, watchdogs		
	Message timing violation	Communication	Packet transmission monitoring		
	Message priority violation	Communication	Packet transmission monitoring		
Power network	Outage, undervoltage, overvoltage, short-circuit	Systematic coupling	Power management IC status checks		
DSM CSM	Arbitrary fault (timing, message content)	Component failure	Challenge-response with watchdog		

© SAE International

of disengaging. The operation modes are defined by the driving subsystem, the driving goal, passenger comfort, and fault tolerance:

1. Nominal. In this operation mode the vehicle arrives at the planned destination following the planned trajectory.
2. Detour. Instead of driving to the destination, in this mode the vehicle detours to the nearest car repair shop for the repairment of the detected faults in the AD system.
3. Comfort Stop. In this operation mode the AD system stops the vehicle on a safe side of the road by performing a graceful pull-over.
4. Safe Stop. In this mode the AD system performs a pull-over or only an in-lane stop depending on the system availability.
5. Emergency Stop. In this mode the AD system bluntly brakes and stops the vehicle.

Transitions between the modes are governed by our degradation policy shown in Figure 2. In this figure the labels on each transition represent the trigger of the degradation. The label “comfort” indicates the passenger comfort provided by the driving subsystem; “resilience” denotes the fault tolerance level of the vehicle. When the AD system is in the fault-free Nominal mode, the VC FUN layer is controlling the vehicle. All the safety layers are performing different monitoring tasks at the same time. Once a fault is detected by any of the safety layers, the DSM activates a degraded operation mode based on the current AD system status. Note that a degraded operation mode doesn’t require costly replication of the nominal system, yet it is sufficient to drive the vehicle to a safe location despite multiple faults.

In the following subsections, we describe the DSM behavior in each operation mode.

5.1. Nominal Mode

Figure 3 illustrates the behavior of the AD system with our DSM in the fault-free Nominal mode. In this mode all the AD system components are fully functional. The vehicle is controlled by the VC FUN layer in the nominal channel. Meanwhile the safety channel is on hot standby. The VSM layer receives and monitors the safety sensor data via the secondary network channel, but does not send any control command to the actuators. All types of monitoring between safety layers are going on as well:

a. All the AD sensors, ODD sensors, and FUN layers are monitored by the corresponding SFM layers. The safety sensors are monitored by the VSM layer.
b. Each SFM layer and the NoC are monitored by the CSM layer on the same function controller.
c. All NVC function controllers are monitored by the VC CSM layer (not shown in Figure 3 for simplicity).
d. The VSM layer monitors both networks.
e. The VC CSM and the VSM layers monitor each other using the challenge-response protocol [21] through the primary network.

5.2. Detour Mode

The AD system nominal channel is fully functional in the Detour mode. Upon detecting a fault in the safety channel, the VC CSM layer is able to make the vehicle detour to the nearest car repair shop without sacrificing passenger comfort. Since the monitoring activities in the nominal channel work properly, the AD system can further degrade to the Emergency Stop mode if faults or a hazardous situation is detected in the nominal channel as illustrated in Figure 2.

FIGURE 2 DSM degradation policy.

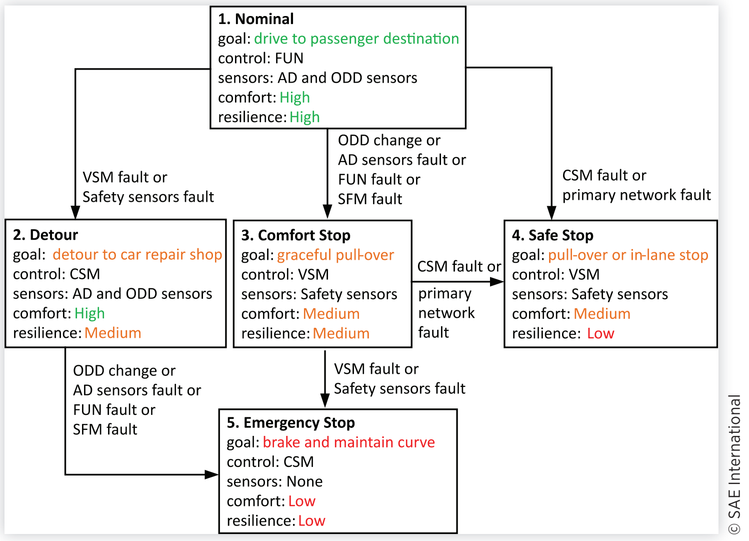
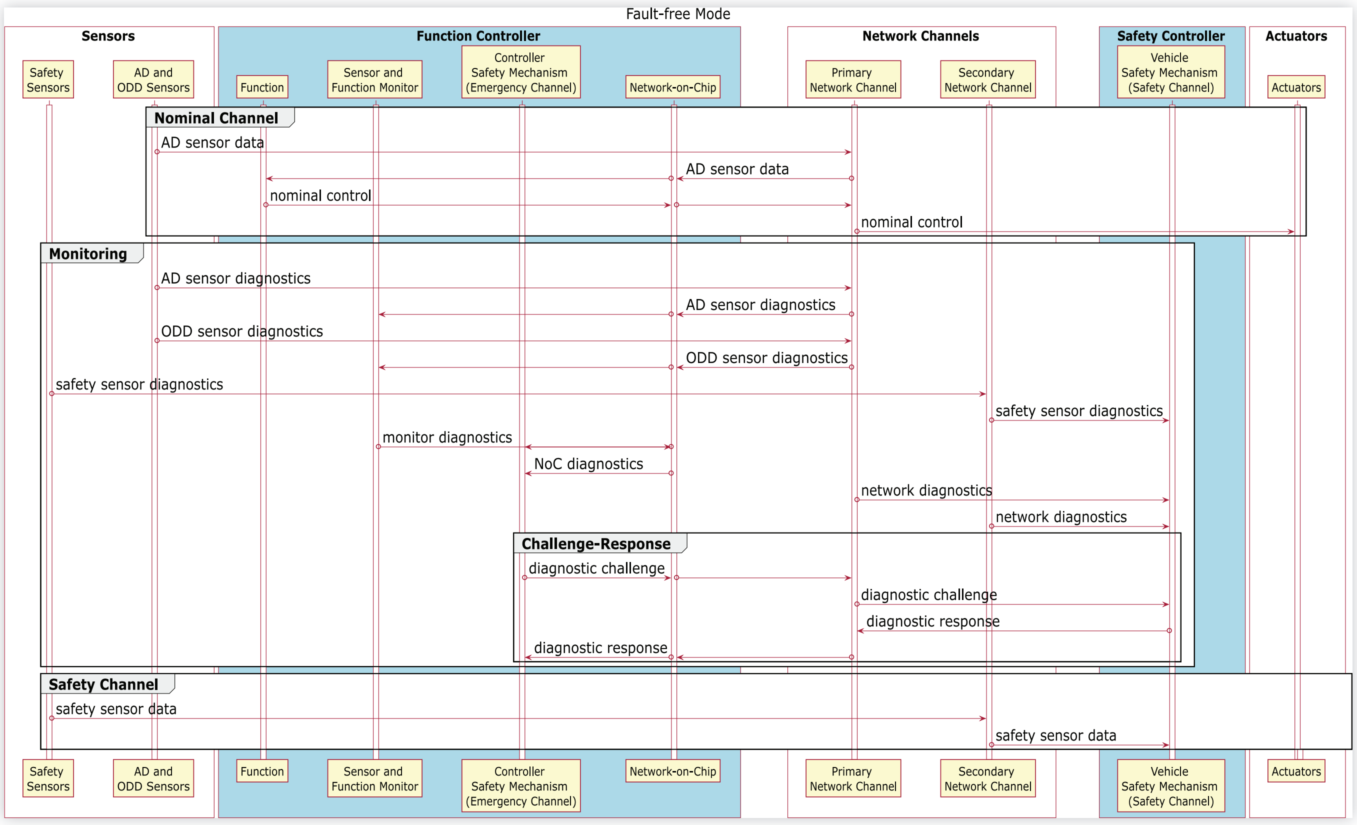

1. Nominal
goal: drive to passenger destination
control: FUN
sensors: AD and ODD sensors
comfort: High
resilience: High
VSM fault or Safety sensors fault
ODD change or AD sensors fault or FUN fault or SFM fault
CSM fault or primary network fault
2. Detour
goal: detour to car repair shop
control: CSM
sensors: AD and ODD sensors
comfort: High
resilience: Medium
3. Comfort Stop
goal: graceful pull-over
control: VSM
sensors: Safety sensors
comfort: Medium
resilience: Medium
CSM fault or primary network fault
4. Safe Stop
goal: pull-over or in-lane stop
control: VSM
sensors: Safety sensors
comfort: Medium
resilience: Low
ODD change or AD sensors fault or FUN fault or SFM fault
VSM fault or Safety sensors fault
5. Emergency Stop
goal: brake and maintain curve
control: CSM
sensors: None
comfort: Low
resilience: Low
© SAE International

FIGURE 3 Fault-free nominal mode of the DSM.



5.3. Comfort Stop Mode

The AD system safety channel is fully functional but only safety sensors are used in the Comfort Stop mode. The VSM layer in the safety channel can control the vehicle to do a graceful pull-over and guarantee sufficient passenger comfort. As Figure 2 shows, while in this mode the AD system can degrade further to the Safe Stop mode when the VSM detects a fault in the primary network or the VC CSM or to the Emergency Stop mode when VC CSM detects a VSM fault.

5.4. Safe Stop Mode

In this mode the AD system safety channel is still fully functional, but the nominal channel cannot perform any degradation mode anymore due to the VC CSM fault or primary network fault as shown in Figure 2. Because we assume components on the function controller are fail-arbitrary, to perform the safe stop the VSM first powers off the VC function controller to prevent it from producing any arbitrary output, then it starts the Safe Stop procedure to quickly pull over the vehicle or even performs an in-lane stop when necessary. We assume in this mode the vehicle is immediately in a safe state.

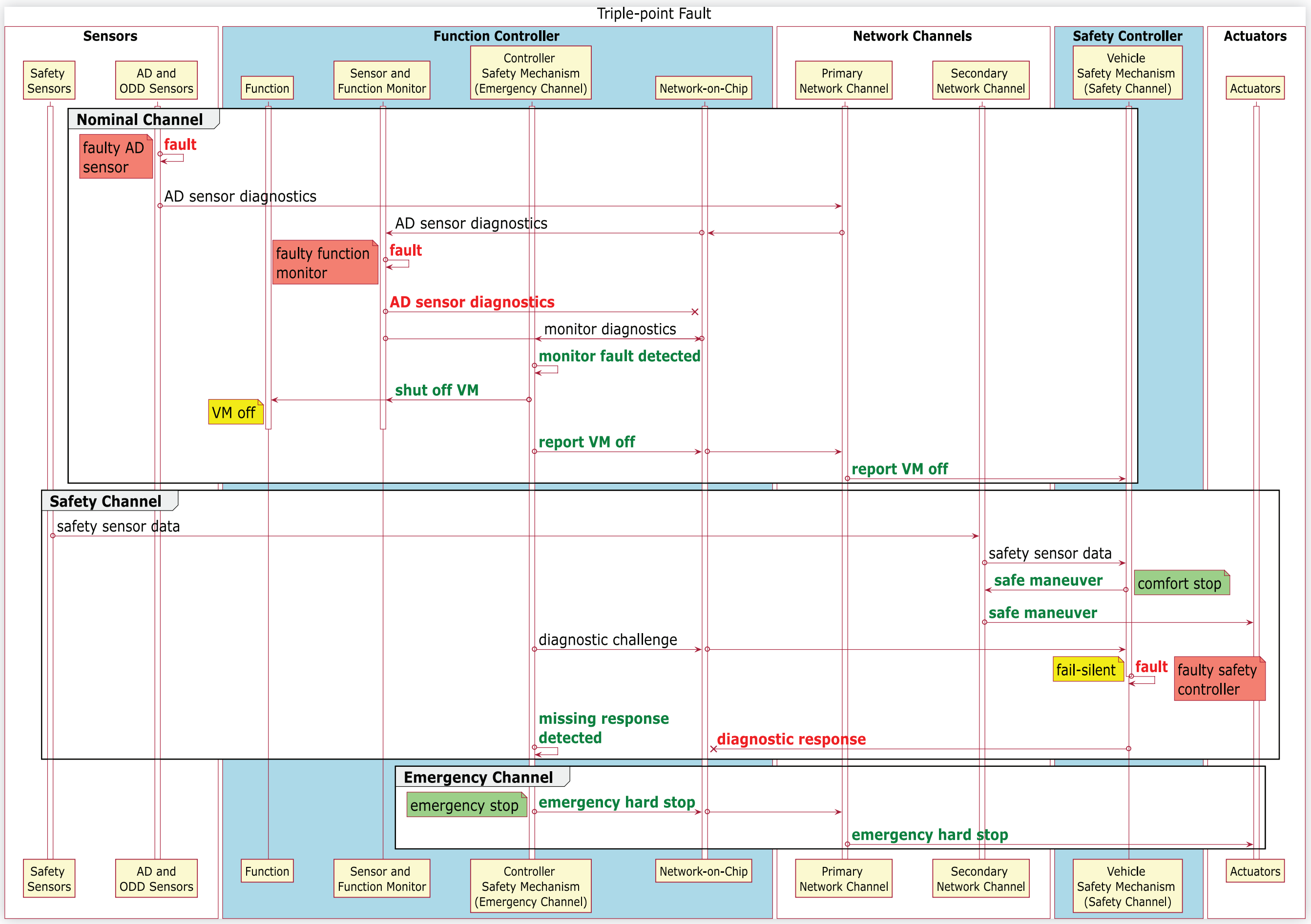
5.5. Emergency Stop Mode

The Emergency Stop mode can be reached by degrading from the Detour mode or the Comfort Stop mode. In this mode, the AD system nominal channel is only partially functional, which means the VC CSM does not have all information needed for the advanced Detour operation but can still control the vehicle via the primary network. In addition, the AD system safety channel in the Emergency Stop mode cannot perform any degradation anymore due to the VSM or safety sensor faults as shown in Figure 2. Therefore, the VC CSM simply sends braking commands to the vehicle actuators to blindly stop the vehicle as quickly as possible. Obviously, the passenger comfort cannot be guaranteed during this procedure. We assume the vehicle is immediately in a safe state in this mode.

5.6. Example Use Cases

As shown in Figure 2, the AD system can be degraded from the Nominal mode first to the Comfort Stop mode and then to the Emergency Stop mode. Such a use case is shown in Figure 4. At first a fault occurs in an AD sensor. Note that the fail-arbitrary AD sensor can provide arbitrary output to the FUN layer, which results in an arbitrary FUN control output. But the faulty control can reach the vehicle actuators without violating the safety goal if it is quickly suppressed by the safety mechanism. In this case, the AD sensor fault is detected by the SFM layer, but then a second fault occurs in the SFM layer before it is able to report to the CSM layer. Nevertheless, the CSM layer detects the SFM layer malfunction and reacts to it by shutting down the vehicle control VM where the faulty SFM layer is running and reports the VM state to the VSM

FIGURE 4 Multiple faults handling: first an AD sensor fault then a function monitor fault then a vehicle safety mechanism fault.



layer. Subsequently, the VSM layer degrades the AD system to the Comfort Stop mode. Before the vehicle reaches the safe state; however, a third fault occurs in the VSM layer. Thanks to the challenge-response monitoring technique, the CSM layer detects the VSM layer fail-silent fault and further degrades the AD system to the Emergency Stop mode.

When faults occur in the AD function modules, our DSM makes the AD system degrade from Nominal mode to Comfort Stop mode as shown in Figure 2. As listed in Table 1, a FUN layer fault could be a deadlock in the AD path planner or a perception module failure, etc. Figure 5 shows how the DSM handles, for example, an AD perception module fault. Since the FUN layer is monitored by the SFM layer, the SFM layer will perform a plausibility check on the perception module output. The faulty perception module output will fail the plausibility check on the SFM layer. Then the SFM layer will report the fault to the CSM layer via middleware, which then reacts to the diagnostics within Fault Tolerant Time Interval by shutting down the VM where the faulty FUN layer is running and reports the VM state to the VSM layer. Finally, the VSM layer takes over control and degrades the AD system to the Comfort Stop mode.

We demonstrate how the DSM is capable to handle a SOTIF [2] scenario in Figure 6. In this case, the AD system degrades from the Nominal mode to the Comfort Stop mode when there is no systematic fault, but the ODD checker on the SFM layer detects that the ODD has changed and adequate performance of the AD system cannot be guaranteed.

6. Model and Formal Verification

With the growing complexity of the AD system, traditional safety analysis such as Failure Mode and Effects Analysis, FTA, as well as road testing and simulation methods remain mandatory, yet become insufficient to guarantee the correctness of the AD system, because it is infeasible to exhaustively analyze or test the numerous system states. Modeling and formal verification, however, can help exhaustively and mathematically prove the desired safety properties of the AD system model, reducing specification and requirement errors in the early design phase. Furthermore, certifiable code can be generated from the formally verified model. The executable model can serve as a golden reference throughout the product life-cycle and help clarify communication among design stakeholders.

In this work we modeled the DSM architecture shown in Figure 1 using the mCRL2 formal specification language [25] and formalized the safety requirements of the model in the

FIGURE 5 Handling a fault in the AD function.

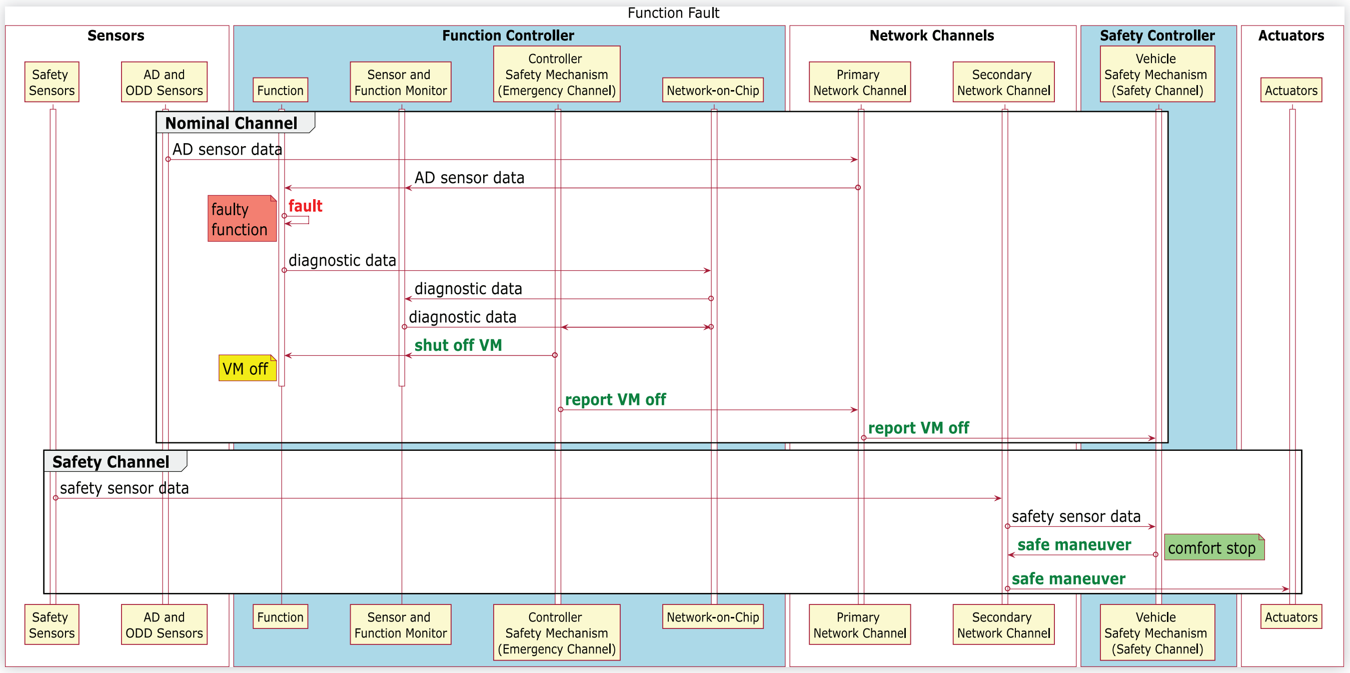


FIGURE 6 Handling of a SOTIF scenario when leaving the ODD.

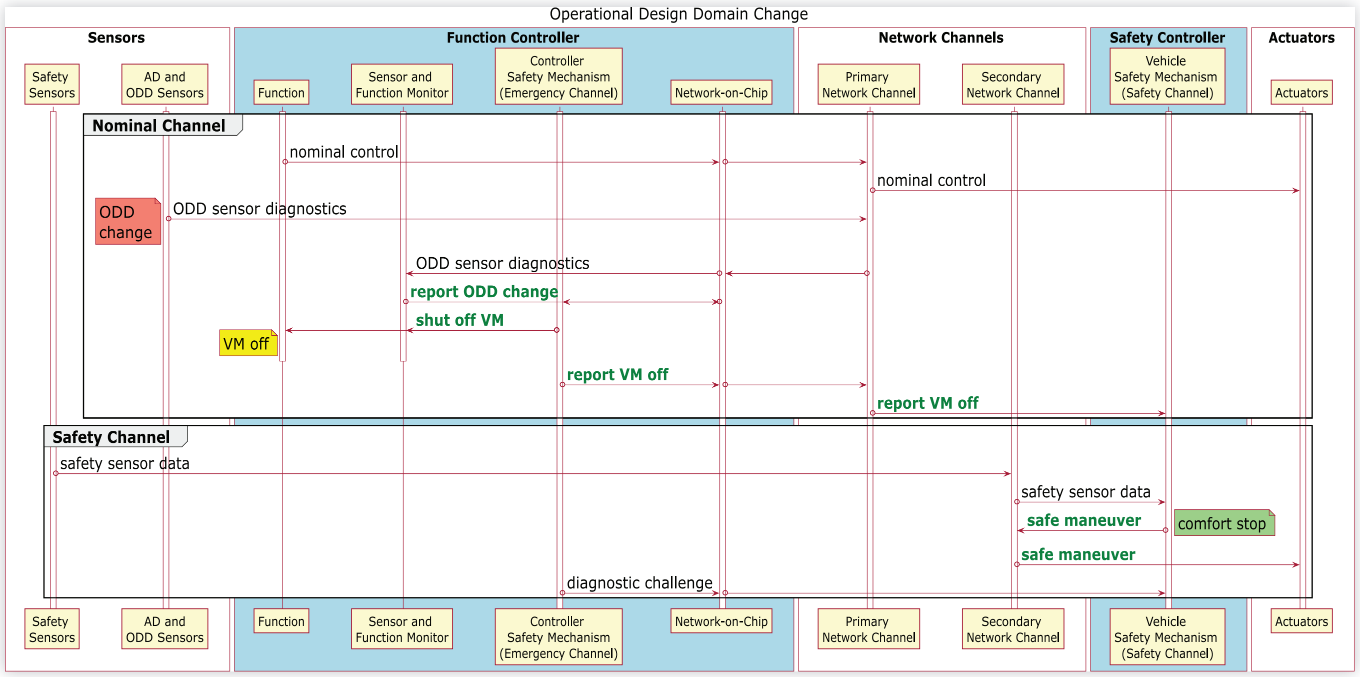


TABLE 2 State spaces of the DSM state machines.

State machine	# states	# transition triggers	# transitions
AD/ODD sensors	3	4	3
Safety sensors	3	4	3
FUN	6	9	32
SFM	8	10	58
CSM	10	23	167
VSM	10	10	39
Actuators	5	6	23

modal μ-calculus [26]. Each and every component in Figure 1 is modeled as FSMs. The FSM transitions specify the DSM behavior in monitoring and fault-handling procedures, which is illustrated in Figure 2 with the high-level degradation policy diagram and in sequence diagrams in Figures 3 to 6.

The model consists of 746 lines of mCRL2 code and 71 lines of μ-calculus formulas. The mCRL2 model is also executable, and its behavior can easily be simulated in the mCRL2 simulator. Table 2 shows the state space of each FSM in our DSM model. The AD system model has all these FSMs run in parallel, resulting in a state space of more than 2 million states and 2 billion transitions. The mCRL2 tool is able to mathematically prove the properties of this model while it is intractable to exhaustively test all these transitions using traditional road testing or an AD simulation setup. In the following subsections, we present details of the DSM model.

6.1. Model Specifications

The AD system components communicate using the software middleware stacks implementing the publish-subscribe pattern [20]. Thus it is essential to properly model the publish-subscribe communication behavior. In our DSM model, information is shared as messages. Any participant on the same network can receive messages by subscribing to a message topic. We model two types of networks in the DSM, namely, the NoC and the off-chip networks. The DSM layers communicate using three message types:

1. Sensor Data Messages. These messages contain the sensor data communicated in the AD system.
2. Vehicle Control Command Messages. These control commands are sent to the vehicle actuators by the VC FUN layer, the VC CSM layer, or the VSM layer.
3. Diagnostic Messages. These messages report the status of the monitored component.

All the messages consist of an action carrying four identification parameters. For example, the action of the vehicle control CMS layer sending a diagnostic message via the primary network channel reporting the VC VM being shut off is modeled as send(VC, VM_VC, VC_VM_OFF, primary).

The four parameters from left to right are explained in Table 3. For sensor data communication, cid and vid identify the receiver component. For power management actions,

TABLE 3 Message parameters and examples.

Parameter	Description	Parameter examples
cid	controller ID or the name of another component	FUN, SFM, VC (short for VC CSM), NVC (short for NVC CSM), VSM, SAFE_SENSOR, NETWORK.
vid	virtual machine ID	VM_VC (the VM running VC FUN), NULL (if the SoC runs no VM), VM_1, VM_2, etc.
info	message content or environmental fault identification	VC_VM_OFF, NOMINAL, Sens_OK, EMERGENCY_STOP, FUN_FAULT.
network	network channel identification	primary, secondary, NoC, power.

network is set to power. Actions modeled using the same structure are send and receive for communication via the off-chip networks, and NoCsend and NoCreceive for communication via the NoC.

All the AD system behavior is modeled by actions in the model. We present a few examples here:

- network_ingress, network_egress: The ingress and egress phases of the network communication.
- fault_injection_*: These actions model the occurrences of faults. The * can be replaced by any of the target components. The events in Figures 4 to 6 next to the red labels are of this type.
- vc_csm_shuts_off_vc_vm: These actions model the VC CSM shutting off the VC VM it manages. It is illustrated in Figure 4 as "shut off VM."
- vsm_powers_off_vc_controller: This action models the VSM layer powering off the VC CSM SoC.

6.2. Model Assumptions

We made the following model assumptions to clarify the scope of our study and keep the model small enough [27] to be verifiable while maintaining the model's relevance:

a. All faults are permanent, atomic, not safe, and always detected. Fault detection can be improved with adequate monitoring techniques without compromising the model. Transient faults are out of scope in this work.

b. NoCs have infinite capacity, transfer data atomically, and never fail. Note that faults in NoCs can be detected by different monitors in the AD system thanks to the layered architecture in our DSM concept. But fault handling in NoCs is part of our future work.

c. The power supply never fails. Power loss in parts of the AD system can be detected by various methods like heartbeat monitoring or a challenge-response mechanism and therefore can be mitigated with degraded modes thanks to the layered structure of the

DSM concept. However, power management is in general out of our scope.

d. The secondary network never fails and is only accessible by the safety sensors, VSM, and the actuators. We assume the secondary network in the safety channel is more reliable than the nominal channel due to its different and simpler implementation with limited functionalities.
e. All DSM transitions are instantaneous and do not fail.
f. VSM, safety sensors, and both off-chip networks are fail-silent, all the other components are fail-arbitrary.
g. Communications through the off-chip primary and secondary networks are split into ingress and egress phases and have a buffer of three messages. The egress messages are sent out in random order. We set the buffer size to 3 to model the out-of-order ingress and egress data flows and keep the state space small.
h. Only certain combinations of multiple-point faults are allowed as illustrated in Figure 2, such that there is always a degraded mode available to bring the vehicle to a safe state.

6.3. Safety Requirements and Formal Properties

The DSM model safety requirements, which would correspond to the Technical Safety Requirements (Req) according to ISO 26262, are listed below:

Req 1 The AD system is deadlock-free.

Req 2 There is always one and only one non-faulty component is in control of the vehicle.

Req 3 The AD system Nominal mode always degrades to Detour, Comfort Stop, or Safe Stop when necessary.

Req 4 Detour must degrade to Emergency Stop when necessary.

Req 5 Comfort Stop must degrade to Safe Stop or Emergency Stop when necessary.

The DSM safety requirements are formalized using modal μ-calculus formulas [24]. Below we show two examples:

Req 1 [true*]<true>true

Req 4 [**true***.network_egress(VC, VM_VC, DETOUR, primary)]
[**true***] **forall** cid: CTRLR_ID, vid: VM_ID.
[(fault_injection_fun(cid, vid, FUN_FAULT, NoC) ||
fault_injection_sfm(cid, vid, SFM_FAULT, NoC) ||
fault_injection_csm(cid, NULL, CSM_FAULT, NoC) ||
fault_injection_sensor(cid, vid, SENS_FAULT, NoC))]
[!network_egress(VC, NULL, EMERGENCY_STOP, primary)*]
<**true***.network_egress(VC, NULL, EMERGENCY_STOP, primary)
>**true**

6.4. Verification Results

The modeling and verification processes help to reduce specification errors in the DSM design which are hard to detect manually or in road testing. For example, an early version of our DSM model with the network buffer size set to 3 has a state space of 215 million states and 2.92 billion transitions. We found a deadlock in this model after running the verification process using the mCRL2 toolset [25] for more than 10 consecutive days. Noteworthy, when the network buffer size was set to 1 and 2, the same model was verified to meet all the safety requirements. We debugged the issue by stepping through the FSMs transitions in different fault handling procedures. With the help of the mCRL2 simulator, we quickly discovered that the deadlock occurred when the primary network was fully occupied by the diagnostics about the safety sensor fault sent by the VSM; meanwhile the VC CSM detected the VSM fault and tried to activate the Detour mode via the fully occupied primary network. In the previously verified models, however, the diagnostics were not buffered long enough to trigger the deadlock due to the smaller size of the buffer. The issue was resolved by adding missing transitions in the VC CSM process to allow it to receive diagnostics from the VSM layer when the VC command cannot be sent out yet.

The final DSM model configuration had two function controllers each running two VMs and the network buffer size set to three, resulting in a state space of over 1 billion states and 10 billion transitions. All the safety requirements were verified for the final model, which took several days of runtime on a powerful server. Although the verification already hit the compute limits of our formal verification server, there is still a reality gap between the formal model and the actual AD system implementation. Nevertheless, the formal approach helps reduce specification errors by mathematically and exhaustively proving the correctness of the arbitration logic in the early design phase prior to the costly AD system implementation and in-field testing. Therefore we believe we made a reasonable trade-off between the modeling accuracy of the real AD system implementation and the feasibility of formal verification.

7. Conclusion

Our work presents a fail-operational safety concept combining safety measures for fault handling, performance limitations mitigation, and driving decision-making. The adopted safety mechanisms handle multiple-point faults in both the AD system and in the safety mechanisms themselves. The proposed AD system architecture is made cost-effective by using mostly fail-arbitrary components as well as additional emergency and safety channels which are simpler than the nominal channel. Based on the AD system health status and situational awareness, the DSM running on multiple SoCs and VMs activates various degraded modes of AD to keep the vehicle as safe as possible. Furthermore, we present an example allocation of fault models to our safety mechanism layers and discuss monitoring techniques to detect faults and ODD changes. For such a complex system, it is hard and costly to find specification errors by implementing the system or running traditional tests. Thus we modeled our DSM arbitration logic in the mCRL2 language and formally verified five safety requirements in the μ-calculus. Our formally verified highly resilient safety concept can serve as a holistic design pattern for a safe and cost-effective AD system, minimizing road fatalities and reducing costly testing.

Promising future research directions include modeling timing properties, incorporation of the doer/checker architecture [12], computation of failure rates using probabilistic models, handling of transient faults, and quantification of cost-effectiveness of the presented safety concept.

References

1. "Road Vehicles: Functional Safety," ISO Standard 26262-1, 2018.
2. "Road Vehicles: Safety of the Intended Functionality," ISO/PAS Standard 21448, 2019.
3. Whitelegg, J. and Haq, G., "Vision Zero: Adopting a Target of Zero for Road Traffic Fatalities and Serious Injuries," Technical Report, The Stockholm Environment Institute, 2006.
4. D'Ambrosio, J.G. and Debouk, R., "ASIL Decomposition: The Good, the Bad, and the Ugly," SAE Technical Paper 2013-01-0195, 2013, https://doi.org/10.4271/2013-01-0195.
5. "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," SAE Standard J3016, 2018.
6. Bijlsma, T. et al., "A Distributed Safety Mechanism Using Middleware and Hypervisors for Autonomous Vehicles," in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, 2020.
7. Fürst, S., "Scalable, Safe and Multi-OEM Capable Architecture for Autonomous Driving," in *9th Vector Congress*, Germany, 2018.
8. Fruehling, T. et al., "Architectural Safety Perspectives & Considerations Regarding the AI-Based AV Domain Controller," in *Proceedings of the IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, Graz, Austria, 2019.
9. Bijlsma, T. and Hendriks, T., "A Fail-Operational Truck Platooning Architecture," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, California, USA, 2017.
10. Luo, Y., Saberi, A.K., Bijlsma, T., Lukkien, J.J. et al., "An Architecture Pattern for Safety Critical Automated Driving Applications: Design and Analysis," in *Proceedings IEEE International Systems Conference (SysCon)*, Montreal, Quebec, Canada, 2017.
11. Armoush, A., "Design Patterns for Safety-Critical Embedded Systems," PhD dissertation, RWTH Aachen University, 2010.
12. Koopman, P. and Wagner, M., "Challenges in Autonomous Vehicle Testing and Validation," *SAE Int. J. Trans. Safety* 4, no. 1 (2016): 15-24, https://doi.org/10.4271/2016-01-0128.
13. Aptiv, Audi, Baidu, BMW Group, Continental, Daimler, Fiat Chrysler, HERE, Infineon, Intel, and VW, "Safety First for Automated Driving," White Paper, 2019.
14. Saberi, A.K., Hegge, J., Fruehling, T., and Groote, J.F., "Beyond SOTIF: Black Swans and Formal Method," in *IEEE International Systems Conference (SysCon)*, 2020.
15. Shalev-Shwartz, S., Shammah, S., and Shashua, A., "On a Formal Model of Safe and Scalable Self-Driving Cars," arXiv:1708.06374, 2017.
16. IEEE 2846 Working Group, "A Formal Model for Safety Considerations in Automated Vehicle Decision Making," https://sagroups.ieee.org/2846/, 2020.
17. Selvaraj, Y., Ahrendt, W., and Fabian, M., "Verification of Decision Making Software in an Autonomous Vehicle: An Industrial Case Study," in *International Workshop on Formal Methods for Industrial Critical Systems*, Vol. 11687 (Cham: Springer, 2019).
18. Schmid, T., Schraufstetter, S., Wagner, S., and Hellhake, D., "A Safety Argumentation for Fail-Operational Automotive Systems in Compliance with ISO 26262," in *Proceedings of the 4th International Conference on System Reliability and Safety*, Rome, Italy, 2019.
19. Friedenthal, S., Moore, A., and Steiner, R., *A Practical Guide to SysML: The Systems Modeling Language* (Waltham, MA: Morgan Kaufmann, 2014)
20. Geihs, K., "Middleware Challenges Ahead," *Computer* 34 (2001): 24-31.
21. Kane, A., Chowdhury, O., Datta, A., and Koopman, P., "A Case Study on Runtime Monitoring of an Autonomous Research Vehicle (ARV) System," in *Proceedings of the Runtime Verification*, Lecture Notes in Computer Science, Vol. 9333 (Cham: Springer, 2015).
22. Schlesselman, J.M., Pardo-Castellote, G., and Farabaugh, B., "OMG Data-Distribution Service (DDS): Architectural Update," *IEEE Military Communications Conference* 2 (2004): 961-967.
23. Baidu, "Apollo 3.5 Software Architecture," https://apollo.auto/, accessed April 12, 2021.
24. Luckcuck, M., Farrell, M., Dennis, L.A., Dixon, C. et al., "Formal Specification and Verification of Autonomous Robotic Systems: A Survey," *ACM Computing Surveys (CSUR)* 52, no. 5 (2019): 1-41.
25. Groote, J.F. and Mousavi, M.R., *Modeling and Analysis of Communicating Systems* (Cambridge, MA: The MIT Press, 2014). https://www.mcrl2.org.
26. Groote, J.F. and Mateescu, R., "Verification of Temporal Properties of Processes in a Setting with Data," in

International Conference on Algebraic Methodology and Software Technology, 1999, 74-90.

27. Groote, J.F., Kouters, T.W.D.M., and Osaiweran, A., "Specification Guidelines to Avoid the State Space Explosion Problem," *Softw. Test. Verif. Reliab.* 25 (2015): 4-33.
28. "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," SAE Recommended Practice J3016, 2016, http://standards.sae.org/j3016_201609/.
29. Thorn, E., Kimmel, S.C., Chaka, M., and Hamilton, B.A., "A Framework for Automated Driving System Testable Cases and Scenarios," Technical Report No. DOT HS 812 623, U.S. Department of Transportation, National Highway Traffic Safety Administration, 2018.
30. Emzivat, Y., Ibanez-Guzman, J., Martinet, P., and Roux, O.H., "Dynamic Driving Task Fallback for an Automated Driving System Whose Ability to Monitor the Driving Environment Has Been Compromised," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, California, USA, 2017.
31. Yeh, Y.C., "Triple-Triple Redundant 777 Primary Flight Computer," in *Proceedings of the IEEE Aerospace Applications Conference*, 1996, Vol. 1, 293-307.
32. Oliveira, R., Pereira, D., Maia, C., and Santos, P., "A Domain Specific Language for Automotive Systems Integration," in *Proceedings of the IECON 2019—45th Annual Conference of the IEEE Industrial Electronics Society*, Lisbon, Portugal, 2019, 4483-4488.
33. AUTOSAR, "Specification of ECU State Manager," AUTOSAR CP R20-11, 2020, https://www.autosar.org/fileadmin/user_upload/standards/classic/20-11/AUTOSAR_SWS_ECUStateManager.pdf.