THE
AUTONOMOUS
Working Group Safety & Architecture

Safe Automated Driving: Requirements and Architectures

Full Report

AUTHORS

BASELABS

Gabi Escuela
gabi.escuela@baselabs.de

CoreAVI
INNOVATING A SAFER TOMORROW

Neil Stroud
neil.stroud@coreavi.com

DENSO
DENSO AUTOMOTIVE Deutschland GmbH

Kazuhito Takenaka
k.takenaka@eu.denso.com

Justin-Kiyoshi Tiele
j.tiele@eu.denso.com

infineon

Udo Dannebaum
udo.dannebaum@infineon.com

Jens Rosenbusch
jens.rosenbusch@infineon.com

KTH
VETENSKAP OCH KONST

Martin Törngren
martint@kth.se

TTTechAuto

Georg Niedrist
georg.niedrist@tttech-auto.com

Moritz Antlanger
moritz.antlanger@tttech-auto.com

Christian Mangold
christian.mangold@tttech-auto.com

Friedrich Reisenberger
friedrich.reisenberger@tttech-auto.com

Nahla Ben Mosbeh
Chaitanya Shinde
Ayhan Mehmed

Christoph Schulze
Shailesh More

Lucas Fryzek
Matthew Storr

REVIEWERS

The Working Group Safety & Architecture would like to thank the reviewers of the report for their valuable input and feedback.

Ansys

Bernhard Kaiser

Carnegie Mellon University

Philip Koopman

Fraunhofer
IESE

Jan Reich
Rasmus Adler
Daniel Schneider

NXP

Andrei Terechko

VECTOR

Jan Toennemann

CONTENTS

VERSION HISTORY

Version	Date	Revision description
1.0	01.12.2023	Initial release

EXECUTIVE SUMMARY

The Autonomous is an initiative and open platform bringing together leading executives and experts of the mobility ecosystem to align on subjects relevant to the safety of autonomous driving (AD); in its Safety & Architecture Working Group, members of international research institutes and industrial companies came together to investigate what the system-level conceptual architecture of an automated vehicle could look like, in order to address the functional and safety challenges of automated driving.

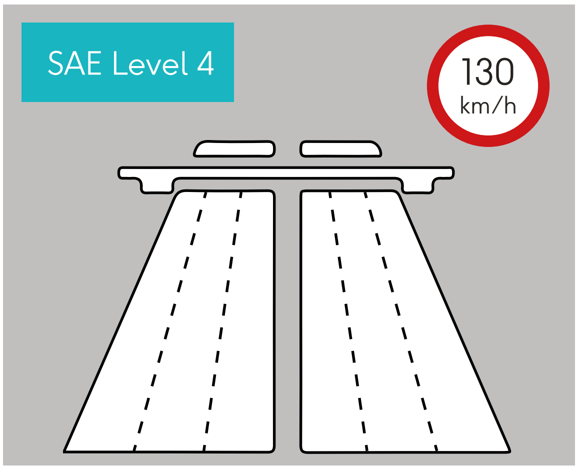
This report was compiled from June 2021 to December 2023 in three major report increments that were accompanied by external reviewers from industry and academic experts.
Our work is structured as follows: we start by outlining the reference use case of an SAE L4 Highway Pilot, derive key system requirements, and establish general constraints and established design principles for implementing such a use case in an AD system. We continue with candidate architectures from market and literature research and derive their properties. Finally, we compare the architectures with respect to a set of criteria that we consider crucial (such as system availability, robustness, and scalability) and conclude with development considerations and implementation hints.

The intended readers are system owners who make architectural decisions and ensure consistency on many different abstraction levels, from high-level conceptual architectures to low-level physical implementations. Our intention is to support them in making such decisions and building up a safety argumentation.

ABSTRACTION LEVEL AND REFERENCE USE CASE

It is commonly understood and accepted that the development of a safe automated driving system for complex driving tasks is a big challenge. Even when developed to the highest standards, complex HW and SW elements will exhibit faults that can materialize in an arbitrary way. Still, the overall autonomous driving system needs to tolerate such faults and keep up operation at least for a minimum time frame – i.e., it needs to be fail-degraded. A well-chosen architecture is indispensable to manage the complexity of autonomous driving systems and to ensure fault tolerance in an effective and efficient way.

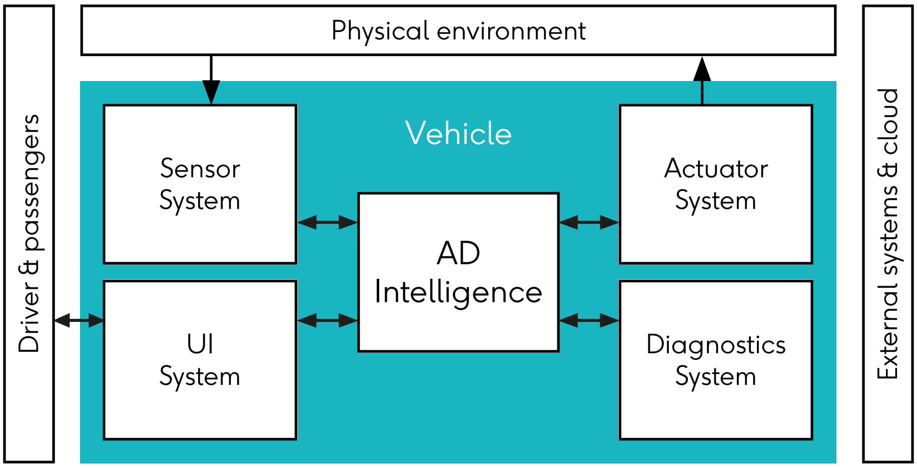
For our analysis, we have chosen what we call the conceptual system architecture level: we consider the system as a set of well-encapsulated subsystems that can comprise up to an entire processing channel, or at least a dedicated subset of the overall functionality.



As our reference use case, we chose an SAE Level 4 Highway Pilot to act as a backdrop for compiling assumptions and deriving system requirements and design principles applicable to conceptual system architectures. Such a feature, which is expected within the next few years, will need to deal with complex traffic situations and will necessitate non-trivial system architectures due to high availability requirements and complexity.

SYSTEM REQUIREMENTS, DESIGN CONSTRAINTS AND DESIGN PRINCIPLES

The Safety & Architecture Working Group focuses on a system providing AD functionality, which we call the **Automated Driving Intelligence (ADI)**. This system covers all cognitive tasks previously performed by the driver. A simplified representation is shown below, illustrating the four other systems the ADI is connected to, as well as the elements that „close the loop" with the physical environment.



A set of key system requirements (summarized in the table below) should be applied to the ADI to ensure the safety of commands to the actuators. Besides the expected timeliness, correctness, and consistency of commands, their availability is highly safety-relevant for an SAE L4 function. Additionally, an ADI architecture shall foresee self-diagnostic mechanisms and shall support detecting and handling functional insufficiencies (including, but not limited to the perception functions).

S1	ADI output timeliness
S2	ADI output availability
S3	ADI output correctness
S4	ADI output consistency
S5	Perception malfunction detection
S6	ADI diagnostics

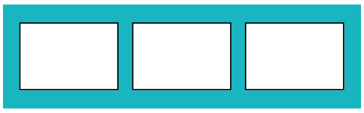
When coming up with conceptual system architectures intended to satisfy these system requirements, technological limitations constrain how high-reliability systems can be designed, built, and tested using realistic HW and SW components. Such general constraints need to be addressed by an architecture for automated driving, e.g.: it is impossible to avoid design faults and single-event upsets in large and complex monolithic systems, and it is impossible to achieve high availability by testing or to specify all edges cases that an AD function must cope with.

In addition, well-established practices should be respected in a sound conceptual system architecture. We identify a number of such design principles, e.g., using well-encapsulated independent subsystems (“Fault Containment Units”, FCUs), applying diversity and redundancy, preventing emergent behavior by limiting interactions between subsystems, and mitigating hazards by adopting the Swiss cheese model.

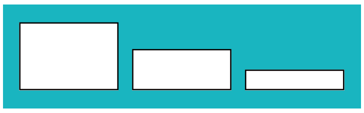
CANDIDATE ARCHITECTURES

From industry publications, academic papers, and patent publications we have identified candidate architectures and grouped them into three basic categories of conceptual system architectures:

1. MONOLITHIC ARCHITECTURES
present the status quo for SAE L2 systems and are a natural basis for incremental development to L3 systems.



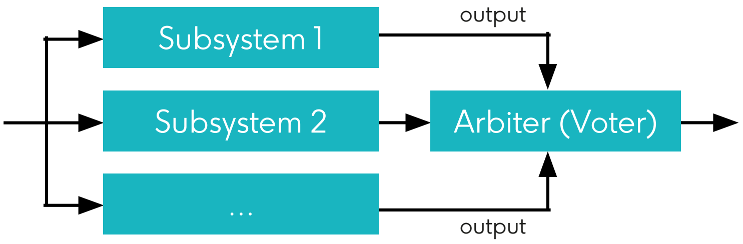
2. SYMMETRIC ARCHITECTURES
rely on multiple channels providing the same or similar functions, often with some voting mechanism for arbitration.



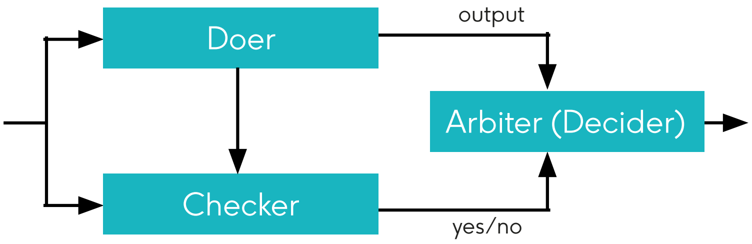
3. ASYMMETRIC ARCHITECTURES
employ asymmetric decompositions to reduce the complexity of some subsystems, e.g., via Doer/Checker or Active/Hot Stand-By patterns.

These architectures employ several underlying patterns:

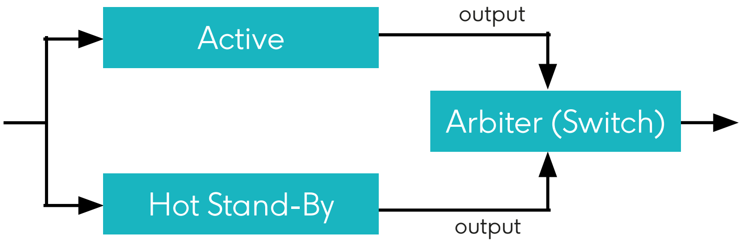
The Arbitration pattern manages redundancy by deciding (e.g., voting) between equal channels. The Agreement pattern is similar, but without an external arbiter.



The Doer / Checker pattern asymmetrically decomposes (for correctness) a channel into a Doer performing the intended function and a Checker approving it.



The Active / Hot Stand-By pattern asymmetrically decomposes (for availability) into a preferred Main channel and – if that is not available – a Fallback channel.



A representative of the monolithic architectures is the **Single-Channel** architecture, where a single ECU performs all tasks of the Automated Driving function, i.e., processes the sensor data into a consistent environment model, generates trajectories and, finally, set points for the actuators. Examples of this architecture are the AUDI zFAS System for an SAE L3 Traffic Jam Pilot (2017) or, more recently, Tesla's "Full Self Driving" (FSD), as far as can be judged from available documentation, or monolithic end-to-end AI systems. At least for the zFAS, availability requirements are relaxed compared to an L4 Highway Pilot, and the system does not need to provide complex fallback functionality in case of a fault – hence it can be backed up by a different ECU outside the AD system.

The **Majority Voting** architecture as a representative of the symmetric architectures implements a number of channels (three or more), each of which can perform the full nominal function. The voter compares (exactly or inexactly) the channels' results and forwards the majority opinion to the actuators. If all three results differ, no decision can be made. To achieve fault tolerance, multiple instances of the voter may be necessary.

The first of the considered asymmetric architectures is the **Channel-Wise Doer/Checker/Fallback** architecture, where a Doer performs the nominal driving function and can resemble an SAE L2 system, while a Fallback performs only Minimal Risk Maneuvers. A Checker validates both the Doer's and the Fallback's output. A Selector receives the Checker's verdict and forwards either the trajectory from the Doer or from the Fallback to the actuators.
Doer/Checker/Fallback are complex subsystems, and each of them forms an FCU that can fail arbitrarily and independently. They are implemented in a diverse way to minimize common-cause failures, to ensure sufficient independence. The Selector is simple, has low performance requirements, and can be fully verified to preclude systematic faults. To achieve fault tolerance, it consists of two identical instances.

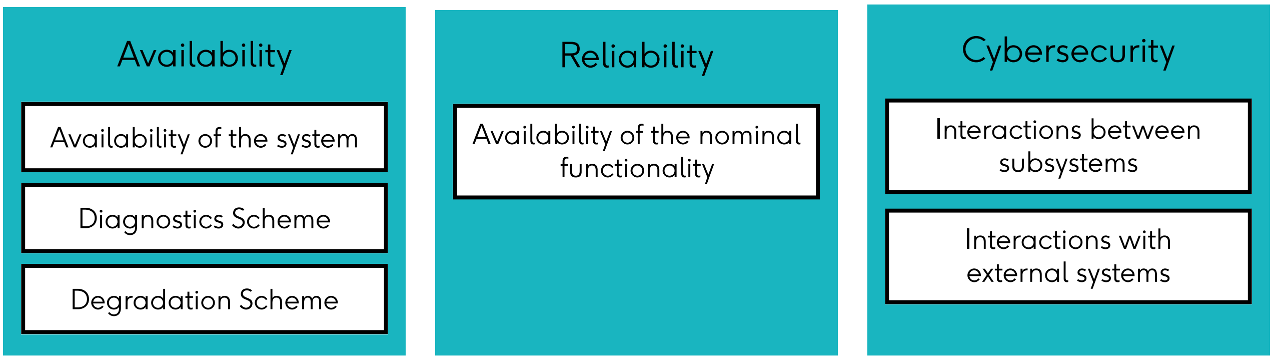
Another asymmetric architecture is the **Layer-Wise Doer/Checker/Fallback**, essentially a multi-channel approach with at least one primary and one safing channel, which provides a degraded mode of operation in case the primary channel fails. Each channel consists of Doer/Checker pairs, arranged in multiple layers of the Sense-Plan-Act model. A Priority Selector determines the output to be sent to the actuators, depending on the states of the channels.
The Priority Selector is a high safety integrity component, simpler than the Checkers. It must continue to operate in the presence of failures to deliver either the primary or the safing output, or to trigger an emergency stop. It may fail silently so long as that failure triggers an emergency (blind) stop.

As a final asymmetric example, we study the Distributed Safety Mechanism architecture, which can be seen as a more complex, distributed variant of the Doer/Checker/Fallback approach. The architecture is composed of three channels, each of them containing safety monitors – a Nominal Channel, consisting of the function itself and controlled by a Function Monitor, an Emergency Channel, which is controlled by a Controller Safety Mechanism, and a Safety Channel, which is controlled by a Vehicle Safety Mechanism. The Function Monitor is checking for SOTIF issues, the Controller Safety Mechanism is responsible for monitoring all the function controllers (including hardware and software platforms) and the Vehicle Safety Mechanism.

The Vehicle Safety Mechanism is responsible for monitoring the communication networks and the Controller Safety Mechanism. It can send control commands to the vehicle actuators in case of comfort or safe stop, by using independent sensor data.

ARCHITECTURE EVALUATION METHODOLOGY, CRITERIA, AND FINDINGS

We evaluate the presented architectures with respect to several key criteria:



Availability

Availability of the system

Diagnostics Scheme

Degradation Scheme

Reliability

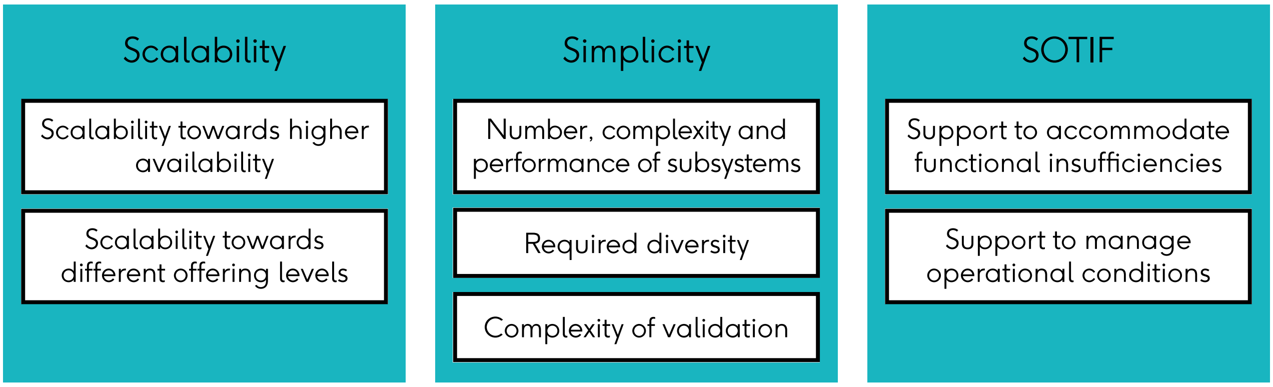
Availability of the nominal functionality

Cybersecurity

Interactions between subsystems

Interactions with external systems

- Availability: to what extent would the architecture support the fail-operational property, i.e., enable safe operation even in the case of unavoidable electronic or software faults?
- Reliability: would continuity of the nominal functionality be well supported, to help ensure a positive user experience, e.g., by avoiding function degradation?
- Cybersecurity: would the architecture be susceptible to security threats, or would it support resilience measures against attacks?



Scalability

Scalability towards higher availability

Scalability towards different offering levels

Simplicity

Number, complexity and performance of subsystems

Required diversity

Complexity of validation

SOTIF

Support to accommodate functional insufficiencies

Support to manage operational conditions

- Scalability: to what extent would cost-efficient downscaling to lower SAE levels (for vehicle options), or upscaling to higher SAE levels (for future enhancements), be supported?
- Simplicity: would the architecture be conceptually simple, to support modular development, verification, and validation?
- Safety of the Intended Functionality: would the architecture help ensure robustness and safe operation in the presence of functional insufficiencies and unavoidable edge cases?

For the evaluation itself, we proceed in three steps: To form an unbiased basis for the evaluation, we start with a generic evaluation of each architecture, by listing observations (properties of each architecture) related to each criterion. Next, we evaluate the relative significance of the above criteria for the selected use case of an SAE L4 Highway Pilot. Finally, we directly compare the architectures, considering the observed properties from the generic evaluation and inferring merits or weaknesses with respect to each evaluation criterion, and qualitatively ranking them under that criterion.

As a result, it turns out that the **asymmetric architectures are generally preferable to symmetric ones**. By virtue of their inherent diversity of computational streams, they exhibit more robustness with respect to availability, cybersecurity, and SOTIF because the channels complement each other and tend to mutually compensate their potential weaknesses. The asymmetric architectures also offer more options with respect to scalability, as omitting channels quite naturally leads to lower SAE level functionality, and higher levels can be reached by adding channels. Superficially, they might appear more complex and less reliable (in the sense of keeping the intended functionality) than symmetric architectures. However, their diversity actually facilitates modular development and independent verification of the channels, which in turn is expected to lead to lower development costs and enhanced availability.

The symmetric architectures, such as voting approaches, are seen as highly susceptible to common cause deficiencies that might impact all channels at the same time – be it from the functional safety, SOTIF, or even the cybersecurity perspective. If this problem is addressed by heterogeneous channel implementations (e.g., different chipsets), then the feasibility of voting is questionable since channels might come to different but equally valid solutions. Finally, the monolithic single-channel architecture is not seen as a feasible solution: it does not fulfill any of the criteria without additional internal redundancy and supervision mechanisms that are introduced during implementation. This would make it evolve into one of the other architectures.

IMPLEMENTATION CONSIDERATIONS

For further refinement of the conceptual system architecture into combined hardware/software solutions with redundant channels, we need to consider dependent failures of the elements. In other words, sufficient independence of the channels (which includes freedom from interference), and the absence of single points of failure need to be ensured. We discuss dependent failure initiators and provide hints on how to overcome them.

Similarly, we consider selected topics related to the further refinement of the conceptual system architecture into a software safety concept. This includes a discussion on different software architectural styles – depending on the use case – as well as common safety measures.

To achieve a sound safety argumentation for the chosen architectures, we refer to the relevant safety standards, in particular ISO 26262 and ISO 21448. In addition, we propose advanced methods like formal verification on the architecture level and for the logical-to-physical mapping, as well as Markov modeling to quantify the overall system availability, to meet an ASIL D target.

INTRODUCTION AND PURPOSE

THE AUTONOMOUS

The Autonomous is the global community shaping the future of safe autonomous mobility. Initiated by TTTech Auto in 2019, The Autonomous is an open platform building an ecosystem of all actors involved in the development of safe autonomous mobility. Ecosystem partners range from car manufacturers, technology suppliers and regulatory authorities to disruptors, thought leaders, academia, and government institutions. The goal of The Autonomous is to generate new knowledge and technological solutions in the field of autonomous mobility, thus accelerating the transition to market readiness and series development of safe self-driving vehicles. To achieve this, The Autonomous has put in place two strategic streams:

1. Event Stream – facilitates discussions and networking for leading executives and experts from the autonomous mobility ecosystem.
2. Innovation Stream – facilitates cooperation across the industry to work on global reference solutions for safety challenges. These reference solutions conform to relevant standards and will facilitate the adoption of safe autonomous mobility on a global scale. As part of the Innovation Stream, The Autonomous launches and facilitates Working Groups and Expert Circles in order to develop pre-competitive concepts, concrete technical solutions, best practices, and recommendations in key areas of autonomous driving – from E/E architectures and artificial intelligence to regulatory frameworks and societal acceptance.

The findings of The Autonomous Working Groups are presented yearly at The Autonomous Main Event.

WORKING GROUP SAFETY & ARCHITECTURE

The first initiated Working Group of the Innovation Stream of The Autonomous is the one on "Safety and Architecture": International research institutes and industry leaders come together to address the fundamental question of what the conceptual system architecture of an automated vehicle (SAE level 4 and higher) should look like, i.e., how the system's partitioning into computational streams, for instance for safety and redundancy purposes, could be performed (for further explanations, see section 1.4). The present report produced by the Working Group "Safety & Architecture" addresses this topic.

It is commonly understood and accepted that the development and implementation of a failure-free automated driving system for complex driving tasks is an extremely tough challenge. Even having been developed to the highest standards, complex HW and SW elements can exhibit malfunctions that can materialize in an arbitrary way. Still, the overall autonomous driving system needs to tolerate these and keep up operation at least for a defined time frame – i.e., needs to be fail-operational or at least fail-degraded. Regarding faults, this study generally concerns how to achieve a dependable computational system architecture and is thus not limited to faults like the ones caused by a lack of functional safety or to malfunctions due to a lack of "safety of the intended functionality". A good summary of dependability aspects that need be considered can be found in [1].

The chosen level of conceptual representation is on the one hand sufficiently specific to be useful as a reference and on the other hand sufficiently generic to allow for different implementations. More details on this conceptual representation will be given in section 1.4 - Abstraction level. This report focuses on the computational unit between the sensors and actuators which will be called "Automated Driving Intelligence" [55][1] (ADI), see Figure 1. This includes sensory processing, fusion, trajectory finding and decision making, but excludes raw data sensors and the actuators. Detailed hardware and software architectures are topics for potential follow-up activities of the Working Group after this report.

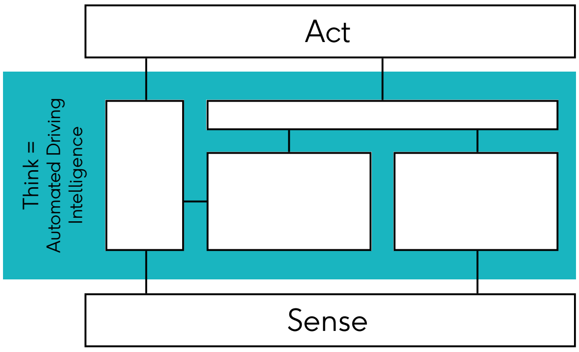


Figure 1: Automated Driving Intelligence (ADI)

PURPOSE AND STRUCTURE OF THIS DOCUMENT

Architecture and design occur on multiple different abstraction levels (see Figure 2). It lies within the responsibility of "system owners", whom we consider the intended readers of this document, to ensure a consistent design across all such levels. System owners (see also section 2.1.1 - System owner persona) may work for OEMs, mobility companies, or their suppliers and need to make architectural decisions both on a high, abstract level and on a lower, implementation level (see also section 2.1.2 - Architecture design process and decisions).

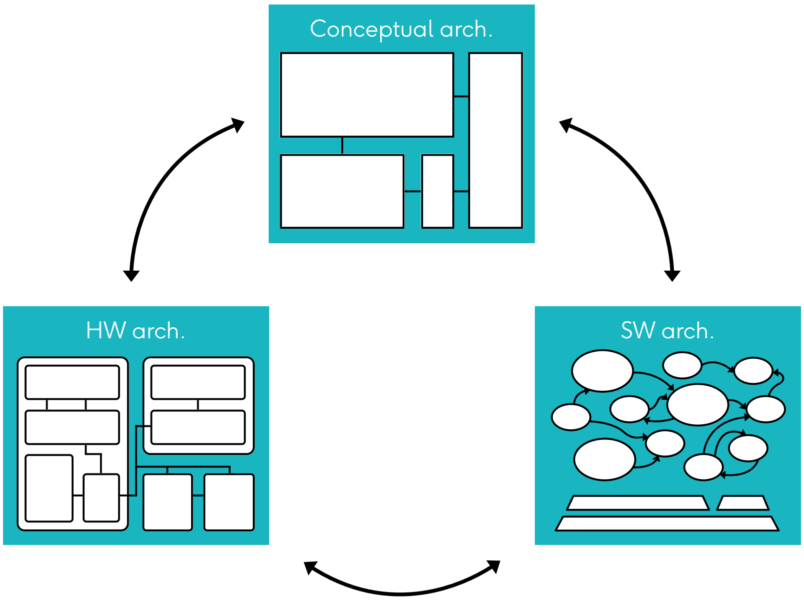


Figure 2: Ensuring consistency between architectures on different abstraction levels.

[1] We use this term [55] instead of the more generic "AD System" to indicate that it excludes other systems such as raw data sensors or actuators.

This document is intended to support system owners in making high-level architectural decisions and mapping these to low-level implementations. It aims to provide a structured analysis of high-level architectures in the Automated Driving (AD) context, as well as supportive arguments for demonstrating that crucial requirements are satisfied and important KPIs are well met.

This document is structured as follows (see also Figure 3):

- In section 1 - Background and premises, we define the context in which we look at high-level system architectures.
    - We start by outlining a reference AD feature that captures the challenges regarding safety and availability. For this, we chose an assumed version of an SAE L4 Highway Pilot feature (see section 1.1).
    - The reference AD feature is assumed to be provided by an AD system. The system boundary of this “AD Intelligence” is described in section 1.2.
    - Based on these, we then derive high-level system requirements for the “AD Intelligence”, with a focus on safety and availability (see section 1.3).
    - The architectural abstraction level that we consider is described in detail in section 1.4.
    - Finally, we also collect general constraints and design principles relevant to system architectures within our chosen context and on our chosen abstraction level (see section 1.5).
- In section 2 - Architecture evaluation criteria, we define evaluation criteria relevant to high-level system architectures.
    - In order to choose evaluation criteria relevant to our intended readers, we start by describing the architectural choices they may need to make (see section 2.1).
    - Many attributes of a well-made AD system do not directly depend on the high-level architecture. We thus summarize these attributes and assume that they are covered (see section 2.2).
    - Attributes that are more closely linked to the choice of high-level system architecture are collected in sections 2.3 to 2.8 and summarized in tabular form in section 2.9. Each of these attributes is broken down into multiple evaluation criteria (and associated key questions) that we later apply in the architecture evaluation.
- In section 3 - Candidate architectures, we collect and describe different high-level system architectures.
    - We start by describing the process we used to collect candidate high-level system architectures (see section 3.1).
    - Since some of these share certain basic principles, we chose to extract these and describe their intention and mechanism in a generic way (see section 3.2).
    - The six candidate conceptual system architectures are clustered in three major groups and described in a comparable way in sections 3.3, 3.4, and 3.5.
- In section 4 - Architecture evaluation, we evaluate the collected conceptual system architecture candidates.
- Our evaluation methodology is described in section 4.1.
- We use the evaluation criteria defined earlier to make a series of general observations on each candidate architecture (see section 4.2). These are not specific to an AD use case.
- This is then followed up by considering these observations in the context of our reference AD use case (see section 4.3).
- In section 5 - Implementation considerations, we provide considerations for mapping conceptual system architectures to specific HW and SW architectures.
- Considerations for mapping a conceptual system architecture to a physical HW architec-

ture are collected in section 5.1.

- Considerations for mapping a conceptual system architecture to a physical SW architecture are collected in section 5.2.
- Finally, the process for constructing a safety argumentation is outlined in section 5.3.

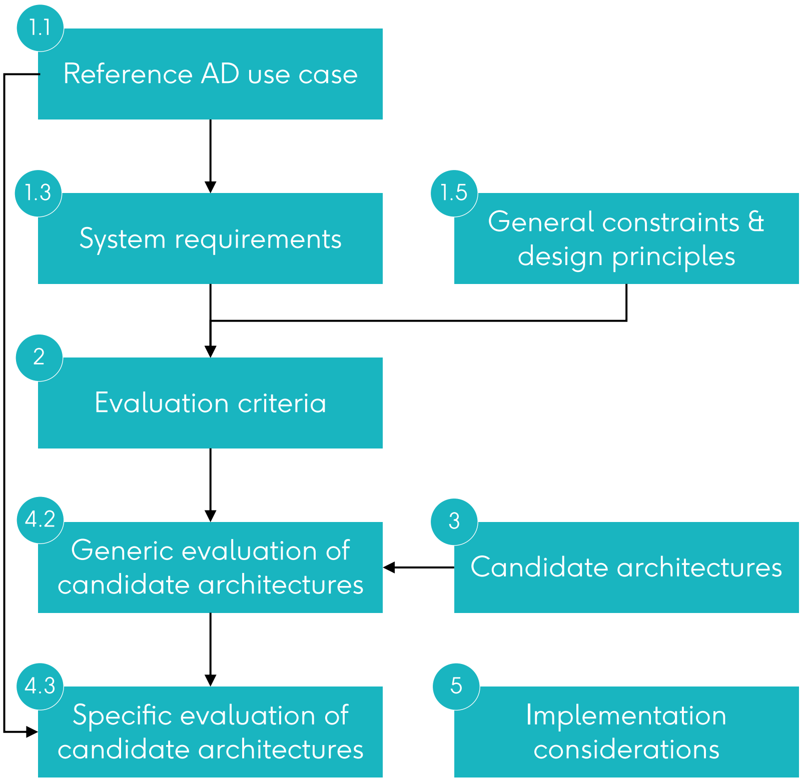

1.1
Reference AD use case
1.3
System requirements
1.5
General constraints & design principles
2
Evaluation criteria
4.2
Generic evaluation of candidate architectures
3
Candidate architectures
4.3
Specific evaluation of candidate architectures
5
Implementation considerations

Figure 3: Structure of this document.

1 BACKGROUND AND PREMISES

The requirements, general constraints, and design principles as introduced here relate to a system of interest, referred to as the ADI (recall Figure 1), and more specifically, to the computational system architecture of the ADI. Ensuring the safety of automated driving requires a system safety perspective that takes the AD intelligence, the vehicle platform, the behavior of surrounding actors as well as the traffic environment into account, including the full set of responsibilities previously assumed by the driver. The work described in this report does not take on this entire grand challenge, but rather focuses on architectural aspects of the AD Intelligence and their contributions to safety.

With the overall goal to propose and evaluate architectures for AD systems, a main emphasis is placed on meeting appropriate functional safety requirements, with considerations of system and software complexity, and hardware reliability (referring to ISO 26262 [2]). With the introduction of advanced external perception and machine learning, additional safety hazards must be addressed for automated driving, as traditionally, functional safety standards have assumed that “requirements are known” and “nominal operation” with no software or electronics failures, is safe. This has led to new standards, such as ISO 21448 “Safety of Intended Functionality” (SOTIF) [3], which attempts to address these challenges. SOTIF is part of the considerations for this work regarding causes of failures and qualitative aspects of diversity (further elaborated in section 1.5.1 - General).

For highly Automated Vehicles (AVs), the increasing complexity and risks of failures lead to open issues, including what constitutes safe road behavior and what measures are needed to assure a “positive risk balance” [4] such that an AV would at least perform better than an average driver. These are topics being treated in ongoing standardization work, such as ISO/AWI TS 5083—safety of automated driving systems. A positive risk balance is considered for the architectural work in terms of, for example, reliability goals. Behavioral aspects such as tactical safety [5] are beyond the scope of this work.

Cybersecurity will also be key for automated vehicles and their relation to safety, as manifested by the new standards like ISO/SAE 21434:2021 [6] for automotive. Cybersecurity aspects are, however, not covered extensively in this release, partly due to the chosen abstraction level. Some aspects of cybersecurity that have an indirect impact on security considerations will be covered through a few evaluation criteria, see further section 2.5 - Cybersecurity.

1.1 REFERENCE AD USE CASE

1.1.1 MOTIVATION

This section outlines the reference AD use case targeted by the "Safety & Architecture" Working Group of The Autonomous. This may later be supplemented by additional AD use cases in the second iteration of the Working Group (see Outlook).

This reference AD use case **shall serve** the following purposes:

- As an input for defining reasonable assumed requirements (see section 1.3 - System requirements). In ISO 26262, safety-related requirements are ultimately derived from item-specific safety goals, e.g., that the system shall avoid collisions and loss of vehicle control.
- As an input for establishing what level of algorithmic complexity is required to perceive varied environments and handle different and dynamic traffic scenarios.

- As an input for defining general assumptions (see section 1.5 - General constraints and design principles).
- As an input for defining and quantifying dependability goals.

This reference AD use case **may help** with the following purposes:

- To derive a rough estimate of what computational resources are required.
- To derive a rough estimate of currently achievable failure rates for the computational resources as well as the estimated rate of hazardous behavior of the intended functionality[2] from software (application and infrastructure code).
- To refine requirements that are related to vehicle-level use cases and scenarios into more detailed requirements on the algorithm level, e.g., perception, activation / deactivation, degradation, or warnings[3].

Note: This reference AD use case is intended to give the reader a general understanding of what such a feature could look like. While some of these descriptions have direct relevance for architectural considerations later on (marked in bold font in the following reference AD use case sub-sections), many others merely serve as a background to outline the many different aspects and perspectives involved.

1.1.2 CHOICE OF REFERENCE AD USE CASE

At the moment, various OEMs are discussing a number of different AD use cases, each having different architectural implications. We have screened these on a high level according to several criteria to identify a suitable reference AD use case:

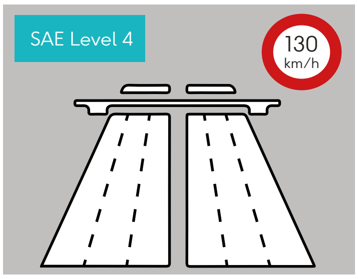
- Timeline
  The reference AD use case should (likely) become technically and commercially feasible within the next 5-10 years.
- Complexity of the Operational Design Domain (ODD)
  The reference AD use case should apply to an ODD of at least medium complexity. This implies that complex algorithms and high-performance hardware are necessary.
- System availability
  The reference AD use case should have high integrity and availability requirements, i.e., require resilience against faults (fail-operational/fail-degraded). This implies that a non-trivial conceptual system architecture is necessary to compensate for the weaknesses of complex algorithms and powerful hardware.

[2] Malfunctioning behavior can arise due to faults (e.g., bugs), due to functional insufficiencies (e.g., environmental aspects neglected in the specifications), due to operational disturbances (e.g., environmental conditions), or due to misuse. In systems involving machine learning, this may also be caused by bad or biased training data. There are some empirical estimates for the number of undiscovered bugs per line of code remaining despite using proper development processes [10].

[3] This may also include performance-related aspects such as timing, accuracy, and detection reliability.

TABLE 1: LIST OF AD USE CASES UNDER DISCUSSION (NOT COMPREHENSIVE).

AD use case	Timeline	ODD complexity	System availability
Traffic Jam Pilot	●●●●●	●●○○○	●●●○○
Highway Pilot	●●●●○	●●●○○	●●●●●
Mobility as a Service (MaaS)	●●●○○	●●●●●	●●●○○
Valet Parking	●●●●○	●●○○○	●●○○○
Low-speed AD (shuttle)	●●●●○	●●○○○	●●○○○



The AD use case we consider the most suitable (see Table 1) is an **SAE Level 4 Highway Pilot (HWP)** feature. This function is expected to be introduced between 2025 and 2028.

We explicitly target the "High Automation" level/"SAE L4" (according to the classification scheme proposed by the Society of Automotive Engineers [7][4]) over SAE L3 (see also [8, 9]). This entails the following:

- The system assumes full responsibility for the Dynamic Driving Task (DDT) in all dimensions, i.e., the driver can be „hands-off".
- The system assumes full responsibility for its own supervision, i.e., the driver can be „eyes-off" and „brain-off".
- The system will never require the driver to take back control, which would be both difficult to achieve [9] and would also have a pronounced detrimental effect on the „quality time" gained from an AD feature.
- The system may only request that the driver take back control within more than a few dozen seconds to allow a smooth transition to user-operated mode. If the driver does not take back control when asked to, the system needs to enter a safe state on its own.

If the AD system encounters a fault and/or if the driver does not respond to a request to intervene (exact time frame subject to concrete system specifications), the vehicle will perform a DDT fallback operation. We assume that this consists of the execution of a Minimal Risk Maneuver (MRM) [7] to enter a Minimal Risk Condition (MRC), e.g., pulling over to the right side/emergency lane and coming to a controlled stop or (if this is no longer feasible) coming to a controlled stop in the current lane, but excludes recovery, i.e., the AD system does not attempt to continue driving without a full reset after entering the MRC.

1.1.3 FUNCTIONALITY PROVIDED TO USER

In the following, we define an assumed version of a HWP feature, similar to proposals from different OEMs. These allow the driver of a passenger car (sedan, SUV, crossover, or similar vehicle with relatively simple vehicle dynamics) to take their eyes off the road and perform other tasks while on a highway, with the AD system performing the entire DDT (lateral and longitudinal vehicle motion control and complete Object and Event Detection and Reaction (OEDR)) and assuming full responsibility.

[4] These classification schemes are still evolving, which is why we consider a more detailed outline of the AD use case (including feature activation and deactivation) necessary.

ID	Statement
U1	**The HWP feature supports lane keeping.**
U2	**The HWP feature supports lane changes.**
U3	The HWP feature supports traffic jams (stop & go traffic).
U4	The HWP feature can be set to continue driving on the current highway.
U5	The HWP feature can be set to go to a target highway exit.
U6	**The HWP feature supports speeds of up to 130 km/h.**
U7	The HWP feature visually presents its status (e.g., off / on / malfunctioning) as well as its world model and motion plan to the passengers.

The Operational Design Domain (ODD) of the HWP feature is outlined in more detail in Appendix A: ODD outline of reference AD use case.

1.1.4 FEATURE ACTIVATION, DEACTIVATION, AND REQUESTS TO INTERVENE

ID	Statement
U8	We assume that “regular activation” of the HWP feature could proceed as follows: • The driver presses the "activate HWP" button. • The system checks that all conditions for its activation are fulfilled (see Appendix A: ODD outline of reference AD use case) and indicates the result to the driver. • The system gradually offers more resistance to steering wheel and pedals.
U9	We assume that “regular system-initiated deactivation” of the HWP feature could proceed as follows: • The system visually represents the automated driving system’s world model, motion plan and diagnostics to the user to simplify the (requested) control takeover for the user. • The system indicates that it is approaching a point where the conditions for activation will no longer be fulfilled (end of the mission, change of external circumstances, detected failure, etc.). • The driver presses the "acknowledge deactivation" button. • The system checks that the driver is capable of driving (attentive and hands on the steering wheel) and indicates the result to the driver. • The system gradually offers less resistance to steering wheel and pedals. • If the driver fails to resume control, the system executes an MRM when the conditions for activation are no longer fulfilled.
U10	We assume that “regular driver-initiated deactivation” of the HWP feature could proceed similarly to “regular system-initiated deactivation”, but without the first two steps.
U11	We assume that “fast driver-initiated deactivation” of the HWP feature could proceed as follows: • The driver puts their hands on the steering wheel and/or feet on the pedals. • The driver overrides the resistance offered by the system. • The system indicates to the driver that it has relinquished control.

ID	Statement
U12	We assume that "driver-initiated emergency deactivation" of the HWP could proceed as follows: • The driver presses the "pull over" button. • The system indicates to the driver that it will come to a controlled stop. • The system executes an MRM.

1.1.5 DEGRADED FUNCTIONALITY

ID	Statement
U13	**The HWP feature has a nominal mode (routine/normal operation), during which it is capable of executing the mission.**
U14	**The HWP feature has a degraded mode (see also Figure 4), during which it will execute an MRM (pulling over, controlled stop, or emergency stop) [7].**
U15	**The HWP feature will enter degraded mode if any part of the AD system encounters errors seen as critical to the ADI functionality or if the ODD is violated.**
U16	After entering degraded mode (unable to continue mission), the HWP feature will not activate again without a full reboot.
U17	**In degraded mode, the HWP feature will try to come to a controlled stop in (what is understood as) a safe enough location (i.e., emergency lane or right-most lane). [First choice]**
U18	If this is not possible, the HWP feature will try to come to a controlled stop in the current lane of travel. [Second choice]
U19	If this is not possible, the HWP feature will try to come to an emergency stop. [Third choice]
U20	The HWP feature **does not** have a limp-home mode, during which it is capable of continuing the mission with reduced functionality (e.g., reduced speed) and/or try to restore full functionality (e.g., partial reboot while continuing to drive).

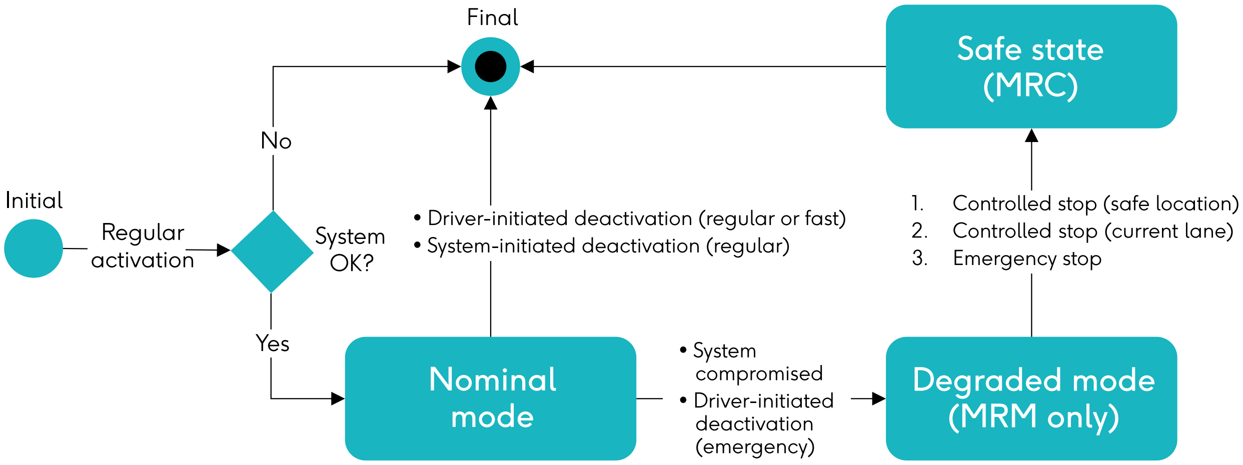

Final
Safe state (MRC)
No
Initial
Regular activation
System OK?
• Driver-initiated deactivation (regular or fast)
• System-initiated deactivation (regular)
1. Controlled stop (safe location)
2. Controlled stop (current lane)
3. Emergency stop
Yes
Nominal mode
• System compromised
• Driver-initiated deactivation (emergency)
Degraded mode (MRM only)

Figure 4: State diagram of different operating modes.

1.2 SYSTEM BOUNDARY

1.2.1 OVERVIEW

The Working Group „Safety & Architecture“ primarily considers a system providing AD functionality, i.e., the Automated Driving Intelligence introduced previously (recall Figure 1). In this section, we lay out the boundary of this AD Intelligence and its interactions with other systems outside this system boundary. Due to our focus on system conceptual architectures (as opposed to detailed SW or HW architectures), we only describe the data and control flow on interfaces and omit HW-related aspects such as concrete network topologies, power supply or cooling. Figure 4 shows such a layout, providing a simplified representation including the elements that “close the loops”, i.e., the physical vehicle and the human making use of the UI system.

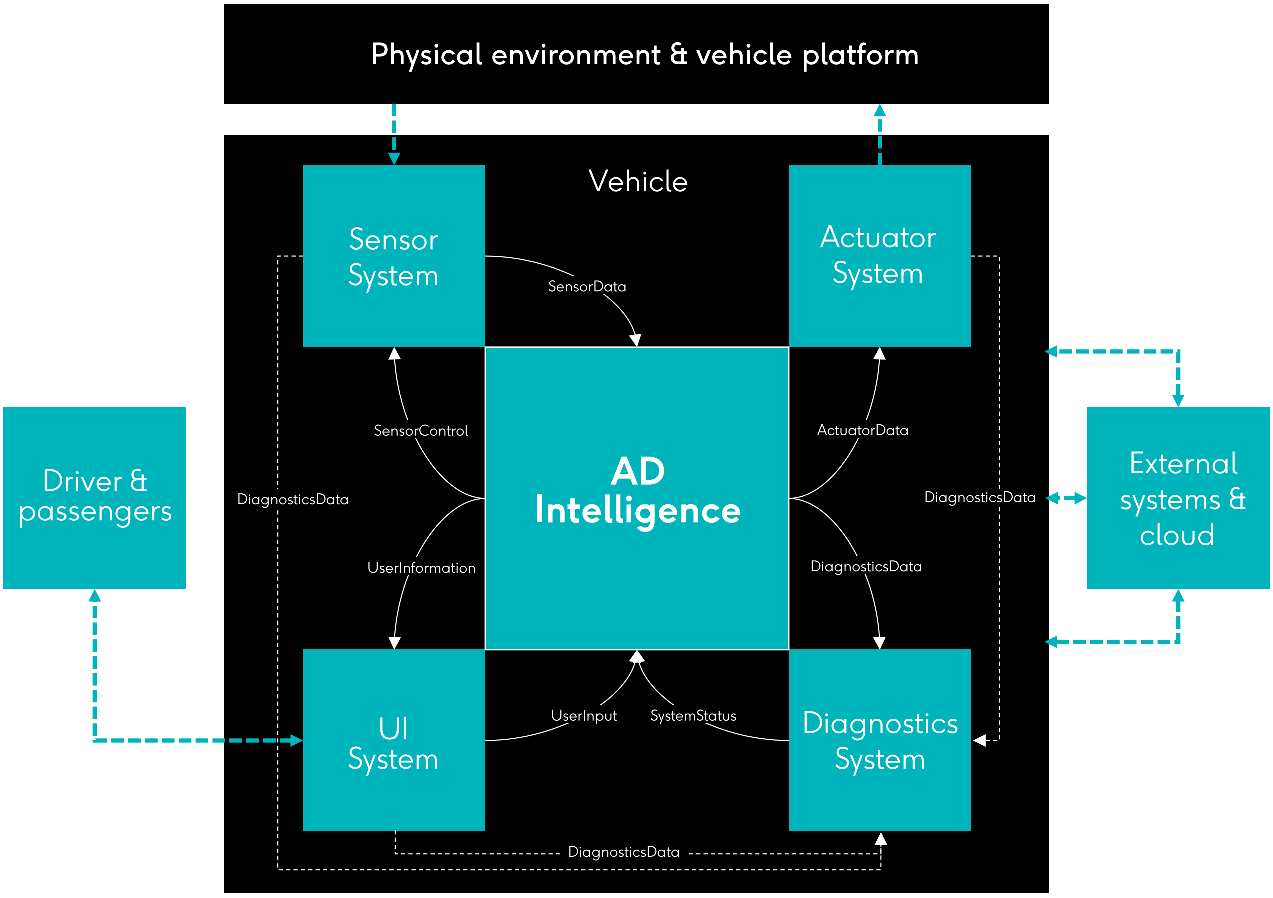


Figure 5: AD Intelligence and its interfaces to surrounding systems.

The AD Intelligence is connected to four other systems (see Figure), which are described in more detail in the following subsections. The main data flow is from the Sensor System to the AD Intelligence and then from the AD Intelligence to the Actuator System (receivers). The AD Intelligence’s main service interface is to the Actuator System. The other service interfaces of the AD Intelligence are mainly for sensor control and diagnostics.

1.2.2 SENSOR SYSTEM

The Sensor System provides the main inputs to the AD Intelligence. It consists of a set of sensors and/or related ECUs (e.g., zonal controllers).

- The Sensor System provides measurement data from a sensor set (SensorData). This interface must be capable of real-time behavior and must be fail-operational (e.g., redundant with absence of dependent failures, encompassing common cause and cascading failures).

- The Sensor System also provides diagnostic information to the Diagnostics System (DiagnosticsData).
- The Sensor System receives calibration and control data (SensorControl).
- The sensor set must be sufficient[5] for the AD Intelligence to offer its service (nominal and degraded functionality). The sensor set comprises "outward-looking" sensors (e.g., radar, camera, lidar, or ultrasonics), "inward-looking" sensors (e.g., IMU), and digital information (e.g., V2X or HD Maps).

1.2.3 ACTUATOR SYSTEM (RECEIVERS)

The Actuator System is the consumer of the service provided by the AD Intelligence. It consists of a set of "receivers", which may be actuator control ECUs and/or smart actuators.

- The Actuator System receives a set of setpoints (ActuatorData). This interface must be capable of real-time behavior and must be fail-operational. On the other hand, the Actuator System might be capable of fail-degraded operation.
- The Actuator System also provides diagnostic information to the Diagnostics System (DiagnosticsData).
- The actuator set must be sufficient to control the vehicle even in the presence of a single fault.

1.2.4 UI SYSTEM

The UI System allows the user to control the AD Intelligence. Some parts of the UI are safety-critical, e.g., to prevent unintended activation/deactivation or driver monitoring.

- The UI System provides commands such as activation/deactivation requests, acceleration, steering and brake requests, destination input, or pull-over request (UserInput).
- The UI System receives requests and status information such as take-over request, or environment model for the Heads-Up Display (HUD) and presents those to the user (UserInformation).
- The UI System also provides diagnostic information to the Diagnostics System (DiagnosticsData).

1.2.5 DIAGNOSTICS SYSTEM

The Diagnostics System collects status information from all systems in the vehicle and may also contain data recording functionality (logging and/or black box). In contrast to traditional automotive diagnostics, the Diagnostics System we refer to here is focused on the AD operation and should be seen as an abstraction of existing and required (new) features. At least part of this system needs to be onboard the vehicle.

- The Diagnostics System provides status information such as detected malfunctions in other systems (SystemStatus).
- The Diagnostics System receives status information from all other systems (DiagnosticsData).

[5] "Sufficient" covers multiple aspects, which go beyond the scope of the "Safety & Architecture" Working Group. Among these are that the sensor set needs to have a sufficient coverage area (detection range) for the intended functionality, needs to be able to detect all relevant objects (e.g., via employing different sensor types), and needs to be robust to faulty sensors (e.g., via redundancy).

1.3 SYSTEM REQUIREMENTS

While Section 1.1 describes the functionality offered by the AD Intelligence from a user perspective, we define assumed high-level requirements regarding the services offered by the system from a technical perspective in this section.

S1: AD INTELLIGENCE OUTPUT TIMELINESS

ID	Statement	Notes
S1	The AD Intelligence shall provide outputs to the Actuator System (receivers) in a timely manner.	„Timely manner" is here used to refer to fast enough (for the dynamics at hand) and predictably (e.g., with sufficiently low jitter, and in every cycle)

S2: AD INTELLIGENCE OUTPUT AVAILABILITY

ID	Statement	Notes
S2	The AD Intelligence shall provide outputs to the Actuator System (receivers) in a fail-operational way to each receiver.	„In a fail-operational way" means that the AD Intelligence continues to perform its nominal function or a degraded function in the presence of any one single fault.

S3: AD INTELLIGENCE OUTPUT CORRECTNESS

ID	Statement	Notes
S3	The AD Intelligence shall not provide erroneous outputs to the Actuator System (receivers), implying that appropriate error detection, error handling or fault-masking should be introduced to reduce the likelihood of propagating failures (stemming from errors within the AD Intelligence).	Allowing an erroneous output to reach the actuators would lead to potential harm to the passengers or other traffic participants, e.g., due to a collision.

S4: AD INTELLIGENCE OUTPUT CONSISTENCY

ID	Statement	Notes
S4	The AD Intelligence shall enable the Actuator System (receivers) to ensure the consistency of executed actuator setpoints.	This applies to consistency between the setpoints executed by redundant receivers / actuators, even for the case where multiple communication channels are used, possibly connecting to multiple receivers.

S5: PERCEPTION MALFUNCTION DETECTION

ID	Statement	Notes
S5	The AD Intelligence shall implement strategies to detect and react to perception malfunctions and performance limitations due to environmental conditions or other causes related to the Sensor System.	This is not expected to be a differentiating factor between different conceptual architecture candidates. Perception (and also localization, prediction, and ODD detection) is outside the scope of this study, but nevertheless considered at the functional level in section 2.8

S6: AD INTELLIGENCE DIAGNOSTICS

ID	Statement	Notes
S6	The AD Intelligence shall report its status to the Diagnostics System.	This is not expected to be a differentiating factor between different conceptual architecture candidates.

1.4 ABSTRACTION LEVEL

The discussion of system architecture can occur on several abstraction levels, which may be suited better or worse to the consideration of certain issues. In the following, we outline the levels relevant to the Safety & Architecture Working Group.

On a **high abstraction level**, we talk about **“conceptual architectures”**. Here, the system is composed of a small set of well-encapsulated subsystems that fail independently (so-called “Fault Containment Units” or FCUs). Each subsystem can comprise parts of a processing channel or even an entire processing channel (sensors to actuators). A point of particular interest on this abstraction level is how to achieve and manage sufficiently independent redundancy within the system.

On a **low abstraction level**, we talk about **HW and SW architectures**. Here, the system is composed of a potentially large set of HW and SW components, which may be highly particular to the specific implementation and system vendor.

The Safety & Architecture Working Group focuses on the discussion of conceptual architectures for two main reasons:

- Conceptual architectures are sufficiently non-trivial, i.e., a reference solution and industry-wide cooperation can provide benefits to system owners. Identifying the underlying principles for achieving high integrity and high availability and combining them in a transparent way leads to a better understanding.
- Conceptual architectures are sufficiently generic, i.e., a reference solution can be applicable to most system owners. Taking vendor-specific constraints, e.g., commercial considerations or integration with legacy systems, into account is shifted to the specific HW and SW implementation.

Specific HW or SW architectures are not considered. Not only would implementation-specific considerations constrain applicability and distract from the underlying principles of the conceptual system architectures, but they are also likely to quickly become obsolete. However, we will identify considerations that apply when mapping a conceptual architecture to HW and SW in order to ensure the desired system properties.

1.5 GENERAL CONSTRAINTS AND DESIGN PRINCIPLES

When coming up with conceptual system architectures intended to satisfy the system requirements in section 1.3, several aspects should be considered:

- There are certain basic technological limitations which constrain how very high reliability systems can be designed, built using realistic HW and SW components, and tested. Such general constraints are summarized in section 1.5.1.
- In addition, there is a set of empirical best practices that should be respected in a sound conceptual system architecture. Such design principles are summarized in section 1.5.2.

1.5.1 GENERAL CONSTRAINTS

G1: DESIGN FAULTS IN LARGE AND COMPLEX MONOLITHIC SYSTEMS

ID	Statement	Notes
G1	We assume that it is impossible to find all design faults in a large and complex monolithic SW system.	Including sample omissions and biases in machine learning training.

Rationale

- A SW system with more than ~10k lines of code will statistically contain at least one SW fault despite adequate testing [10] [11]. This does not mean that a SW system with fewer lines of code will necessarily be free from faults with adequate testing (e.g., control flows can still be complex). Heisenbug type faults [12] [13], which may appear to “alter” its behavior when attempting to investigate or reproduce it, can prove particularly hard to detect and eliminate. A typical example of a Heisenbug fault is a race condition in concurrent software.
- The ADI can be assumed to contain several subsystems that each contain more than a million lines of source code.

G2: SINGLE-EVENT UPSETS IN NON-REDUNDANT HW

ID	Statement	Notes
G2	We assume that it is impossible to mitigate single-event upsets (SEUs) in non-redundant HW.	While the architectural evaluation in this report will not go to a detailed hardware level, the implication is that errors due to SEUs must be considered in the system-level architecture design.

Rationale

- SEUs are caused by ionizing particles, e.g., cosmic radiation, which affects electronic devices such as processors or memory. The impact of this depends on the executed SW, but in the worst case, e.g., for brittle neural networks, even a single bitflip can lead to misclassification [14].

G3: TESTING AND SIMULATION OF VERY HIGH SAFETY-RELATED AVAILABILITY OF LARGE MONOLITHIC SYSTEM

ID	Statement	Notes
G3	We assume that it is impossible to establish the very high safety-related availability of a large monolithic system by testing and simulation alone.	The order of magnitude considered here is similar to the rate of random HW faults for ASIL D.

Rationale

- Depending on the testing assumptions, it would take hundreds of millions to hundreds of billions of miles driven to demonstrate the reliability of autonomous vehicles [15] [16]. Additional methods like simulations can only alleviate this to some extent.

G4: SPECIFICATION OF CRITICAL SCENARIOS

ID	Statement	Notes
G4	We assume that it is impossible to precisely specify all critical scenarios that can be encountered within the ODD specified for automated driving.	The corresponding risks can be reduced by guidance from relevant standards such as SOTIF and UL4600, including through ODD and field monitoring.

Rationale

- The challenge relates to the „open" environments of many ODDs and moreover to the fact that traffic behavior will change as AVs are introduced [17].
- The field of automated driving is relatively new compared to the aerospace industry. Even in the comparably "simple" environment of the sky, it took several decades of collecting and studying rare and exceptional situations to establish similarly high dependability.
- Examples for such edge cases are rare traffic participants (costumed pedestrian, vehicle with odd shape, vehicle with sky blue paint or mirror finish, etc.), rare events (complex traffic accident, confusing lost load on road, etc.), or rare environmental conditions (moon in ash cloud, etc.).

G5: FREQUENT SWITCHING

ID	Statement	Notes
G5	We assume that it is unsafe to frequently switch back and forth between the trajectories generated by different subsystems.	It is known from basic control theory that „bumpless transfer" requires some form of interaction between controllers involved in switching. Trajectories generated by different subsystems might not implement the same driving strategies, e.g., with respect to passing obstacles vs. braking.

Rationale

- Under certain conditions, switching back and forth between the trajectories/sets of setpoints of two subsystems may lead to unsafe behavior, e.g., when "mixing" evasive action to the left and the right and thus never moving far from the center.

G6: CHECKS TO DETERMINE PLAUSIBILITY OF ANOTHER SUBSYSTEM

ID	Statement	Notes
G6	We assume that it is possible to develop a subsystem which can assess the plausibility of the proposed trajectory produced by another subsystem based on the former's internal environment model.	No subsystem will have access to ground truth but is in principle able to assess the perceived correctness of the trajectory provided by another system. Comment: Special focus must be put on eliminating dependent failures. With perceived correctness of the trajectory, we refer to satisfying certain safety requirements (trajectory verification).

Rationale

- There are many ways of designing safety mechanisms that cover the essential safety goals of the AD Intelligence. Instead of the trajectory, checking may also apply at the level of a set of setpoints for the Actuator System.
    - Proposed trajectories can be checked against another environment model (than the one that was used to generate it), i.e., whether certain safety goals are violated.
    - Proposed sets of setpoints can be checked against another environment model and against the corresponding proposed trajectory, i.e., whether the two are consistent.
    - A runtime environment model can be checked for violations of assumptions or of the ODD.

G7: RATE OF SAFETY INCIDENTS

ID	Statement	Notes
G7	We assume that the target rate of hazardous behavior for the AD Intelligence functionality covers random faults, systematic faults, and functional insufficiencies.	This assumption is included to be able to reason about architectural candidates, reflecting a failure rate that does not take into account the causes of malfunctions.

Rationale

- While the failure rate targets in ISO 26262 only apply to random HW faults, the dependability goals for the AD Intelligence apply jointly to hazards arising from random and systematic HW faults, systematic SW faults, and insufficient specifications or performance limitations (SOTIF).

G8: IMPACT OF SYSTEM FAILURE

ID	Statement	Notes
G8	We conservatively assume that all failures lead to hazardous behavior of the AD Intelligence, leading to accidents in the worst case.	This is a pessimistic assumption, but conservativeness was deliberately chosen to be on the safe side. While some failures, e.g., a collision trajectory, will be highly hazardous, other errors may not impact risk to a large extent (e.g., a slightly altered trajectory).

Rationale

- This applies only to situations where the AD Intelligence as a whole fails. The failure of single subsystems can be compensated for by the conceptual architecture.
- While active, the AD Intelligence replaces the human driver. However, according to traffic statistics [15] [18], only a small fraction of reported accidents (human-driven cars) involves fatalities (0.1-0.2%) or severe injuries. In addition, a significant fraction of minor accidents is not even reported to authorities (25-60%) [15] [19]. Most wrong decisions made by human drivers thus do not have severe consequences. We cannot make the outright assumption that the severity distribution in accidents caused by human drivers is in any way similar to those caused by an AD Intelligence; however, it is clear that not all failures of the AD intelligence will lead to fatal accidents.
- The assumption is related (complementary) to the "improvement factor" demanded of an AD Intelligence over the average human driver.

G9: HW FAULT CHARACTERISTICS

ID	Statement	Notes
G9	For FIT rate calculations, we assume that HW failure rates are constant over time (i.e., not a function of time), and that there are no dependent failures (i.e., that no cross-correlations between different subsystems exist).	HW FIT rate calculations focus on random HW faults only. Systematic HW faults have to be considered in a different context.

Rationale

- This explicitly excludes the consideration of early failures or wear-out/aging effects ("bathtub" curve).

G10: INTERFERENCE FROM OTHER SYSTEMS

ID	Statement	Notes
G10	We assume that other safety-related systems do not interfere negatively with the AD Intelligence.	Alternatively phrased, we assume an architecture which coordinates the safety-related behavior of the vehicle.

Rationale

- Inputs from other safety-related systems, e.g., Automated Emergency Braking (AEB) or similar, can be overridden on the Actuator System side while the AD Intelligence is in operation.

1.5.2 DESIGN PRINCIPLES

D1: FAULT CONTAINMENT UNITS

ID	Statement	Notes
D1	The AD Intelligence shall consist of a set of independent subsystems that each form a Fault Containment Unit (FCU).	Special emphasis needs to be placed on avoiding dependent failures. The appropriate strategies for achieving this depend on the complexity of the subsystem and the consequences that dependent failures can cause.

Rationale

- See G1: Design faults in large and complex monolithic systems, G2: Single-event upsets in non-redundant HW, and G3: Testing and simulation of very high safety-related availability of large monolithic system.
- To reduce the complexity of a large system, one of the simplest and most robust techniques is to allocate separable functions to subsystems that can be shown to be as independent from each other as possible [10]. Such subsystems should form FCUs, which can be validated separately.

D2: SIMPLE AND COMPLEX SUBSYSTEMS

ID	Statement	Notes
D2	The conceptual architecture of the AD Intelligence shall distinguish between simple subsystems (fully verifiable – preferably with formal techniques – and deterministic, e.g., due to being formally specified, having few lines of code, and avoiding algorithmic complexity) and complex subsystems.	

Rationale

- See G1: Design faults in large and complex monolithic systems, G3: Testing and simulation of very high safety-related availability of large monolithic system, and G4: Specification of critical scenarios.
- Simple subsystems should be developed fully to ASIL D, be fully formally specified (to preclude Byzantine faults[6] during runtime), and contain a relatively small number of lines of code (i.e., thousands, not millions).

D3: DIVERSITY AND REDUNDANCY FOR COMPLEX SUBSYSTEMS

ID	Statement	Notes
D3	The complex subsystems of the AD Intelligence shall be diverse in design.	Groups of redundant subsystems may have similar or identical purposes; in which case they should have different designs.

Rationale

[6] See https://en.wikipedia.org/wiki/Byzantine_fault.

- See G1: Design faults in large and complex monolithic systems, G3: Testing and simulation of very high safety-related availability of large monolithic system, G4: Specification of critical scenarios, D1: Fault Containment Units, and D2: Simple and complex subsystems.
- Complex subsystems must be assumed to exhibit Byzantine faults, i.e., inconsistent or arbitrary behavior when faulty. Due to their size and complexity, design faults and HW failures become inevitable and must be addressed by employing redundancy and design diversity.

D4: PROVABLE CORRECTNESS FOR SIMPLE SUBSYSTEMS

ID	Statement	Notes
D4	The simple subsystems of the AD Intelligence shall be sufficiently simple such that they are fully verifiable (formally specified, few lines of code).	It is assumed that the simple subsystems will be concerned with arbitration involving logic.

Rationale

- See G1: Design faults in large and complex monolithic systems, D1: Fault Containment Units, D2: Simple and complex subsystems.
- It is difficult to achieve replica determinism, i.e., identical behavior from two instances of the same implementation, for complex subsystems. However, this can be achievable for relatively simple decision logic, using simple, fully verifiable SW running on fault-tolerant HW.

D5: AVOIDANCE OF EMERGENT BEHAVIOR

ID	Statement	Notes
D5	The conceptual architecture shall minimize interactions among the different subsystems.	

Rationale

- See G3: Testing and simulation of very high safety-related availability of large monolithic system and G4: Specification of critical scenarios.
- As establishing the very high dependability of a monolithic system is not feasible, it is necessary to provide evidence of each constituent subsystem's dependability separately. Such an effort is vastly facilitated if these subsystems are coherent and avoid emergent behavior when interacting with other subsystems.

D6: TRANSIENT AND PERMANENT FAULTS

ID	Statement	Notes
D6	If the AD Intelligence detects a large number of transient faults within one of its subsystems, it shall consider this a permanent fault in this subsystem.	

Rationale

- See G6: Checks to determine plausibility of another subsystem.
- When transient faults occur too often, it is reasonable to consider this a permanent fault and to react appropriately (e.g., request driver to take over and/or execute an MRM).

D7: MITIGATION OF COMMON-CAUSE HAZARDS

ID	Statement	Notes
D7	The conceptual architecture shall minimize the possible propagation paths of hazards by mitigating against common-cause faults and functional insufficiency across the design pattern [20] [21].	Adaptation of the Swiss-cheese model in Figure 6 and Figure 7 is proposed to guide awareness regarding propagation of hazards through system channels. It furthermore supports the abstract design goal formulation of minimizing overlap of the holes. An example is provided in Appendix C: Sample analysis points regarding different conceptual architecture patterns.

Rationale

- The various hazards that can lead to ADS losses are a function of the ODD and use case. Functional insufficiency is a major (if not majority) contributor to hazardous ADS behavior [22]. There is an opportunity to address them at the design level too, not just within V&V efforts.
- Each channel of the architecture pattern can be characterized by capabilities. As introduced in the SaFAD whitepaper [4], these capabilities can be understood as being the fundamental set of system properties that are responsible for safety (nominal/degraded functionality – implemented via elements). The channel's functional complexity is indicated by the depth of the slice, and the operational domain coverage (i.e., ODD, Operational Domain, or Target Operational Domain) is indicated by its area.
- Output insufficiency is a lack of the capability that is intended to be provided; shared output insufficiency between channels allows triggering conditions to manifest as losses and are analogous to the (un-)known unsafe area in SOTIF [22]. Likewise, shared errors between channels allow fault root causes to manifest as losses.
    - Therefore, the conceptual design goal of minimizing the shared lack of capability across channels can be formulated. The conceptual dimensioning and positioning of errors and output insufficiency across channels must be well understood for mitigation efforts (such as diverse, heterogeneous implementations) to offer complementary capability.
    - As indicated by the model, it cannot be assumed that efforts to achieve diversity with respect to fault tolerance will also satisfy the diversity required to mitigate functional insufficiency. There must be an awareness that fault-tolerant implementations alone do not exclude the possibility of functional insufficiencies leading to losses.

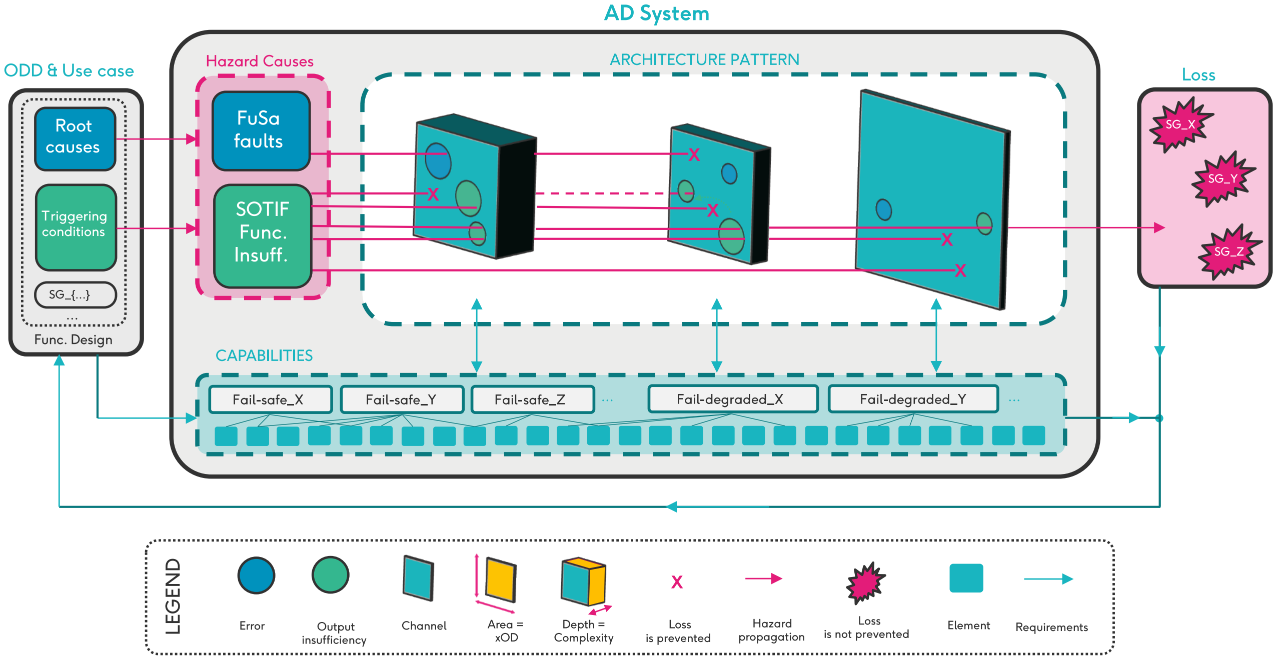


Figure 6: Conceptual architecture-level hazard propagation, as expressed via an adaptation of the Swiss-cheese model

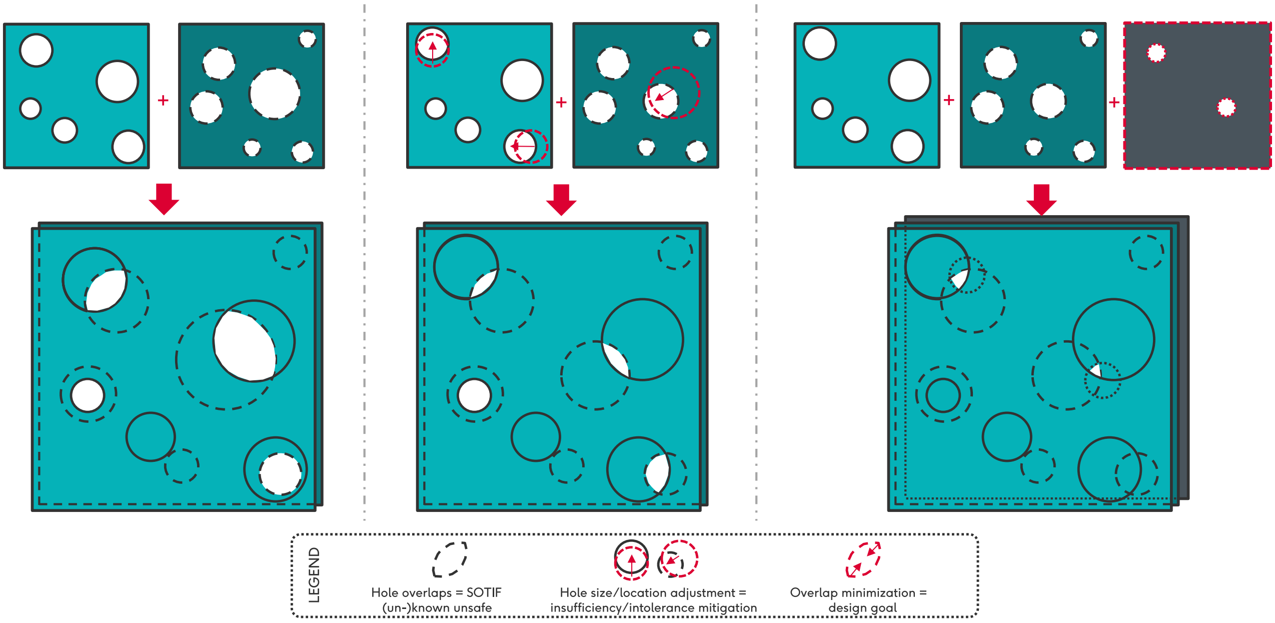

+
LEGEND
Hole overlaps = SOTIF (un-)known unsafe
Hole size/location adjustment = insufficiency/intolerance mitigation
Overlap minimization = design goal

Figure 7: Bird's-eye view - Reduction of hole overlaps (errors, output insufficiency) as a design goal for channel design

2 ARCHITECTURE EVALUATION CRITERIA

2.1 ARCHITECTURAL DECISIONS AND PROCESSES

The term "architecture" can cover both commercial aspects, e.g., as business architectures, and technical aspects. In the latter, it can cover different abstraction levels, e.g., functional architectures, conceptual architectures, logical architectures, down to very specific physical HW and SW architectures.

As stated in section 1.4, the focus of the Safety & Architecture Working Group lies on the conceptual abstraction level. To make sound architectural decisions here, we first need to define a set of evaluation criteria suitable for this abstraction level. These should not exist in isolation, i.e., they should relate to relevant decisions the target reader of this document needs to make. To ensure this relevance, we first outline a persona for the assumed reader, i.e., someone responsible for making architectural decisions as part of a systems design process.

2.1.1 SYSTEM OWNER PERSONA

It lies within the responsibility of "system owners" (often system architects), whom we consider the intended readers of this document, to ensure a consistent systems design across all abstraction levels (recall Figure 2). In the following, we outline the system owner persona.

The system owner can work for an OEM, a mobility company, or for a system supplier:

- Large and/or technologically leading OEMs may try to bring most of the architectural design in-house. In this case, the system owner needs to make all architectural decisions, perform mapping between abstraction levels, and ensure consistency.
- Small and/or technologically following OEMs, as well as mobility companies, may try to buy off-the-shelf system solutions. In this case, many architectural decisions are made by the system supplier, but the system owner still needs to understand the different architecture perspectives to pick a suitable system solution.
- System suppliers are often focused on providing off-the-shelf HW platforms but may also extend to SW platforms and application SW solutions. In this case, the system owner may need to demonstrate to prospective customers that the offered solutions can be combined into a suitable AD system.

1.1.2 ARCHITECTURE DESIGN PROCESS AND DECISIONS

Textbook systems design should start at the top-most, user-focused level and then – step by step – become more and more detailed and specific as the design is refined. This may involve some of the following steps (see also Figure 8):

- High-level users and use cases are defined.
- Use cases are broken down into high-level system requirements.
- The system requirements are used to develop the high-level systems design.
- The systems design is used to derive more detailed application SW requirements.
- The application SW requirements are used to develop the application SW design.
- The application SW design is used to derive requirements for the SW platform and HW platform.

- The platform requirements are used to develop the SW and HW platform designs.

In practice, the architecture design process is often not top-down. Several factors can contribute to this:

- An incomplete understanding of the problem space or insufficient domain knowledge may necessitate building a prototype before writing requirements.
- Emergent properties in the environment (e.g., the environment changing when exposed to the system) can also only be understood once a prototype is in the field.
- External constraints and commercial considerations (e.g., the much longer lead times and in HW development) can also shape the design before requirements are even known. In addition, legacy constraints may also come into play.

Working bottom-up can lead to situations where the design of the HW platform constrains the design of the application SW and ultimately also the conceptual architecture.

The system owner must make architectural decisions at each of the steps described above:

- What is a suitable conceptual architecture for the particular use cases?
- What is a suitable SW architecture for the particular use cases? Does it match the conceptual architecture? Is it commercially viable?
- What is a suitable HW architecture for the SW stack? Does it match the conceptual architecture? Is it commercially viable?
- Which of the available system solution offerings is suitable for the particular use cases?

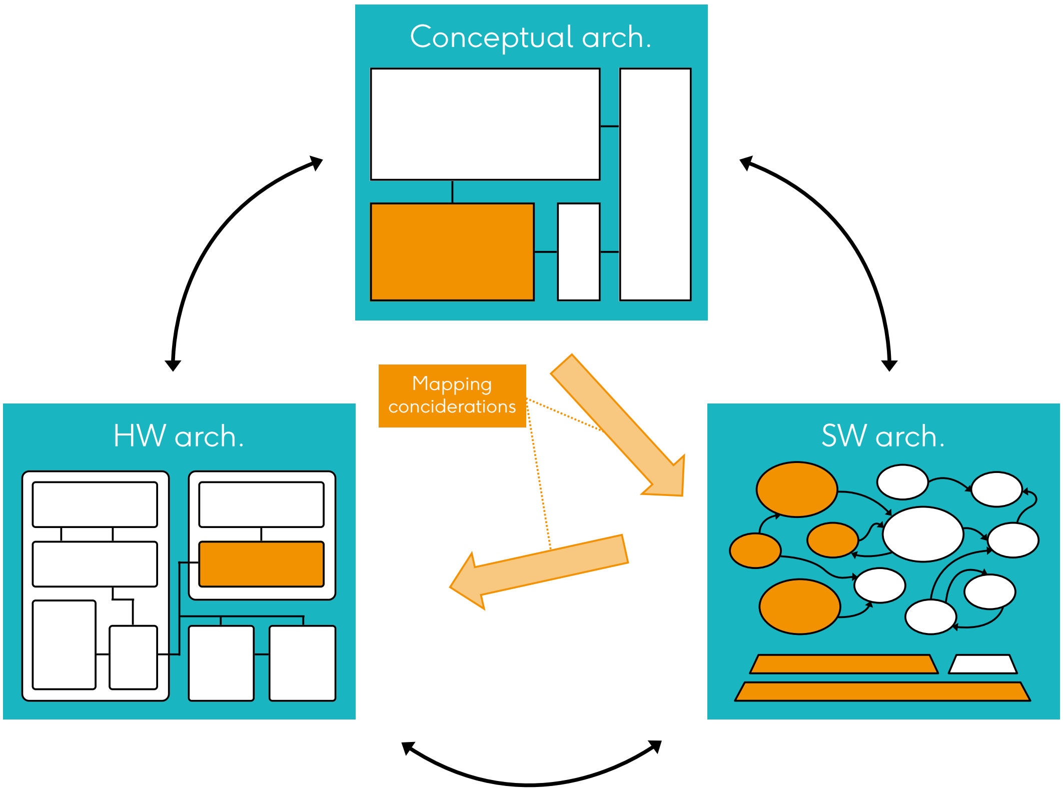


Figure 8: Idealized mapping process between different architectural abstraction levels.

2.2 GENERAL REQUIREMENTS

There are many properties that a well-designed AD Intelligence needs to have. Only some of these are suitable for differentiating different architectures on the conceptual level. Many of the attributes applicable at the physical level can be assumed to be present as long as the mapping of conceptual architecture elements to HW and SW components is done properly, and automotive development processes are followed.

For completeness, we list some of these properties in the following.

2.2.1 AUTOMOTIVE QUALITY

All components used in the AD Intelligence need to satisfy the usual automotive quality standards such as AEC-Q100 to ensure suitability for automotive use cases. This can involve robustness to shocks, high and low temperatures, etc.

2.2.2 ADHERENCE TO STANDARDS

There are several industry standards that need to be followed in the development and production of the AD Intelligence. The ones that immediately come to mind are ISO 26262 (Functional Safety) [2], ISO 21448 (Safety of the Intended Functionality) [3], ISO 21434 (Cybersecurity) [6], UL 4600 (Safety Case Assessment) [23], and SAE J3018 (Safety of On-Road Testing) [24].

2.2.3 FIELD MONITORING AND UPDATE PROCESS

Even with the most rigorous safety development process, a sufficiently complex system will almost inevitably have flaws that were underestimated or unforeseen. Therefore, it is necessary to continuously monitor vehicles in the field and analyze the collected data, e.g., to establish that assumptions made in the safety analysis continue to hold true over the lifetime of the vehicle. Flaws can be addressed by providing timely updates to minimize exposure to both safety and security vulnerabilities.

2.2.4 COMFORT AND FUNCTIONALITY

Ultimately, the AD feature needs to provide benefits to the end user. This implies that the AD function controls the vehicle in a manner that is both comfortable (e.g., low acceleration and low jerk) and beneficial to the passengers (e.g., a useful speed limit).

2.2.5 MODULARITY AND MAINTAINABILITY

Road vehicles often have an intended minimum economically viable lifetime of around 15 years[7]. Over such an extended period, it is likely that several components, particularly complex ones such as high-performance electronics, will need to be maintained or replaced. As AD systems and their components are relatively expensive, it is advantageous to design them in a modular (and thus more easily maintainable) manner.

2.2.6 PHYSICAL IMPLEMENTATION

Some attributes are specific to the physical implementation of the AD Intelligence. In general, the Electronic Control Units (ECUs) involved in the AD functionality need to be sufficiently small to fit inside the constrained internal space of the vehicle. They also need to have sufficiently low

[7] Of course, many vehicles continue in service much longer.

power consumption to not have a severe impact on the range of electric vehicles and/or cause issues with heat dissipation. Finally, the affordability of the system should also not be neglected.

2.2.7 SAFETY

The AD Intelligence must be developed to the highest applicable level as defined in ISO 26262 (i.e., ASIL D) and ISO 21448 (see also sections 5.3.2 and 5.3.3, respectively)[8]. There are two elements of safety for a fail-operational/fail-degraded system: the availability of the system, which is the probability that the system keeps operating properly when a failure occurs[9], and the safety of the available outputs itself, which avoids an unreasonable risk due to their execution (e.g., collisions).

ISO 26262 uses the FIT rate (Failures in Time, i.e., per billion hours of operation) as a metric to quantify the occurrence of random HW faults. Other relevant causes for safety incidents such as systematic HW faults, systematic SW faults (bugs), and functional insufficiencies (SOTIF), are mainly addressed by prescribing safety processes[10].

To quantify the required level of safety of the system more comprehensively, we define the total rate of safety incidents (including all the underlying causes listed above) that can lead to unsafe situations (see G7: Rate of safety incidents). This rate of safety incidents for the system can be calculated through a Failure Modes, Effects, and Diagnostics Analysis (FMEDA) and a Fault Tree Analysis (FTA). Based on the reference AD use case, we propose a tentative target for the rate of safety incidents of 10-100 per billion hours of operation (10^{-8} – 10^{-7} per hour).

Different parties from industry and academia have discussed widely varying target rates [25] [26] [27]. These range from ~10^{-9} per hour (or even lower) up to ~10^{-7} per hour. These considerations are often based on the average rate of traffic accidents (or fatalities) for a particular use case (total or just highway) and an improvement factor over the average human driver.

Such a derivation roughly proceeds as follows:

- The rate of reported traffic accidents (fatal and non-fatal) can be estimated from traffic statistics [18] [15] [28]. This varies to a degree between countries and by use case, depending on the typical speed, the traffic situation complexity, and what other traffic participants are involved. The rate of fatal accidents is in the range of 1.7 x 10^{-7} – 5 x 10^{-7} per hour[11].
- The rate of reported non-fatal accidents from the same statistics is typically 100x – 1000x higher, ranging from 7.1 x 10^{-5} to 2 x 10^{-4} per hour. However, it cannot necessarily be assumed that this ratio will be similar for AD. To demonstrate a positive risk balance, we should therefore aim to build an ADI that has fewer safety incidents than humans have fatal accidents (see G8: Impact of system failure).
- The demanded improvement factor over the average human driver depends on public acceptance. Values here can range from as high as 1000x [25], which is used as a reference in aerospace, to as low as 4-5x [29], which people already find acceptable in surveys. An intermediate value of 10x – 100x may be reasonable [26].
- We also need to neglect contributions from other causes that cannot be addressed by the ADI (see Figure 9). Only causes equivalent to the cognitive tasks otherwise performed by the driver can be considered for the target rate of safety incidents.

[7] Of course, many vehicles continue in service much longer.

[8] Through the use of ASIL decomposition, the ASIL for many subsystems and components can be lowered, e.g., to ASIL B(D).

[9] Loss of functionality, e.g., turning the system off in case of a malfunction, can lead to a hazard.

[10] ISO 21448 specifies mandatory qualitative metrics for the residual risk. An example given for this is the maximum number of accidents per hour.

[11] Some of these rates are given in incidents per kilometers driven. When necessary, we assume an average speed of 60 km/h for all driving and 110 km/h for highway driving to convert.

Our tentative target of $10^{-8} – 10^{-7}$ per hour for the rate of safety incidents is an improvement of ~10x (1.7x - 50x) over the rate of fatal accidents and an improvement of ~1000x (710x -20000x) over the total rate of accidents.

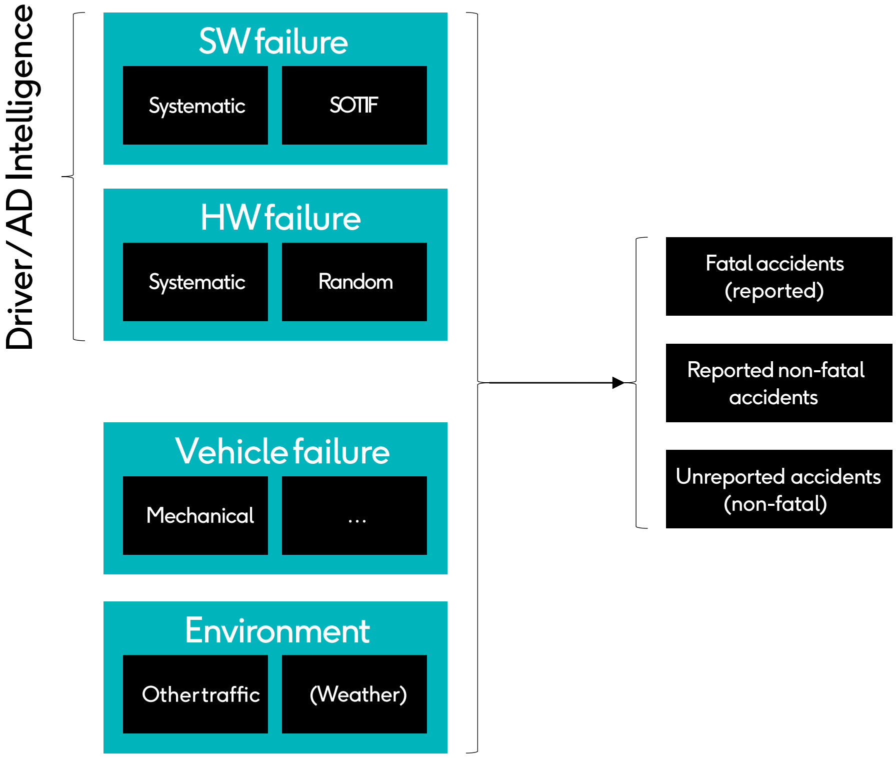


Figure 9: Coarse overview of causes contributing to accidents.

2.3 AVAILABILITY

Because there is no human driver to take over control, the availability of the system, i.e., its readiness for correct service, also becomes crucial. We define three evaluation criteria related to the availability attribute.

2.3.1 AVAILABILITY OF THE SYSTEM

A suitable conceptual system architecture must take Safety-Related Availability (SaRA) into account. This means that it is designed in such a way that no single fault can lead to the failure (or unavailability) of the entire AD Intelligence. At least some degraded functionality needs to be available and dispatchable. Key questions related to this are:

- Does the conceptual system architecture maintain safety (integrity and availability) in the presence of any single fault or functional insufficiency[12]?
- Does the conceptual system architecture also cover all sufficiently probable dual- and multi-point faults (including common cause faults)?

If the conceptual system architecture scores badly on these questions, the system owner should consider it unsuited for AD use cases where unavailability of the system is inherently unsafe, which is most driving situations other than parking.

[12] The definition of single-point faults in ISO 26262 only covers HW faults, whereas we also consider SW faults and functional insufficiencies.

2.3.2 DIAGNOSTICS SCHEME

If the different subsystems have self- or cross-checking diagnostic capabilities, they can facilitate degradation schemes in the AD Intelligence (see section 2.3.3). This enables them to react dynamically to each other's condition, e.g., by proactively switching to a more cautious course of action. Key questions related to this are:

- Are the different subsystems aware of each other's condition?
- Can the different subsystems adapt based on each other's condition?

If the conceptual system architecture scores badly on these questions, the system owner should consider the increased burden on the degradation scheme.

2.3.3 DEGRADATION SCHEME

While a failure of the AD Intelligence needs to be prevented at all costs (and thus be exceptionally rare), failures of a single subsystem will be much more frequent. This can necessitate switching to a degraded mode, where the AD Intelligence executes an MRM [67][13]. If this occurs too frequently or unnecessarily (e.g., due to a transient or recoverable fault), it can adversely affect both the user experience and public safety (e.g., due to blocked public roads). Key questions related to this are:

- How noticeable is it to the end user when an error occurs in the system?
- Are different levels of degradation possible and how graceful are these?
- Is cold / warm / hot standby used (good for availability but bad for power consumption)?

If the conceptual system architecture scores badly on these questions, the system owner should consider the increased burden on the integrity of the implemented function, as no additional lines of defense may exist.

There may be additional practical criteria such as minimizing risk redistribution onto vulnerable population segments, but such issues are beyond the scope of this report.

2.4 RELIABILITY

Whenever degradation is used in the system (see also section 2.3.3), the full, nominal functionality is no longer available. This has a noticeable impact on the user experience. In particular, frequent transient faults and/or false positives should not lead to unnecessary degradation. Therefore, the reliability of the AD Intelligence, i.e., its continuity of correct service, is important. We define one evaluation criterion related to the reliability attribute.

2.4.1 AVAILABILITY OF THE NOMINAL FUNCTIONALITY

A suitable conceptual system architecture is based on concepts that prevent unnecessary degradation, ensuring that the nominal functionality of the AD Intelligence is available as much as possible. This is also related to the redundancy management scheme, which is based on some kind of arbitration and ultimately decides the behavior of the system based on a limited set of inputs. Arbitration algorithms can be relatively simple, e.g., a simple silencing function in Doer/Checker, or rather complex, e.g., inexact voting algorithms. Complex arbitration algorithms may be difficult to implement in a robust way, potentially outweighing benefits from

[13] Degradation schemes can have several levels, which are progressively less safe and desirable. Schemes have been proposed to quantify such cascades and the respective acceptable probabilities of each level [67].

achieving a simpler conceptual architecture (see also sections 2.3.1, 2.3.3, 2.5.1, and 2.7.1). Key questions related to this are:

- Is the system prone to false positives that make the nominal functionality unavailable?
- Is the system sufficiently reliable to avoid creating nuisances like blocking public roads?
- Do the arbitration algorithms require complex and abstract decisions?
- Can these decisions be converted to pseudo-code and broken down into manageable logical statements?

If the conceptual system architecture scores badly on these questions, the system owner should consider the need for a redesign of the system or alternatively the increased burden on the quality of the primary functionality. This may require significantly higher testing efforts.

2.5 CYBERSECURITY

While the focus of the Safety & Architecture Working Group is on safety and we consider a detailed cybersecurity analysis outside our scope, some aspects of conceptual system architectures have an indirect impact on security considerations. We define two evaluation criteria related to the cybersecurity attribute.

2.5.1 INTERACTIONS BETWEEN SUBSYSTEMS

A suitable conceptual system architecture consists of several well-encapsulated subsystems that ensure that faults arising within them do not propagate to the rest of the system, i.e., Fault Containment Units (FCUs). Similar considerations apply from a security perspective, i.e., where few and well-defined interfaces between subsystems are beneficial. Key questions related to this are:

- How many communication interfaces are there between the different subsystems?
- How frequent and extensive (bandwidth) are these interactions?
- Are well-defined and restricted interfaces used?

If the conceptual system architecture scores badly on these questions, the system owner should consider that the security concept must more extensively consider the case where multiple subsystems are compromised simultaneously via propagation.

2.5.2 INTERACTIONS WITH EXTERNAL SYSTEMS

It is generally assumed that the AD Intelligence will need to interact with external systems, e.g., for map and traffic data, V2X, or to receive updates. Reducing the number of subsystems that are involved in this can help reduce the attack surface of the system. Key questions related to this are:

- Which subsystems need to communicate with the outside world?
- How often and for what purposes (HD maps, updates, etc.) is this communication necessary[14]?
- Which subsystems require updates and how often? Do they use the same update mechanisms?

If the conceptual system architecture scores badly on these questions, the system owner should consider that the security concept must more extensively consider the case where multiple subsystems are compromised simultaneously.

[14] This may depend on the use case and ODD, and may also change over time.

2.6 SCALABILITY

From the perspective of the system owner, a particular implementation of the AD Intelligence is not developed in isolation.

- Carrying over already developed systems (or components thereof) can provide huge savings in money and time.
- In addition, most OEMs aim to address different market segments and are therefore interested in multiple (and hopefully scalable) offering levels. These can range from legally required NCAP functionality to premium AD or even driverless functions (e.g., MaaS/robotaxis).

We consider both of these as parts of a scalability attribute, for which we define two evaluation criteria.

2.6.1 SCALABILITY TOWARDS HIGHER AVAILABILITY

AD features classified as SAE Level 4 and above, which are the scope of the Safety & Architecture Working Group, can vary widely, implying vastly different availability goals. For a Highway Pilot feature, remaining available for tens of seconds and coming to a controlled stop is considered sufficient. However, a fully driverless vehicle may require some limp-home functionality, i.e., continuing driving for dozens of minutes up to hours. In the ideal case, the conceptual system architecture can be scaled depending on the availability (or integrity) levels required by a particular use case. Key questions related to this are:

- Does the architecture support higher availability goals than what is necessary for the reference AD use case, e.g., for driverless use cases?
- Which subsystems would be added to achieve this?

If the conceptual system architecture scores badly on these questions, the system owner should consider that it may be difficult to re-use it for more elaborate AD use cases at a later point in time. It may then be necessary to switch to a different conceptual system architecture.

2.6.2 SCALABILITY TOWARDS DIFFERENT OFFERING LEVELS

If multiple price segments or offering levels have to be addressed, it is highly advantageous from a cost perspective to develop all such systems jointly. Higher offering levels (offering AD features) can then be developed as extensions of lower ones (e.g., ADAS features) or vice versa. Such systems may even be similar from a functionality perspective (e.g., both performing highway driving with lane changes at up to 130 km/h) and only differ from an integrity and availability perspective (e.g., requiring supervision from an attentive driver or not). Key questions related to this are:

- Does the architecture support reusing ADAS (with minor modifications) as a subsystem (role and provided functionality)?
- Which subsystems are specific to SAE L3/L4 use cases?

If the conceptual system architecture scores badly on these questions, the system owner should consider that this may entail higher development costs.

2.7 SIMPLICITY

While we do not consider physical implementation options as part of the Safety & Architecture Working Group, some aspects of conceptual system architectures have a pronounced – though indirect – impact on this. Complex architectures with tightly coupled subsystems are generally harder to implement, validate, and verify. Ideally, architectures should be sufficiently simple such that they can be easily understood, and their subsystems can be developed and validated independently of each other. The latter is particularly important as testing a black box system to the required failure rates for AD is nigh impossible (see also G1: Design faults in large and complex monolithic systems). We define three evaluation criteria related to the simplicity attribute.

2.7.1 NUMBER, COMPLEXITY, AND PERFORMANCE OF SUBSYSTEMS

As stated before, suitable conceptual system architectures should consist of loosely coupled, cohesive subsystems (see section 2.5.1). As long as the number of subsystems and interactions is relatively low (e.g., manageable with current methodologies), emergent behavior can be more easily prevented. The development and HW costs of each subsystem depend more strongly on its internal complexity and performance requirements. This can range from essentially a smart switch with minimal logic to high-performance, AI-based subsystems for perception and planning. Key questions related to this are:

- How many subsystems exist in the system (also implying development and HW costs)?
- How complex are these subsystems (e.g., involving ML/AI-based approaches or algorithms that are hard to implement or calibrate properly, also implying SW implementation cost)?
- What are the performance requirements of these subsystems (also implying power consumption and HW cost)?

If the conceptual system architecture scores badly on these questions, the system owner should be aware that the cost to implement and manufacture a corresponding physical architecture is likely to be higher.

2.7.2 REQUIRED DIVERSITY

Ensuring that multiple subsystems do not fail simultaneously due to systematic faults and/or functional insufficiencies (see also G7: Rate of safety incidents) poses a pronounced new challenge in AD. On the level of a conceptual system architecture, this generally requires asking for some level of diversity between subsystems. Exploiting asymmetries, e.g., by making use of Doer/Checker approaches, can make it easier to ensure this. Key questions related to this are:

- Between which subsystems is diversity required (also implying increased development costs)?
- Are these complex and high-performance subsystems where not many different suppliers or approaches exist?

If the conceptual system architecture scores badly on these questions, the system owner should be aware of the additional cost and difficulty to implement provably diverse SW.

2.7.3 COMPLEXITY OF VALIDATION

A well-known challenge in AD is how to demonstrate that the system is safe enough. To do this, testing is necessary – though not sufficient. The associated effort scales dramatically with the target failure rate of the system or subsystem. Key questions related to this are:

- Can subsystems be validated independently from each other?
- If so, does the required validation effort decrease significantly (e.g., 10^{-8} per hour/100 million hours for testing of the integrated system → 10^{-6} per hour/1 million hours for each isolated subsystem)?
- What is the complexity of ensuring the absence of correlated or common cause failures between subsystems?

If the conceptual system architecture scores badly on these questions, the system owner should be aware that testing will pose a significant challenge.

2.8 SAFETY OF THE INTENDED FUNCTIONALITY (SOTIF)

To ensure an acceptable level of dependability for AD/ADAS systems, the analysis of SOTIF aspects must be included in architectural design decisions from the beginning. Although the impact of SOTIF on the conceptual architecture of ADs has not been sufficiently examined, we propose some ideas that could help to determine whether particular architectures have the potential to better support SOTIF.

We focus on the idea that each channel must be designed to ensure a safe vehicle behavior in all the expected operational conditions depending on its functional responsibility (e.g., nominal, or fallback capabilities). Then, the safe interaction between the different architectural elements shall be ensured for system safety. For this, dedicated components supporting SOTIF-related tasks are required. The analysis of the different modes of operation, ODD subsets and the intended functionality of the system may lead to the addition of sensors, components or additional channels to compensate for the functional insufficiencies.

In general, modular architectures support SOTIF. This is evident for the challenges related to ODD and triggering conditions analysis, in combination with scenario-based validation. Acceptance criteria for validation efforts could also be defined per channel or in a more granular manner. Additionally, SOTIF issues are expected to require regular software updates (e.g., new traffic signs, extensions of the environmental model, safety case changes), which is facilitated by modular approaches.

2.8.1 SUPPORT TO ACCOMMODATE FUNCTIONAL INSUFFICIENCIES

- Does the architectural design sufficiently address the corresponding ODD and the vehicle's driving policy (e.g., OEDR, DDT, maneuvers, traffic rules)?
- Is the diversity of the architectural design elements sufficient to cover all the potential triggering conditions and output insufficiencies (e.g., the perception subsystem consists of diverse algorithms applying deep learning vs sensor fusion perception, avoidance of common cause false negatives when detecting/classifying objects)?
- To compensate for performance limitations of the environment perception sensors, the AD architectures include sets of diverse sensor modalities (e.g., vision, lidar, radar, localization).
- Does the architecture facilitate the validation of scenarios (i.e., scenario-based verification and validation from the perspective of SOTIF)?

If the conceptual system architecture scores badly on these questions, the system owner should be aware that the efforts to implement functional modifications addressing SOTIF-related risks is likely to be higher.

2.8.2 SUPPORT TO MANAGE OPERATIONAL CONDITIONS

- Does the architecture include components to monitor adequately the ODD in different operational conditions?
- Does the architecture ensure safe usage of the driving function in all operational conditions (e.g., control takeover, activation/deactivation, degraded mode, emergency mode)?
- Does the architecture support the monitoring and handling of potential misuses (i.e., ability to prevent or detect and mitigate reasonably foreseeable misuses)?
- Does the architecture support the data collection and monitoring of safety performance indicators during field operation (e.g., to improve the set of known scenarios, data possibly collected in real-time)?

If the conceptual system architecture scores badly on these questions, the system owner should be aware that mitigating risks associated with potential functional insufficiencies and/or triggering conditions, including those that are to be uncovered during operation, will likely be difficult to achieve. This can lead to the fact that a restriction of the intended functionality must be taken into consideration more than originally planned.

2.9 TABLE OF EVALUATION CRITERIA

Figure 10 illustrates the structure of the evaluation criteria. Each attribute is split into several evaluation criteria, which in turn have several associated key questions used during the evaluation. The full set of evaluation criteria is listed in Table 2, along with related system requirements (compare section 1.3), general constraints (compare section 1.5.1), and design principles (compare section 1.5.2).

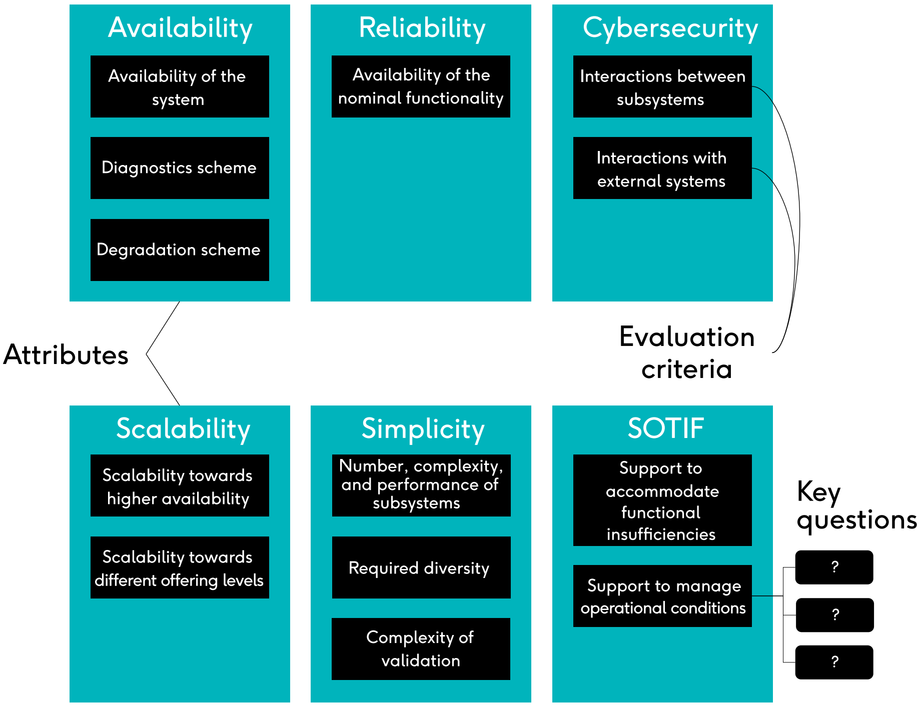

Availability
Availability of the system
Diagnostics scheme
Degradation scheme
Reliability
Availability of the nominal functionality
Cybersecurity
Interactions between subsystems
Interactions with external systems
Attributes
Evaluation criteria
Scalability
Scalability towards higher availability
Scalability towards different offering levels
Simplicity
Number, complexity, and performance of subsystems
Required diversity
Complexity of validation
SOTIF
Support to accommodate functional insufficiencies
Support to manage operational conditions
Key questions
?
?
?

Figure 10: Structure of relevant attributes, evaluation criteria, and key questions.

TABLE 2: SUMMARY OF THE EVALUATION CRITERIA.

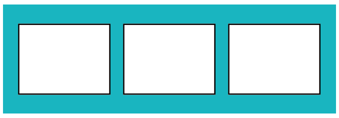
Attribute	Evaluation criterion	Example observations	Related
Availability	Availability of the system	- There are no obvious single-point faults in the architecture. - The architecture can deal with some multi-point faults.	S1: AD Intelligence output timeliness S2: AD Intelligence output availability S3: AD Intelligence output correctness S4: AD Intelligence output consistency G1: Design faults in large and complex monolithic systems G2: Single-event upsets in non-redundant HW G3: Testing and simulation of very high safety-related availability of large monolithic system G4: Specification of critical scenarios D1: Fault Containment Units D3: Diversity and redundancy for complex subsystems D7: Mitigation of common-cause hazards
	Diagnostics scheme	- Subsystems are aware of other subsystems' status and can adapt their behavior accordingly.	S6: AD Intelligence diagnostics
	Degradation scheme	- The architecture has a defined degradation scheme. - The failure of a single subsystem does not immediately lead to an emergency reaction (e.g., MRM).	S2: AD Intelligence output availability D6: Transient and permanent faults D7: Mitigation of common-cause hazards
Reliability	Availability of the nominal functionality	- Frequently occurring transient faults do not lead to an emergency reaction (e.g., MRM). - The arbitration decisions can be broken down into manageable logical statements.	G5: Frequent switching D6: Transient and permanent faults D7: Mitigation of common-cause hazards
Cybersecurity	Interactions between subsystems	- Subsystems only interact via well-defined interfaces.	D1: Fault Containment Units D5: Avoidance of emergent behavior
	Interactions with external systems	- Few subsystems need to communicate with external systems. - Few subsystems require frequent (e.g., OTA) updates. - Some subsystems can make use of a different, slower update mechanism (e.g., in workshop).	

Attribute	Evaluation criterion	Example observations	Related
Scalability	Scalability towards higher availability	- The architecture can be extended by adding more subsystems to achieve higher availability or integrity.	
	Scalability towards different offering levels	- Some of the subsystems are very similar to SAE L2 ADAS systems in functionality and could be carried over with minor modifications.	
Simplicity	Number, complexity, and performance of subsystems	- The number of subsystems is small. - The number of complex subsystems is small. - The number of subsystems with high computational performance requirements is small.	D1: Fault Containment Units D2: Simple and complex subsystems D5: Avoidance of emergent behavior D7: Mitigation of common-cause hazards
	Required diversity	- Diversity is required between few subsystems. - Diverse subsystems perform complementary functions (e.g., Doer/Checker). - Few complex subsystems require diversity.	S2: AD Intelligence output availability S3: AD Intelligence output correctness G6: Checks to determine plausibility of another subsystem D3: Diversity and redundancy for complex subsystems D7: Mitigation of common-cause hazards
	Complexity of validation	- The different subsystems are loosely coupled and cohesive enough to be independently validated. - The target failure rate of each subsystem requires a manageable testing effort.	G1: Design faults in large and complex monolithic systems G3: Testing and simulation of very high safety-related availability of large monolithic system D4: Provable correctness for simple subsystems
Safety of the intended functionality	Support to accommodate functional insufficiencies	- The diversity of the architectural design elements (e.g., independent sensor sets) decreases the risk of unhandled output insufficiencies.	G1: Design faults in large and complex monolithic systems G4: Specification of critical scenarios G7: Rate of safety incidents G8: Impact of system failure D3: Diversity and redundancy for complex subsystems D4: Provable correctness for simple subsystems D7: Mitigation of common-cause hazards
	Support to manage operational conditions	- The separation into independent channels with specific capabilities enables a high level of vehicle situational awareness.	S5: Perception malfunction detection G4: Specification of critical scenarios D7: Mitigation of common-cause hazards

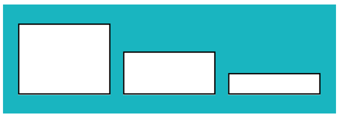
3 CANDIDATE ARCHITECTURES

In this section, we collect and describe different proposed conceptual system architectures that we will evaluate in section 4. We first describe the process we used for collecting such candidate architectures based on publicly available sources and the experience of the Working Group members (see section 3.1). Then, we identify generic underlying principles that are shared between multiple candidate architectures (see section 3.2). Finally, we describe the structure and behavior of each candidate architecture, where we cluster them into three major types:

1. MONOLITHIC ARCHITECTURES
(see section 3.3) present the status quo for SAE L2 ADAS and serve as the baseline for the evaluation.



2. SYMMETRIC ARCHITECTURES
(see section 3.4) rely on multiple channels providing the same or similar functions, often with some voting mechanism (see sections 3.2.1 and 3.2.2) determining which output to use.



3. ASYMMETRIC ARCHITECTURES
(see section 3.5) employ asymmetric decompositions to reduce the complexity of some subsystems, e.g., via Doer / Checker (see section 3.2.3) or Active / Hot Stand-By approaches (see section 3.2.4).

For each candidate architecture, relevant references and the considered variant are stated. If applicable, employed generic principles (see section 3.2) and design principles are listed. The structure of each conceptual system architecture is described via static modeling, while its behavior is described via dynamic modeling. The level of provided detail is intended to give an understanding of the architecture, while very specific details are shifted to the respective appendices.

3.1 COLLECTION PROCESS

In the context of AD, a variety of architectural concepts have been proposed by both commercial and academic players. Many of these are meant to address a specific topic, but do not present a complete architecture covering all abstraction levels. Proposals regarding conceptual system architectures can be somewhat tricky:

- Proposals from commercial players are sometimes incomplete, i.e., they only describe the concepts and components on a high level, but not how they work and interact in detail.
- Proposals from academic players are sometimes challenging from a commercial perspective, i.e., they neglect the high cost of implementing textbook redundancy and diversity.

As part of the activities of the Safety & Architecture Working Group, we have screened proposed architectural concepts for their applicability to the conceptual abstraction level.

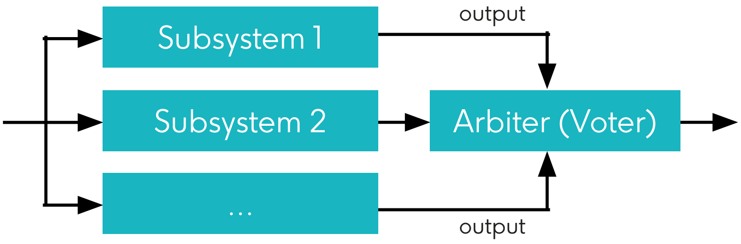
- When possible, we tried to extract generic underlying principles and cluster similar architectures.
- When necessary, we filled in missing details (from partial or very generic proposals) based on reasonable assumptions to be able to evaluate an architecture's behavior in certain scenarios and ultimately whether system requirements can be met.

3.2 OVERVIEW OF ARCHITECTURAL DESIGN PATTERNS

Most architectural design patterns in automotive systems come from other safety-critical areas, such as aviation. It is questionable whether these well-known approaches are sufficient for safe autonomous driving. In this context, we have found that the safety concepts for AD Intelligence have evolved in recent years. While ensuring reliability and availability through redundancy remains the most important strategy, Level 4 AD systems require different structural elements (i.e., subsystems[15]), organized in a hierarchical or distributed way for distinct safety responsibilities. We also recognize that the focus of current approaches is on functional safety, while some work is starting to include SOTIF ([30]).

This section provides an overview of the most common redundancy-based architectures. Although we focus on high-level safety concepts, it is noteworthy to consider that the architectural patterns are also applicable at lower levels of abstraction, depending on specific use cases. Furthermore, additional safety mechanisms, such as sensor fusion for the perception subsystem, are a well-established approach of AD systems to avoid the single failure and weaknesses of any individual sensor. The main challenge is to trade off complexity and performance while ensuring that the implemented safety mechanism covers relevant faults and functional insufficiencies. Other design patterns, such as watchdogs and sanity checks, are not explicitly mentioned as they are considered detailed implementations.

3.2.1 ARBITRATION AND VOTING



With two or more inputs coming from homogeneous (symmetric) or heterogeneous (asymmetric) subsystems, an element named “arbiter” acts as the decision maker that defines the output. The design of such an arbiter requires high safety integrity and low complexity.

There are different implementations of arbitration depending on the type of input data, the number of available input interfaces and the voting criteria. These aspects depend on the responsibility of the arbiter and are decisive for the performance of the safety measures. Subsystem independence (e.g., diversity in generating inputs) is fundamental to manage common cause failures.

Majority voting can be considered a special case of arbitration.

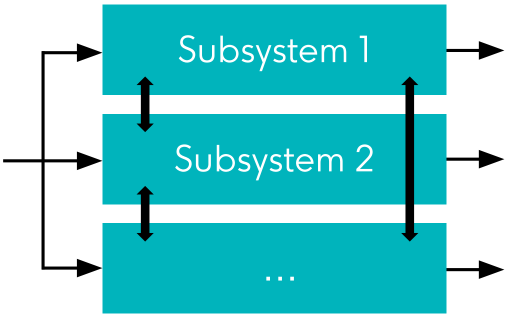
Applicability:

- The original approach, i.e., binary inputs, odd number of subsystems, and majority voting, can be considered the simplest case. Such a simple arbiter could be used, for example, to determine whether to enable a safety channel or path.

[15] A subsystem can refer to an element, a set of elements, or a channel. The channel we are considering here refers to the subsystem composed of a sensor set, a perception element, and a planning element (i.e., the first two stages of the so-called "sense, plan, act" model of autonomous driving).

- For more complex cases, such as continuous-valued signals (e.g., acceleration) or heterogeneous components, a more sophisticated implementation is required. This problem is comparable to inexact agreement.
- Some examples of arbitration criteria are plausibility checks, acceptance tests, risk estimation or scenario-based prioritization.
- An aspect to consider is the potentially high development costs for the independent subsystems.
- To ensure fail-operational arbitration, multiple arbiters may be considered.

3.2.2 AGREEMENT

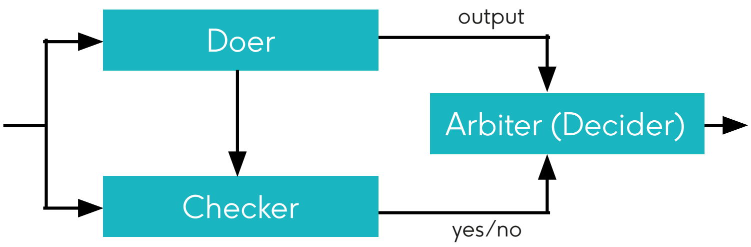


Redundant subsystems called "participants" interact to reach a decision, without an arbitration component. The agreement pattern is based on a closed loop approach and may consist of multiple rounds of information exchange between all (available) participants.

Applicability:

- Like voting, agreement is applicable for redundant subsystems.
- Agreement mechanisms are also used for the detection and isolation of asymmetric faults.
- Like voting, there are challenges related to the implementation of agreement, especially those related to the type of input data (e.g., inexact agreement). Solutions for this can be the use of convergence algorithms, confidence rating, approximate outputs considering a given precision and allowed system accuracy.
- Agreement algorithms might not be viable when there are numerous acceptable decision candidates that might differ significantly due to the use of nondeterministic algorithms.

3.2.3 DOER/CHECKER (OR CONTROL/MONITOR)

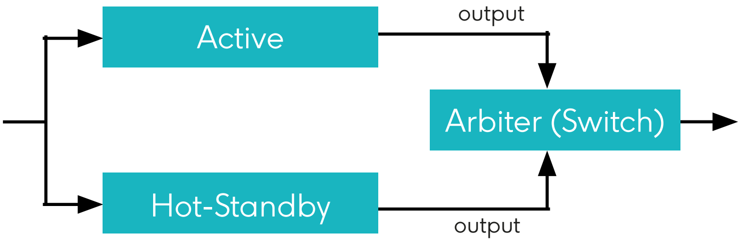


One subsystem, the "doer", performs a function while another, the "checker", monitors it. The checker requires higher safety integrity and lower complexity than the doer. The doer implements the nominal capabilities of the system.

There are different implementations of Doer/Checker depending on the comparison strategy and how monitoring is performed. Additional self-checks and cross-checks may be required to prevent single-point failure in the system. Subsystem independence (e.g., separate hardware) is required for high reliability and availability.
Applicability:

- This approach is useful if an appropriate simpler monitoring component is feasible.
- Some factors such as time lags and computational accuracy might affect the performance of the monitoring function.
- The original doer/checker pattern, as shown in the figure, is only applicable to fail-safe systems.

3.2.4 ACTIVE AND HOT STAND-BY (OR DUPLEX PATTERN)



Two homogeneous or heterogeneous subsystems operate continuously in parallel while only one of them is active at any given time. A fault detection mechanism acts as a switch between the subsystems. The fault detection mechanism requires redundancy (e.g., cross-checking) to avoid single-point failures.

The component acting as comparator and fault detector shall be designed carefully to ensure high fault coverage. Some methods used to identify the faulty subsystem are acceptance test and hardware testing (for details, see [31]).

Related alternatives are:

- Warm Stand-By: the reserve subsystem runs in idle state, and
- Cold Stand-By: the reserve subsystem is normally off.

Applicability:

- This approach is suitable for functionalities with strict time constraints.
- The decision between hot, warm, or cold redundancy depends on the required safety level, response time, and power consumption.

3.3 MONOLITHIC ARCHITECTURES

3.3.1 SINGLE-CHANNEL ARCHITECTURE

In 2015, Audi gave some insights into their then-next generation HW and SW platform [32], intended to cover more complex ADAS use cases and a novel SAE L3 Traffic Jam Pilot AD use case[16]. While the described architecture was designed with a different use case in mind than outlined in Section 1.1, it can remain relevant to make the differences from other architectures more apparent.

[16] This use case was intended to support hands-off/feet-off/eyes-off driving in traffic jam scenarios (up to 60 km/h) on highways. In case of a failure of the system, the driver was supposed to take over within ~10 seconds.

3.3.1.1 UNDERLYING CONCEPTS AND DESIGN PRINCIPLES

Historically, most ADAS were a collection of dedicated ECUs and sensors for every individual function or use case, e.g., Adaptive Cruise Control (ACC), Light Assistant, Parking Assistant, Top View, etc. As more such systems were added to vehicles, cost and complexity scaled poorly and the performance of the provided functions remained limited, since fusing the information from the distributed sensors proved to be difficult.

Starting around 2015, the leading players in the automotive industry started working on more centralized platforms that decoupled specific sensors from specific ADAS functions by introducing a centralized sensor data fusion layer in-between (see Figure 11), with the intention to gain several benefits:

- Multiple control units could be integrated into one unit and their HW resources shared.
- A more modular architecture (due to decoupling of SW from HW) could allow updating functions or deploying additional ones over the lifetime of the vehicle.
- The central environment model could reduce redundancies and make consistent information available to many applications.
- Improved recognition of the vehicle's surroundings and a more detailed environment model gained through multi-sensor data fusion could support more complex SAE L2 (ADAS) use cases and even novel SAE L3 use cases.

At the time, developing an integrated HW and SW platform capable of hosting a large number of applications with widely varying computational needs (e.g., FPGA or GPU) was challenging. Table 3 shows an overview of the different HW components in the Audi zFAS system and the hosted functions.

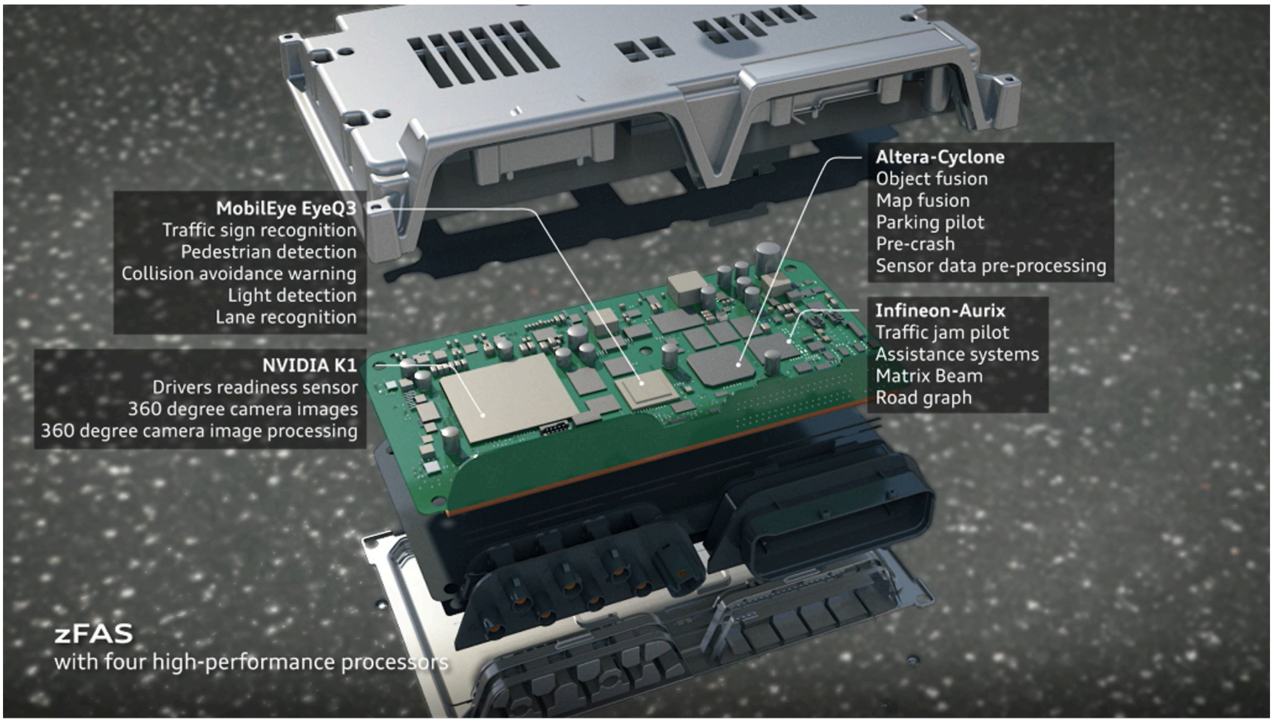


Figure 11: Centralized sensor data fusion layer in the Audi zFAS platform [33].

TABLE 3: HW COMPONENTS IN THE AUDI ZFAS PLATFORM [33].

Automotive-qualified embedded microcontroller	• Various functions (up to ASIL D) • Interface to the rest of the vehicle
FPGA	• Sensor fusion • Sensor processing
Image processing SoC	• Image processing • Computer vision • Driver monitoring
Front camera image processing SoC	• Computer vision • Emergency braking

3.3.1.2 STRUCTURAL DESCRIPTION

The proposed architecture is monolithic, i.e., it only consists of a single FCU with interfaces identical to the external interfaces of the AD Intelligence.

3.3.1.3 BEHAVIORAL DESCRIPTION

Due to its monolithic nature, the behavior of the system is straightforward:

- It processes received sensor data into a consistent environment model.
- It then periodically generates trajectories and corresponding actuator setpoints.
- These setpoints are then sent to the Actuator System.
- If an internal fault is detected, the system remains silent.

Due to the underlying use case of a traffic jam pilot (low speed and a restrained environment), the safety requirements are noticeably different from most AD use cases with respect to integrity (i.e., complex functionality does not need to reach the highest ASIL) and availability (i.e., the system does not need to provide complex fallback functionality in case of a fault).

3.3.1.4 INSIGHTS INTO TESLA'S FSD SYSTEM

Tesla's "Full Self Driving" (FSD) seems to be a more recent implementation of a single-channel architecture. Despite the suggestive label and marketing as a highly autonomous system, FSD and its precursors "Autopilot" and "Enhanced Autopilot" are formally sold as SAE L2 systems, where the driver needs to supervise and be ready to take control of the vehicle at any time. While the Autopilot function is meant as a (semi-)autonomous highway driving system, the FSD system aims to include urban roads. Technical information is made available by Tesla in the course of yearly "AI Days", e.g., [34] and shows in considerable detail that the system is (at the time of writing) purely camera-based and composed of multiple complex machine learning modules that are specialized in various elements of the world model (objects, lanes, ...). A single, common planning module on top is responsible for computing the actual vehicle trajectory and controlling the vehicle motion. There is no mention of any functionally redundant blocks like supervision, or of fault-tolerance mechanisms like comparisons or voting on the SW architecture level.

With the introduction of the so-called "HW3" generation of the central driving computer, Tesla deployed the core SOC twice, in a parallel redundant fashion, stating that if either one were to fail, the redundant component would take over. It remains unclear, however, if that redundancy is exploited on a functional and logical architecture level—quite likely the same single-channel FSD stack is essentially intended to be deployed twice, and the redundant SOC is meant to just address random faults in the underlying electronic components, like core SOC failures and power supply outages, but not functional deficiencies or systematic implementation faults. Therefore, the architecture may still be considered as monolithic single channel logically.

In unconfirmed information, [35] states that ultimately redundancy has been dismissed in HW3 and the second SOC used to increase computational performance instead. With a new "HW4" generation, redundancy of SOCs would be introduced again and augmented by radars (which were dismissed as unnecessary earlier). This might open up the potential of true SAE L3 and higher operation to Tesla and could also indicate a step away from the apparent single-channel architecture.

3.4 SYMMETRIC ARCHITECTURES

3.4.1 MAJORITY VOTING ARCHITECTURE

This section follows the specific variant of majority voting called "triple modular redundancy (TMR)" with a particular focus on the redundancy aspects of the architecture. As described in [36], this type of redundancy is commonly used in very high reliability systems such as those used in aerospace. To the authors' knowledge, a strict application of this architecture in the AD domain has not yet been officially published. Still, we include it in the report, as the voting paradigm is an obvious and tempting approach for AD systems, and its properties therefore deserve a closer look.

3.4.1.1 UNDERLYING CONCEPTS AND DESIGN PRINCIPLES

Triple modular redundancy is a specific implementation of "N-Modular Redundancy" where you have three identical channels that produce results that are fed to a "voter". The voter is responsible for looking at the results from the three channels and deciding which result is likely to be correct.

- The voter operates based on the assumption that common mode failures are much less likely than single-event errors. This implies that the majority is correct. Therefore, if the voter observes two identical results and one dissimilar result, it will assume that the dissimilar result came about through a failure and the two identical results are in fact correct.
- In the strictest sense, only identical results can form such a majority. This can be relaxed to some extent to "sufficiently similar" results via inexact voting approaches.

3.4.1.2 STRUCTURAL DESCRIPTION

The proposed conceptual architecture consists of four FCUs:

- Three independent channels computing results[17].
- The independent voter responsible for deciding which is the correct result[18].

The interfaces to the three channels are identical, and each output of the channel is fed to the voter. The results of the voter are then fed to the actuating systems of the vehicle.

[17] For exact voting, these channels will most likely need to be implemented in an identical way unless the provided functionality is very simple and straightforward. Even then, replica indeterminism may cause issues. For inexact voting, some degree of diversity (to prevent common cause faults) may be allowed.

[18] Some advanced versions of TMR (e.g., triple-triple) include multiple voters, but we do not consider this here.

3.4.1.3 BEHAVIORAL DESCRIPTION

In the proposed architecture, each channel would contain all functions required to implement the entire AD system. This would include taking the sensors of the vehicle as input, generating an environment model, trajectories, and actuator setpoints, checking whether the vehicle is operating inside the ODD, etc. Like the linear/monolithic architecture, each channel would:

- Process received sensor data into a consistent environment model.
- Periodically generate trajectories and corresponding actuator setpoints.
- Send these setpoints to the voter (or ultimately to the Actuator System).
- Remain silent if an internal fault within the channel is detected.

The behavior of the voter can be summarized in the below table of states. The table has been distilled into the minimum number of unique states and does not include every permutation.

Channel A result	Channel B result	Channel C result	Voter decision
Result A	Result A	Result A	Result A
Result A	Result A	Result B	Result A
Result A	Result B	Result C	No decision – fault

From this table we can observe the simplicity in the voter's design. There is a minimum number of possible combinations the voter must consider. We also see some of the majority voting architecture's major flaws. The design of majority voting relies heavily on the assumption that the failure modes are unique and that common mode failures are unlikely. It is also possible for the voter to get into a state where no decision can be made if each of the identical channels produces a different result. Concretely, while simple problems may have a single "best/correct" solution, more complicated or even complex problems may have multiple "good" solutions, which can differ fundamentally (e.g., evading left or right). This may be more pronounced in the automotive domain (busy road) than the aerospace domain (empty sky). See quote from [26] in the next paragraph. This would need to be treated as a fault scenario and a predetermined recovery action would take place.

Quote from [26]: Provably correct Decision System: Whenever two independent redundant subsystems are involved in a decision in a complex environment there is the possibility of two different correct outcomes. The introduction of a third subsystem will only mask a single fault if the involved systems are replica determinate [37].

3.4.2 CROSS-CHECKING PAIR ARCHITECTURE

At the time of creation of this document, the experts of the working group were aware of a symmetric approach proposed by one company specialized in automated driving [38]. The group wanted to investigate this approach further, but did not find public information. The group decided to keep this short section so that the investigation could potentially be continued at a later point in time.

3.5 ASYMMETRIC ARCHITECTURES

3.5.1 CHANNEL-WISE DOER / CHECKER / FALLBACK (DCF) ARCHITECTURE

This section discusses the architecture proposed by Kopetz [26], which can be considered a specific combination of the Doer / Checker and Active / Hot Stand-By (see section 3.2) approaches for decomposition with respect to integrity and availability, respectively. In this case, the decomposition is done for entire processing channels. Involved design principles and their respective intentions are:

- Minimizing interactions between the different subsystems (in this case entire processing channels) is intended to reduce complexity and to prevent emergent behavior (see also: Avoidance of emergent behavior).
- Employing a Time-Triggered Architecture (TTA) is intended to reduce ambiguity between late vs. missing messages and to prevent the formation of mutually inconsistent time domains.

3.5.1.1 UNDERLYING CONCEPTS AND DESIGN PRINCIPLES

The channel-wise Doer / Checker / Fallback architecture is based on a combination of the Doer / Checker approach (for decomposition with respect to integrity) and the Active / Hot Stand-By approach (for decomposition with respect to availability). These are outlined in sections 3.2.3 and 3.2.4, respectively.

The conceptual architecture of the variant proposed by Kopetz is based on a Time-Triggered Architecture (TTA), i.e., scheduled task execution and communication across all subsystems. This simplifies the Doer / Checker and Active / Hot Stand-By decompositions:

- Missing and delayed messages between channels are treated the same way.
- Latencies due to communication between channels and the redundancy management can be bounded and reduced.

As described in [26], the proposed conceptual architecture also requires sufficient independence between channels to prevent common cause faults[19]. This may necessitate some diversity in HW and SW implementations (to be further discussed in Section 5).

3.5.1.2 STRUCTURAL DESCRIPTION

The proposed conceptual system architecture consists of four subsystems (see Figure 12), namely:

1. The Computer-Controlled Driving Subsystem (CCDSS) controls the vehicle under nominal conditions, i.e., it is similar to some SAE L2 systems[26][20]. It periodically produces trajectories (e.g., timed waypoints) and actuator setpoints (e.g., desired acceleration / deceleration and curvature values for steering, powertrain, and brakes) and transmits these to the MSS and the FTDSS.
2. The Monitoring Subsystem (MSS) detects unsafe trajectories produced by the CCDSS, whether nominal conditions prevail, and whether the CEHSS is still alive.
3. The Critical Event-Handling Subsystem (CEHSS) controls the vehicle under off-nominal conditions. It only aims to bring the vehicle into a safe state, i.e., to execute an MRM, but must be able to do that even after an ODD exit. It periodically produces trajectories and actuator setpoints and transmits these to the FTDSS.
4. The Fault-Tolerant Decision Subsystem (FTDSS) decides which setpoints are forwarded to the Actuator System. It consists of two identical instances to achieve fault tolerance.

[19] For complex subsystems (i.e., the CCDSS, MSS, and CEHSS described in the subsequent section) this may necessitate diverse SW implementations. For sufficiently simple subsystems (i.e., the FTDSS) with fully verifiable SW, no diversity is necessary.

[20] The other subsystems effectively take over the tasks performed by a human driver in an SAE L2 system and are collectively called Safety Assurance Subsystem in [26].

A more detailed structural description of the subsystems is given in Appendix B: Detailed description of the channel-wise DCF architecture.

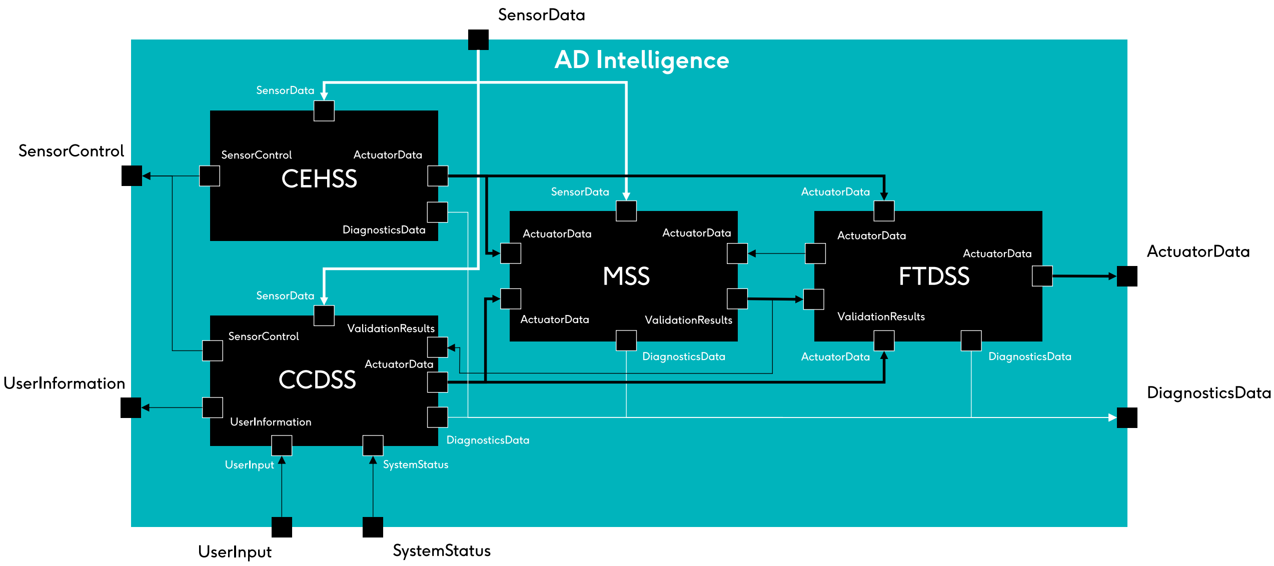


Figure 12: Block diagram of the channel-wise Doer/Checker/Fallback conceptual architecture. Main information flow is indicated by gray arrows, supplementary information flow by green arrows.

3.5.1.3 BEHAVIORAL DESCRIPTION

Table 4 describes the behavior of each of the subsystems in more detail. A corresponding activity diagram for the AD Intelligence is shown in Figure 13 (showing the time-driven tasks with the common main cycle time of the AD Intelligence in the fault-free case). Consensus in the Actuator System (compare: AD Intelligence output consistency) is straightforward: actuators follow the received setpoints with the higher priority, i.e., they prefer CCDSS setpoints over CEHSS setpoints.

TABLE 4: BEHAVIOR OF THE SUBSYSTEMS OF THE CHANNEL-WISE DOER/CHECKER/FALLBACK ARCHITECTURE.

Subsystem	Behavior
CCDSS	• Produces trajectories and actuator setpoints (ActuatorData) for nominal conditions. • Sends ActuatorData to MSS and FTDSS. • Goes into degraded mode (MRM only) if system state demands it (e.g., if CEHSS is faulty).
CEHSS	• Produces trajectories and actuator setpoints (ActuatorData) for off-nominal conditions (MRM only). • Sends ActuatorData to MSS and FTDSS.
MSS	• Validates trajectories from CCDSS (safe, same as received by FTDSS). • Validates trajectories from CEHSS (safe, same as received by FTDSS). • Sends ValidationResult to FTDSS and MSS.
FTDSS	• Forwards CCDSS and CEHSS ActuatorData back to MSS. • Selects CCDSS or CEHSS ActuatorData depending on ValidationResult. • Sends ActuatorData to Actuator System.

For all four subsystems of the AD Intelligence, task execution and communication are based on a time-triggered schedule. If a message is not received in the planned time slot (and has the correct iteration counter), it counts the same as if it hadn't been received at all or if it had been received in a corrupted state (e.g., with an invalid checksum).

Transient faults in one of the complex subsystems can occur quite frequently, so the FTDSS is constructed to allow rapid back-and-forth switching if necessary[21]. However, unduly frequent transient faults are considered indicative of an underlying problem and cause the CCDSS to go into a degraded mode (see also G5: Frequent switching).

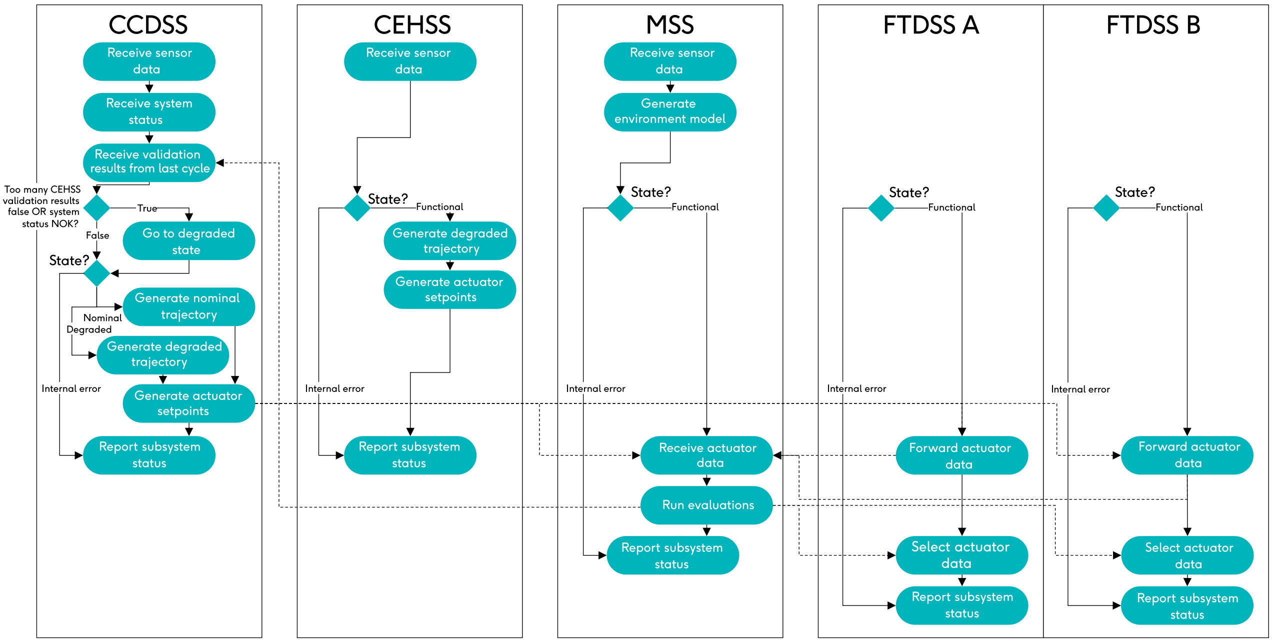


Figure 13: Sequence diagram of the channel-wise Doer/Checker/Fallback architecture in the fault-free case.

A more detailed behavioral description of the subsystems is given in Appendix B: Detailed description of the channel-wise DCF architecture.

3.5.1.4 RELATED EXAMPLES: BMW SCALABLE AV PLATFORM ARCHITECTURE

In 2020, BMW unveiled some details on its scalable AV platform architecture [39] [40], intended for SAE L3 AD features such as a Highway Pilot system similar to the one outlined in section 1.1 - Reference AD use case. The published materials include an overview of the planned HW architectures for different offering levels for the then-planned SOP 2021 (see Figure 14), as well as a conceptual system architecture for the SAE L3 system, dubbed hPAD (see Figure 15).

Based on the structural description in these materials, the conceptual architecture proposed by BMW shares similarities with the channel-wise Doer/Checker/Fallback architecture. Going by the depicted subsystems and high-level functional blocks, the "MAIN" channel appears similar to the "Doer", the "SAFE" channel similar to the "Checker", and the "SAFE fail-degraded" channel similar to the "Fallback". However, the "SAFE" channel also seems to produce trajectories and some cross-checking between "MAIN" and "SAFE" appears to occur, as well.

[21] If transient faults (e.g., erroneous object detections) are of very short duration, e.g., a low single-digit number of frames/iterations, the FTDSS will switch to the CEHSS for just those few frames and back to the CCDSS as soon as it recovers. Though this may occur relatively often, it is not necessarily noticeable to the passengers.

As the published behavioral description is incomplete, we decided not to include this candidate conceptual system architecture in our evaluation. This avoids proceeding based on speculation and assumption.

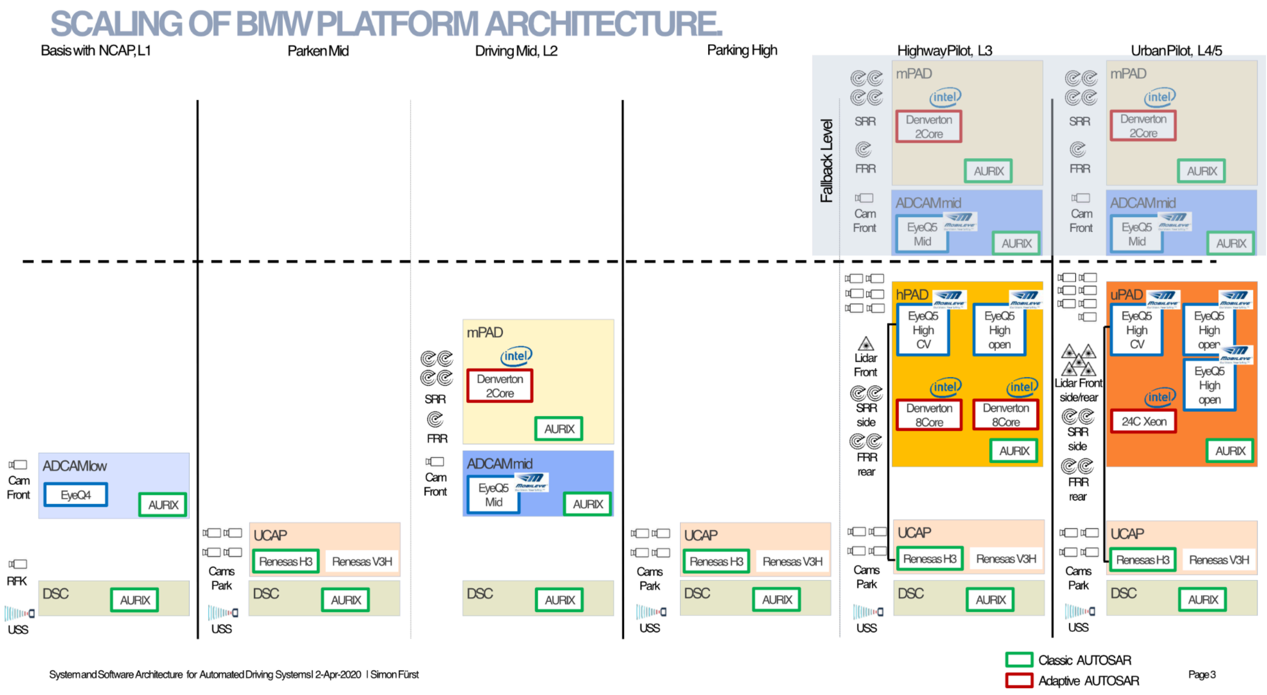


Figure 14: Planned HW architectures for different offering levels as proposed by BMW [40].

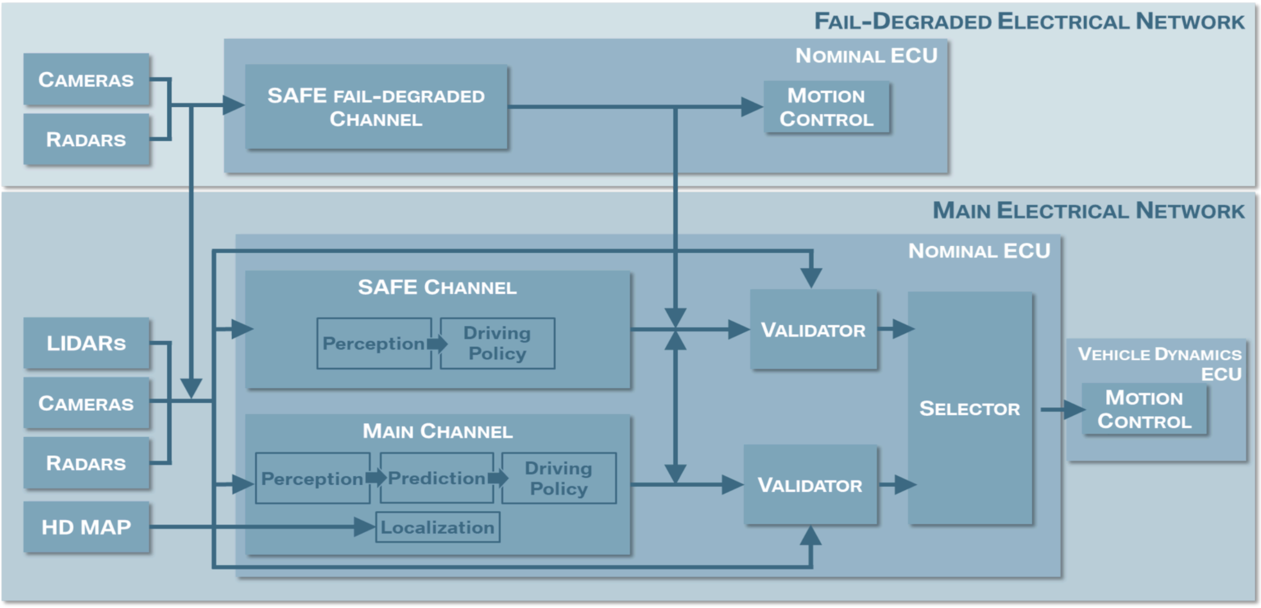


Figure 15: Conceptual architecture proposed by BMW for an SAE L3 system [39].

3.5.2 LAYER-WISE DOER / CHECKER / FALLBACK ARCHITECTURE

In [41], a multi-channel approach combined with the doer/checker pattern is presented in a patent as a safety architecture for AD. This section summarizes the most relevant aspects of this invention.

3.5.2.1 UNDERLYING CONCEPTS AND DESIGN PRINCIPLES

Doer/ Checker pairs are used as the main architectural pattern. The primary Doer/Checker pair acts during normal mode, and a secondary Doer/Checker pair provides a degraded mode of operation in case the primary pair fails. Additionally, the arbiter “Priority Selector” determines the output to be sent to the actuators (see Figure 16).

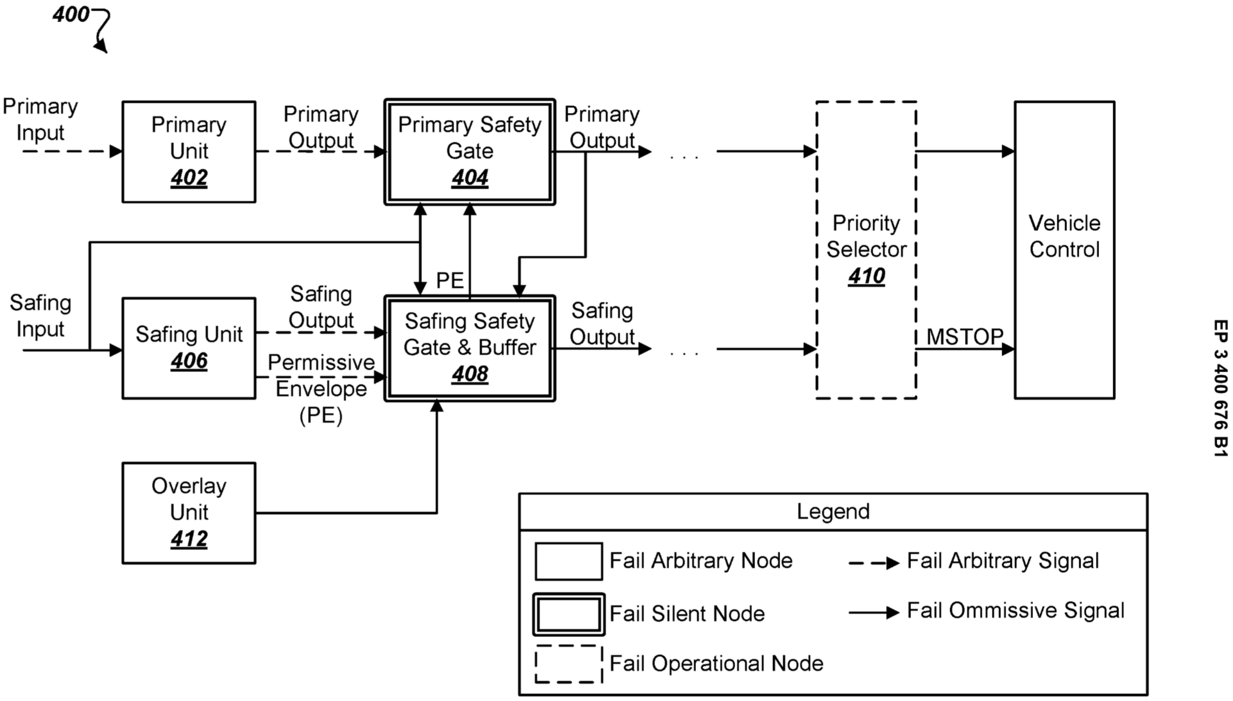


FIG. 4

Figure 16: Generalization of the Layer-wise Doer/Checker/Fallback architecture (figure taken from [41])

3.5.2.2 STRUCTURAL DESCRIPTION

The pattern shown in Figure 16 can be repeated for different layers or stages, as shown in Figure 17 for planning and trajectory execution.

The Safing channel (i.e., the secondary Doer/Checker pair) is more elaborate than the Primary one. It includes a “Permissive Envelope” signal, which indicates a reference used to validate the Primary output. For example, the safing trajectory executor unit may generate a permissive envelope that specifies a maximum acceleration rate. The Safing Safety Gate not only produces the Permissive Envelope but also evaluates whether it is appropriate.

The Primary and Safing Units used to generate outputs in both channels (i.e., the doers) may have low integrity levels and may each fail arbitrarily. The two “safety gate” components (i.e., the checkers) are responsible for checking the outputs of the Primary and Safing Units. They are high-integrity components, but they fail silently if these outputs are unsafe.

The Priority Selector must continue to operate in the presence of failures to deliver either the Primary or Safing output. The Priority Selector may fail silently so long as that failure triggers an emergency stop. This component is simpler than the safety gates, so that a great deal of effort can be spent on its verification to achieve the required high level of integrity.

The optional “overlay” channel can be used as an additional fallback mechanism for testing purposes.

3.5.2.3 BEHAVIORAL DESCRIPTION

Various implementation alternatives are mentioned in the patent (e.g., time-triggered vs. event-triggered architecture). Depending on the choice, the system will behave differently. But in principle, the following applies:

- If an unhandled failure occurs in either the Primary or the Safing Unit, the architecture remains operational and continues to meet the safety goals by means of the correspon-ding Safety Gates.
- If both the Primary Unit and the Safing Unit fail, the system remains safe by recovering actions performed by the downstream stages (e.g., executing an emergency stop).
- In a cyclic (i.e., periodic, deterministic) way, the checker of the fallback channel (i.e., the Safing Safety Gate) buffers a safe trajectory and validates it against the current operational situation. If the Safing Unit for the planning stage malfunctions, the last safe trajectory is used before the vehicle comes to a stop.
- If the Safing Safety Gate for the planning stage fails, both Primary and Safing outputs are inhibited. The downstream stage gets no inputs, and thus sends no outputs, which causes the execution of the last safe trajectory.
- If the Safing Safety Gate for the trajectory execution stage fails, both Primary and Safing output are inhibited, which causes the Priority Selector to execute an emergency stop

.

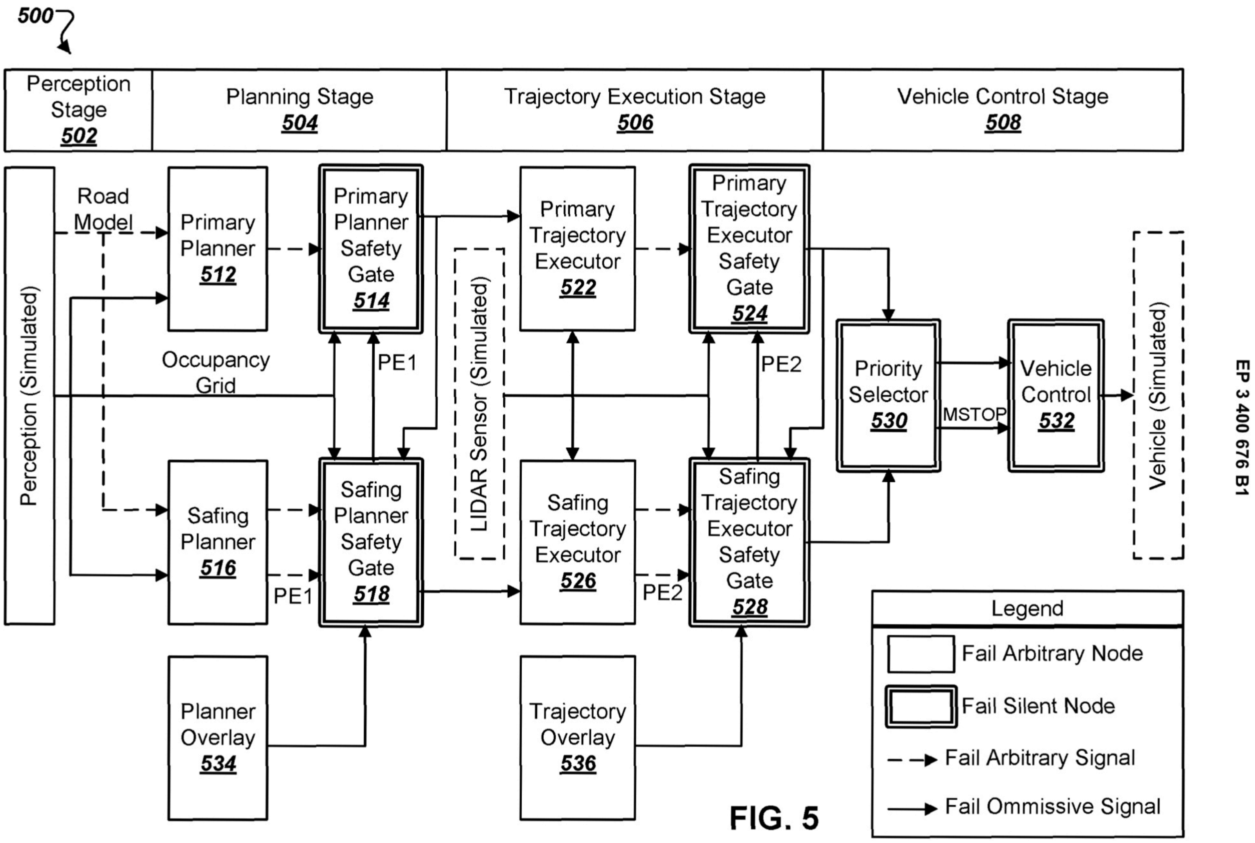


FIG. 5

Figure 17: Architecture of a prototype applying Layer-wise Doer/Checker/Fallback (figure taken from [41])

3.5.3 DISTRIBUTED SAFETY MECHANISM ARCHITECTURE

This section discusses the Distributed Safety Mechanism (DSM) architecture proposed in [42] [43][22], which can be considered a distributed variant of the Doer/Checker/Fallback approach.

3.5.3.1 UNDERLYING CONCEPTS AND DESIGN PRINCIPLES

The DSM architecture has three channels:

- The nominal channel, consisting of the function (FUN) controlled by one or more sensor and function safety monitors (SFM).
- The emergency channel, which is controlled by one or more controller safety mechanisms (CSM).
- The safety channel, which is controlled by the vehicle safety mechanism (VSM).

Compared with the Channel-wise Doer/Checker/Fallback (DCF) architecture variant proposed by Kopetz (see 3.5.1), there is a rough correspondence between:

- FUN and CCDSS,
- SFM and MSS,
- VSM and CEHSS.

However, the redundancy management itself is different: whereas in the Kopetz variant there is a smart switch (FTDSS), in the DSM all subsystems apart from the controlling subsystem are silenced.

Another relevant difference to the Kopetz variant is that the CSM supports the additional emergency stop mode as a further fallback layer, in which no sensors are used, i.e., a braking within the latest available trajectory evaluated as safe takes place. Alternatively, this level can also be implemented in another way, e.g., on the actuator-ECU side.

The DSM is in fact able to deal with some combinations of multiple faulty components due to a decentralized monitoring and response. No redundant components (like FTDSS) are required, but a strong safety responsibility is assigned to the system level.

In the formally verified model presented in [43], it is assumed that a distributed protocol allows the communication of the actuator setpoints of one and only one of the channels, depending on the current system state. Five systems states, each of them representing an operating mode (i.e., nominal, detour, comfort setup, safe stop, and emergency stop) were considered for the DSM architecture.

Figure 18 shows the degradation concept defined for the DSM behavior that covers the different system states and fault-triggered transitions. For the nominal mode, the function (FUN) controls the vehicle with high system availability and resilience (i.e., fault tolerance). The DSM aims to increase resilience to multi-point faults via a multi-layered monitoring concept.

[22] An open-access version of this paper is available at https://arxiv.org/ftp/arxiv/papers/2011/2011.00892.pdf

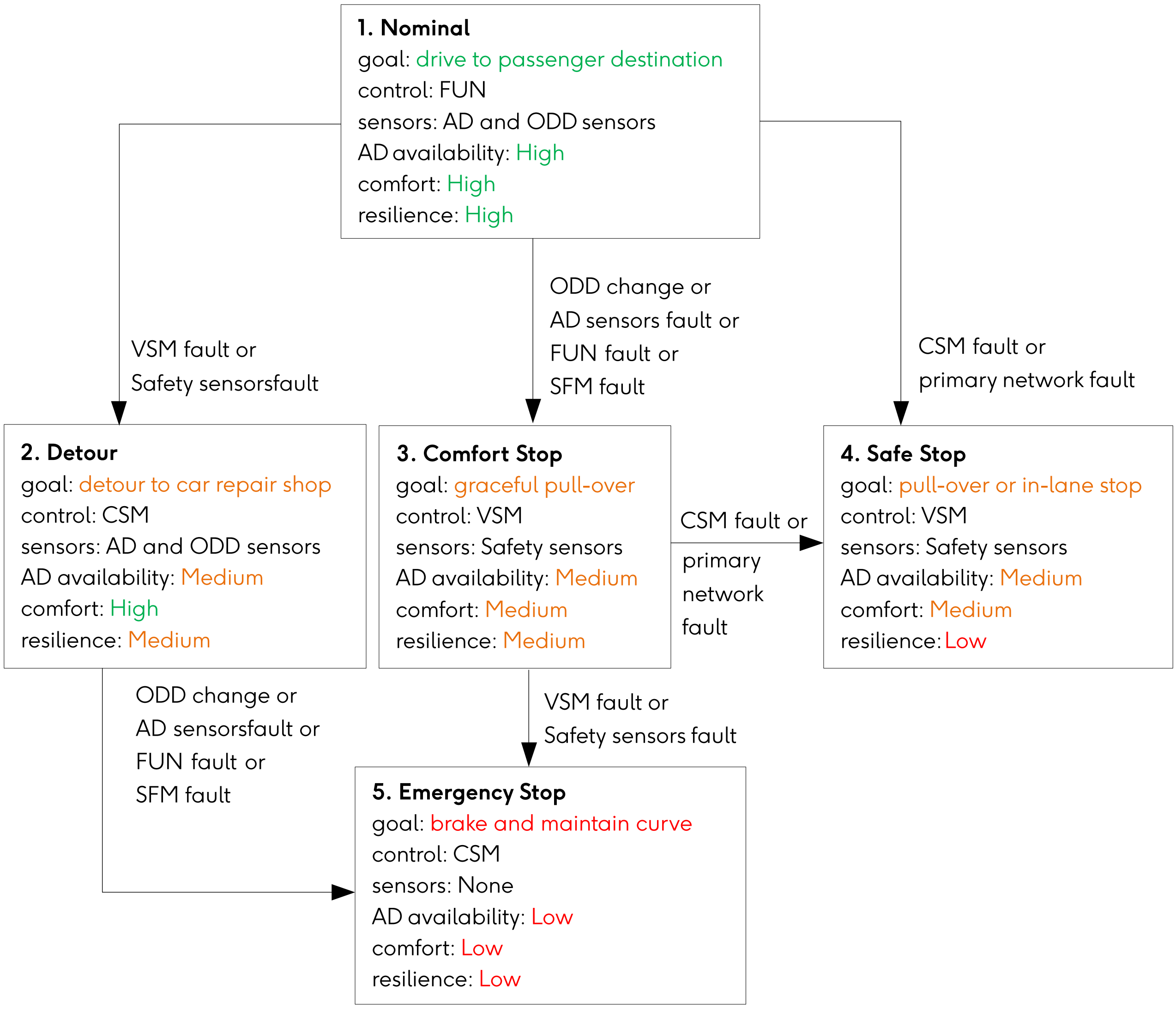


Figure 18: Degradation concept for the DSM presented in [43]

Regarding the redundancy and diversity aspect, the DSM architecture relies on redundant communication networks (primary and secondary ones). Additionally, in contrast to the nominal channel, the safety channel implements the AD function using simpler algorithms and data coming from fewer safety sensors.

A key design principle of the DSM is the highly scalable architecture due to the separation of concerns and the distributed monitoring approach. Inspired by the E-GAS layered monitoring concept [44], it includes safety monitoring components at function, controller, and vehicle levels. With this property, the nominal channel can be easily extended to add more functionality or performance features, without affecting considerably the rest of the system.

High reliability and determinism are required for the system-level distributed communication protocol to support the safety mechanisms.

3.5.3.2 STRUCTURAL DESCRIPTION

The DSM architecture consists of 3 safety monitor types. There is a clear separation of safety concerns defined by specific responsibilities and associated fault modes for each of these monitors. Table 5 summarizes the most relevant characteristics and some implementation details given in the paper.

Note that the CSM and the VSM are cross-checking each other (i.e., challenge-response protocol), and therefore additional safety mechanisms at the communication protocol level are required.

TABLE 5: SAFETY MONITORS OF THE DSM ARCHITECTURE

Monitor	Responsibility	Safety Integrity	Characteristics
Sensor and Function Monitor (SFM)	Safety of AD functions (e.g., SOTIF)	Low (e.g., ASIL B)	• Monitors the status of the function FUN. • Can act as an ODD checker. • Runs on the performance cores of the function controllers
Controller Safety Mechanism (CSM)	Safety of the function controller, hardware, and platform software (hypervisor, OS, firmware)	Medium (e.g., ASIL C)	• Monitors all the function controllers and the VSM layer. • It has access to the primary network channel and can send control commands to the vehicle actuators (detour or emergency stop). • Can compare channels' outputs to identify inconsistencies between the nominal and safety channels. • Runs on the safety cores of the function controllers.
Vehicle Safety Mechanism (VSM)	Vehicle safety, including monitoring of data and power networks' integrity	High (e.g., ASIL D)	• Monitors the CSM, the safety sensor data, and the two communication networks. • Can maneuver the vehicle via the secondary network, using the safety sensor data (comfort or safe stop). • Runs on a separate safety controller. • Fails silently.

Figure 19 shows a simplification of the structure and the most relevant data flows of the DSM architecture. As can be seen, the architecture includes a dedicated set of sensors for the safety channel (VSM controlling the vehicle) which is required for a safety maneuver (i.e., comfort or safe stop operating modes).

There are interfaces playing a key role for detecting and controlling faulty component behavior. For example, in case of a fault in the function monitor (SFM), the CSM triggers a turn-off of the function and reports this to the VSM to enter the comfort stop mode. Additionally, for the mode change from nominal to detour (see Figure 19), if the VSM fails, the CSM takes over the control of the vehicle.

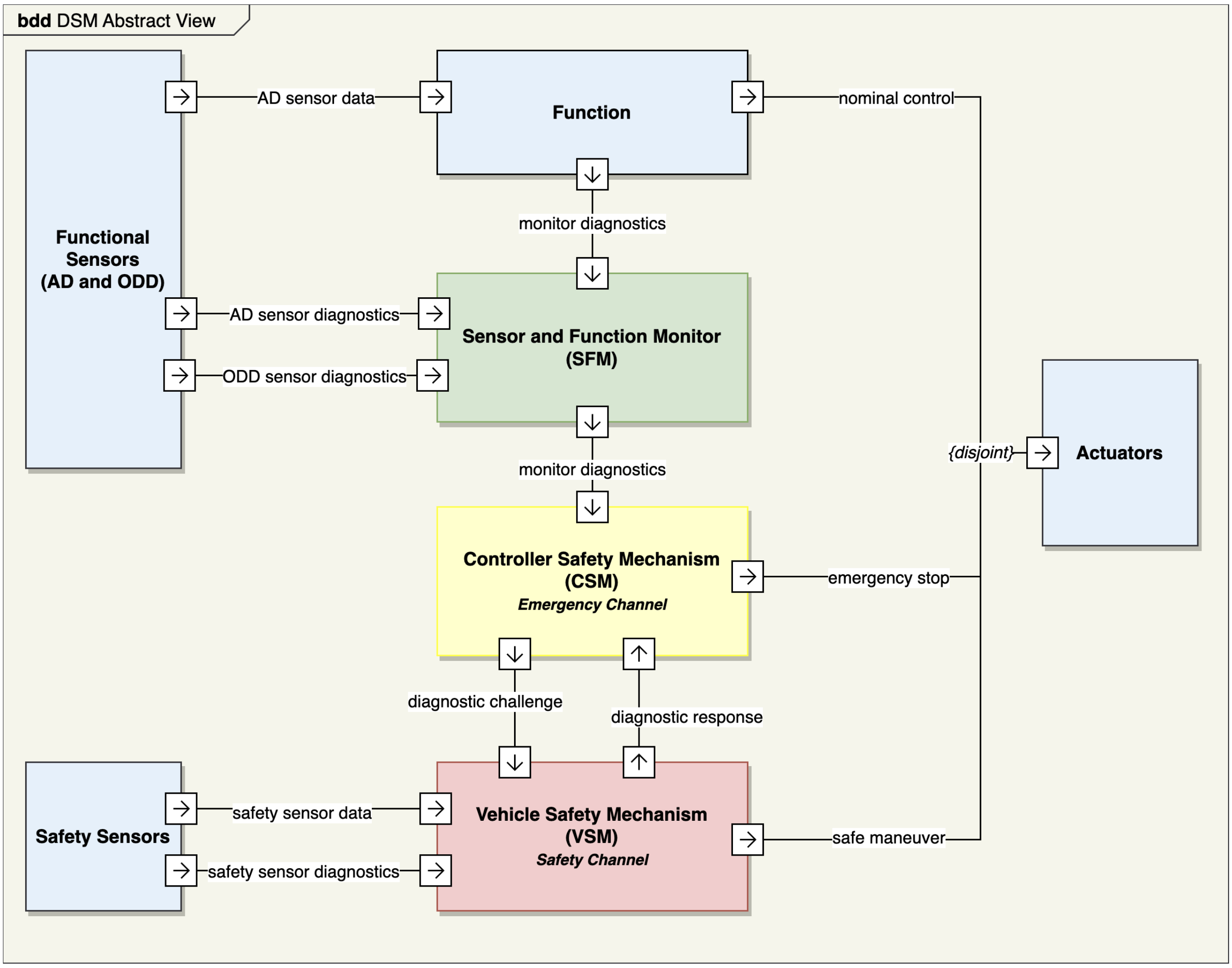


Figure 19: High-level architectural view of the DSM. The Actuators receive one of three possible signals, depending on the current system state.

3.5.3.3 BEHAVIORAL DESCRIPTION

Besides the degradation concept mentioned above, exemplary sequence diagrams presented in [43] describe the expected behavior of the DSM architecture (see Figure 20 and Figure 21).

Note that the safety monitors are performing different monitoring tasks at the same time. During fault-free nominal mode, the VSM runs simultaneously in the redundant safety channel processing the safety sensors without sending any control command to the actuators. The FUN component is controlling the vehicle. The CSM and the VSM monitors cross-check each other through the primary network using a challenge-response protocol.

Once a fault has been detected, the system activates a degraded operation mode. In Figure 21, an extreme case is illustrated. First, a sensor of the nominal channel and the SFM fail, triggering the activation of the safety channel for a comfort stop. But then, the VSM (i.e., the safety channel) is also failing. As a result, the CSM (i.e., the emergency channel) sends braking commands to the vehicle actuators to stop the vehicle as quickly as possible without using any sensor data.

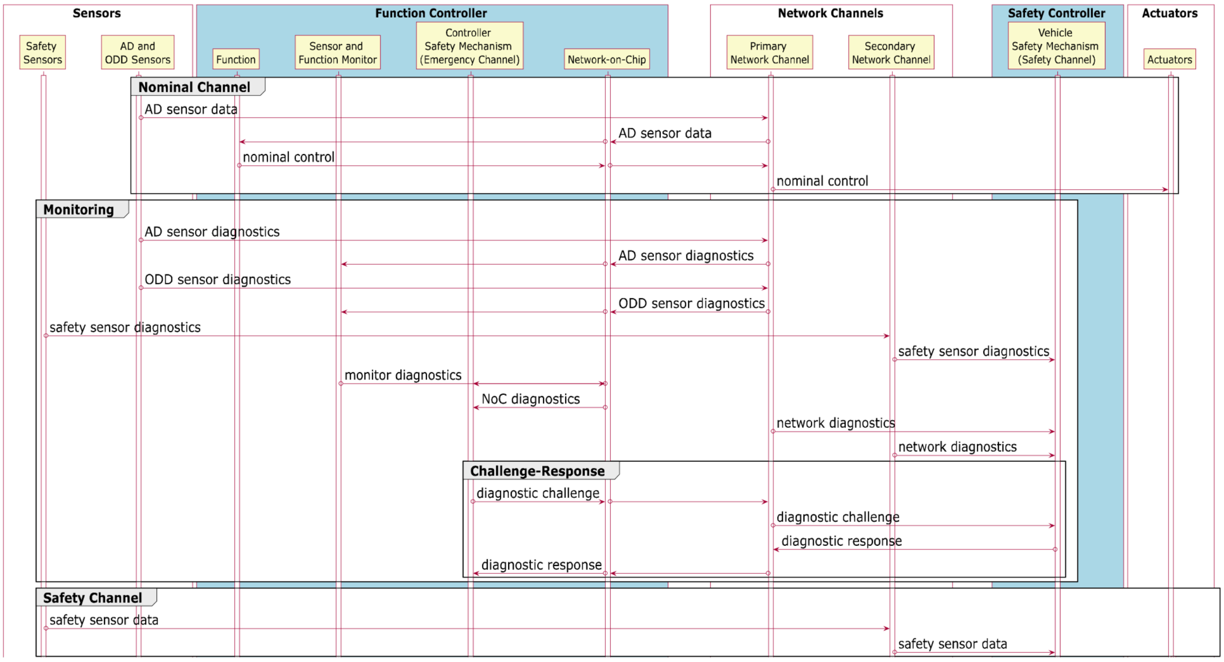


Figure 20: Sequence diagram depicted in [43] of the DSM in nominal mode.

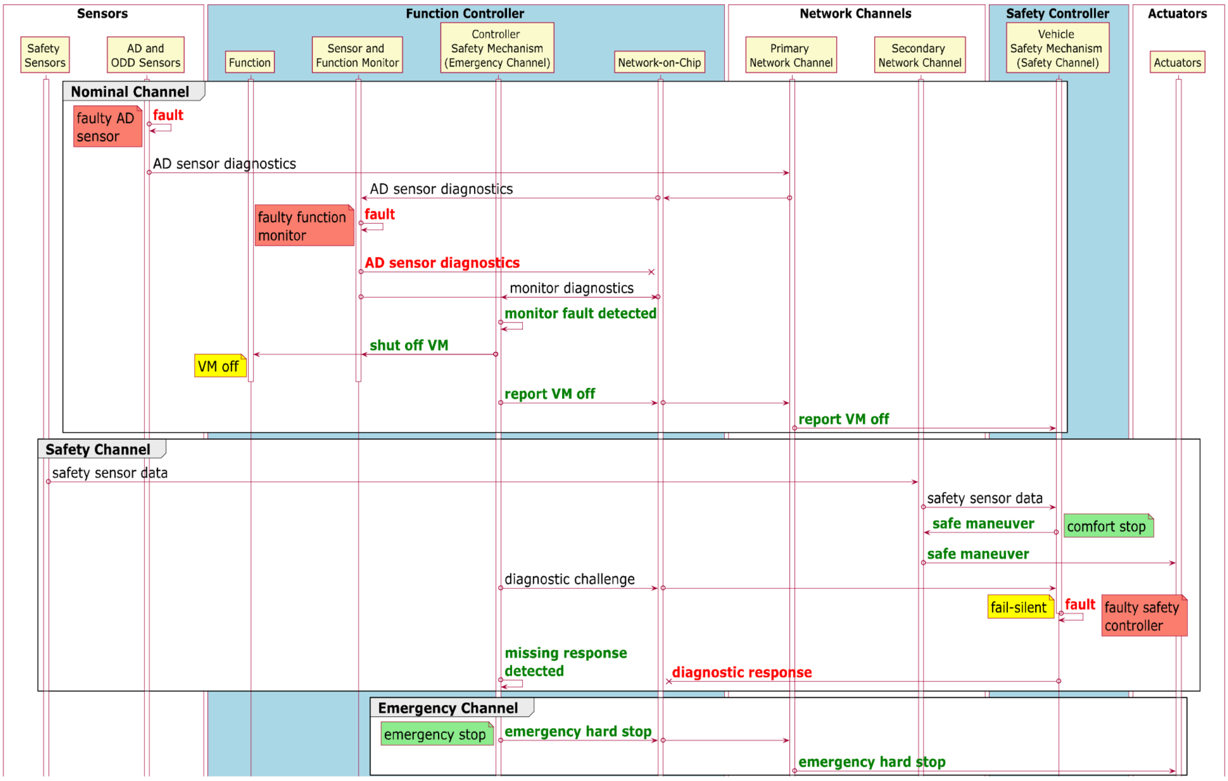


Figure 21: Sequence diagram depicted in [43] of the DSM in case of multiple faulty components.

4 ARCHITECTURE EVALUATION

4.1 EVALUATION PROCESS

Section 3 presented architectures that strive to be practical solutions to the question of how, conceptually, an automated driving architecture should be designed. The architectures are not limited to a specific use case of automated driving, and for the most part they do not explicitly target a specific design criterion, like those described in section 2, although without doubt the safety and availability of the ADI was a key consideration in the design of most candidate architectures.

In this section we seek to describe the merits or potential drawbacks each architecture might show with respect to the evaluation criteria. To form an unbiased basis for the evaluation, we first start with a generic evaluation of each architecture in section 4.2, by listing a number of observations related to each criterion, i.e., properties of each architecture perceived by the Safety and Architecture Working Group team and (if not obvious) their significance for the specific criterion.

As the second step, we give the concrete evaluation of the architectures under the defined reference use case of an SAE L4 Highway Pilot function in section 4.3. To this end, we evaluate the significance of each criterion for that use case – as some will be must-haves for the conceptual architecture, while others might be of lesser significance or merely nice to have. Next, we directly compare the architectures, considering the observed properties from the generic evaluation and inferring merits or weaknesses with respect to each evaluation criterion, and finally ranking them under the criterion.

While some findings may be of principle nature and not easy to overcome, for others the conceptual nature of the architectures and the high level of their descriptions may leave room to define countermeasures against weaknesses in a further, more detailed design step. Also, depending on the particular use case and environment, the relative significance of the evaluation criteria may change, and criteria might be modified and/or added. It is therefore important to emphasize that the evaluation that we provide is not intended as an absolute and final judgement. Rather, it may be understood as a blueprint for the readers of this report for how to analyze an architecture, identify deficiencies, and derive improvement measures in a systematic way.

4.2 GENERIC EVALUATION

4.2.1 EVALUATION OF THE SINGLE-CHANNEL ARCHITECTURE

4.2.1.1 AVAILABILITY

Availability of the system

Monolithic design pattern: The ability to provide DDT in both nominal and failure conditions is dependent on availability of a single FCU with interfaces identical to the external interfaces of the AD Intelligence.

- The lack of a redundant source of the DDT means that single-point faults can lead to failure – and therefore the unavailability – of the entire AD intelligence.
- To mitigate (but not eliminate) the vulnerability to single-point faults, redundancy measures within the internal implementation of the subcomponents could be attempted.

Diagnostics scheme

System remains silent when faults are detected.

- There are no degradation measures that can be triggered by the detection of faults.

Degradation scheme

Fail-silent only. In case of a fault, no complex fallback functionality is provided.

- No possibility of switching to a degraded mode which offers an MRM.

Scalability towards higher availability

Pattern does not support functional degradation.

- There are no redundant channels that can support nominal function availability.
- There are no fallback channels that can provide a DDT for achieving an MRC (MRM).

4.2.1.2 RELIABILITY

Availability of nominal function

A single trajectory result is generated from a single sensor fusion module and executed by the actuator system.

- Potentially lowest base failure rate from low HW and SW complexity.
- The absence of redundant results and fallback paths makes the pattern prone to false negatives/false positives and other functional deficiencies that cannot be compensated for.
- The lack of fallback puts pressure on the complex functionality to reach the highest possible ASIL.

4.2.1.3 CYBERSECURITY

Interactions between subsystems

Availability of the entire DDT is dependent on one channel.

- Lack of SW diversity results in a lack of diverse cybersecurity access paths. If the one channel is compromised, the entire DDT is jeopardized.

Interactions with external systems

Communication to external networks for updates can be expected.

- If implementation complexity becomes high to meet all functional requirements with a single channel, it may not be possible to keep the frequency of updates low, which in turn makes the usage of more secure update mechanisms less feasible (OTA may be required instead).

4.2.1.4 SCALABILITY

Scalability towards differing offering levels

Pattern does not support re-using existing subsystems.

- It is only feasible for this pattern to contribute to scaling strategy by being an existing subsystem that can be re-used by a more elaborate architecture pattern (e.g., provide the fallback channel).

4.2.1.5 SIMPLICITY

Number, complexity, and performance of subsystems

Superficially, the pattern appears simple, but the implementation complexity could be significant if use-case requirements are high.

- High development costs may be necessary for a single channel to offer a complex functionality that reaches the highest possible ASIL.

Required level of diversity

There are no clearly separated subsystems in this pattern.

- Redundancy measures within the internal implementation need to be attempted to offer the highest possible functionality.
- No independent development and integration of additional channels.

Complexity of validation

There are no clearly separated subsystems in this pattern that require independent verification.

- Potential of high HW/SW complexity increases the possible effort required for review, test, analysis, etc.
- Absence of other channels eliminates necessity of integration verification.

4.2.1.6 SAFETY OF THE INTENDED FUNCTIONALITY

Support to accommodate functional insufficiencies

- A single channel responsible for all SOTIF issues appears to be impractical for L4 systems unless the ODD is very restricted.
- The monolithic nature of the single-channel pattern means that it is likely to draw its high-performance requirement from ML-based modules, and that there is no opportunity for further algorithmic/functionality diversity.
- To cover all potential triggering conditions, the use of all possible sensor modalities may be necessary to decide about the nominal trajectory. Depending on the type of failure and the fault containment unit design, a fallback trajectory may or may not be available. The fallback trajectory could be derived from the last known nominal one or be predefined.
- Single-channel patterns may require more analysis and testing efforts, independently of the use case, because of its intrinsic complexity compared with multiple-channel approaches.

Support to manage operational conditions

- Due to the lack of redundancy and diversity, this architecture has a very limited capability to react safely to dynamic operational conditions in general.
- It is assumed that the ODD monitoring functionality is part of the single fault containment unit. Thus, if it fails, a predefined fallback trajectory or MRM must be used. Such a fallback maneuver is used without considering the operational conditions and not in an independent way, which can be unsafe.

4.2.2 EVALUATION OF THE MAJORITY VOTING (M-OO-N) ARCHITECTURE

4.2.2.1 AVAILABILITY

Availability of the system

The majority voting (M-oo-N) architecture consists of homogenous or heterogeneous hot redundancy, and it continues to provide correct results until at least M modules/channels have no fault. A channel consists of data acquisition, data processing, and output processing sub-systems. Additionally, input sensors, voter, and actuators form the complete majority voting architecture.

- The M-oo-N voting logic is used in the voter component to allow the system to provide the required functionality in the presence of random faults without losing the input data. This ensures the availability of the system compared to single-channel architecture.
- The common sensor and the voter could be a result for a single point of failure and hence should be carefully designed and tested. The sensor's and voter's availability is crucial in this concept and hence needs to be implemented in an ASIL D and fail-operational way. N different sensors and/or voters could be implemented instead, with the drawback of increased system complexity.
- The architecture is not appropriate for handling systematic faults. Since the N channels are identical and could have the same possible systematic fault, the system will continue to work, producing invalid data. To overcome this weakness, heterogeneous modules that perform the same functionality could be used, with the drawback of increased development cost. Additionally, heterogeneous sensors and modules will result in potentially different (but each correct) outputs that cannot easily be voted on. This is probably unavoidable even with identical sensors, as already different mounting positions will lead to slightly different world perceptions.

Degradation scheme

The majority voting architecture provides for a controlled degradation of functionality under fault conditions.

- If at least M channels do not produce an identical result, then the voter can either:
  - Switch the system into its fail-safe state if available
  - Discard the result and raise a flag to indicate an unsafe condition
  - Choose an output based on a predefined policy

4.2.2.2 RELIABILITY

Availability of nominal functionality

In the majority voting architecture, the voting element plays the main role since it is used to find the correct result by performing the M-oo-N voting strategy. The voter needs to be designed simple and fault-tolerant.

- To overcome the single point of failure in the sensor (or set of sensors), separate ones can be used for each channel. In the case of varying response speeds of the sensors (if applicable), this needs to be handled.
- To overcome the problem of systematic faults, a hardware diversity concept can be used in the implementation. In this case, the possible deviation in value or time between the correct outputs needs to be taken into consideration in the design of the voter.
- If a homogenous implementation (same hardware as well as software) is used for all the N modules, this leads to difficulty in handling systematic faults in all modules.
- Coincident faults in multiple channels could out-vote the correct channel. Diversity of implementation could help reduce this risk.

4.2.2.3 CYBERSECURITY

Interactions between subsystems

This architecture requires low amounts of data to be exchanged between channels.

- The channels are independent of each other. This can make it harder to corrupt multiple channels after an attack on one of them. A security incident can only occur if there are multiple attacks on different channels. The voter implementation is simple and reliable and is thus assumed to be harder to attack.
- Since the N channels are identical, a known implementation vulnerability could be exploited in all channels. To overcome this weakness, diverse modules that perform the same functionality could be used.

Interactions with external systems

It is expected that this architecture will require communication outside of the system. All subsystems are likely to require regular updates. As a result, the system would likely be connected to a network to perform these updates.

- The voter, sensors and actuators are simple and likely to require less frequent updates, which may thus use more secure update mechanisms.
- The redundant N channels are complex and likely to require updates to fix defects or to install improved functionality, which are thus likely to use OTA. Use of OTA provides an exposed attack surface for gaining remote access to the system.

4.2.2.4 SCALABILITY

Scalability towards different offering levels

It may be possible to carry over components from an existing SAE L2 system.

- The architecture contains identical channels and is thus easily scalable. The reliability of the system increases with an increased number of channels.
- The voter can be modified with easy steps to accommodate the new channels in the voting policy.

4.2.2.5 SIMPLICITY

Number, complexity, and performance of subsystems

This architecture includes a number of complex and high-performance subsystems, which interact in a controlled manner.

- The N modules run separately in parallel, hence they have only little influence on the executing time compared to the single-channel architecture. The voter adds a small delay, affecting the response time from reading the input signal to generating the control actuating signal.
- The architecture does not change the level of modifiability compared to the single-channel architecture, i.e., if one wants to modify the functionality for the M-oo-N architecture, the effort will be almost equivalent to modifying a simple single channel.
- The development cost is also comparable to single-channel architecture, since the N channels are identical and use the same algorithm and the same software. If heterogeneous modules are used to prevent common causes, the effort will scale with the number of channels.
- The architecture has a high recurring cost due to the use of N parallel modules. The recurring cost is N*100% compared to the single-channel architecture.
- Due to the differences in inputs or differences in the implementation of each channel, the outputs from each channel could vary slightly even in the case of a unanimous decision. This requires an approximate comparison within the voter that may be challenging to create and tune correctly.

Required level of diversity

This architecture has no subsystems requiring independent development, safety verification/validation, and integration.

- The N redundant channels are identical and use the same algorithm and the same software. Therefore, all of these components would have the same functional requirements, resulting in identical development, verification, and validation. Therefore, the development cost is also comparable to single-channel architecture.

Complexity of validation

This architecture has no diverse subsystems requiring independent verification.

- The N redundant channels are identical and use the same algorithm and the same software. Therefore, all of these components would have the same functional requirements, resulting in identical development, verification, and validation.

4.2.2.6 SAFETY OF THE INTENDED FUNCTIONALITY

Support to accommodate functional insufficiencies

- Due to the identical channels, there is a high risk of potential common cause false negatives of perception or planning results.
- A majority voting approach does not contribute to addressing functional insufficiencies. The diversity of the different channels is the determining factor. An extended voting approach which considers the "best" trajectory and other SOTIF-related relevant outputs would be needed. In the case of sufficiently diverse channels, the resolution of conflicts due to inexact voting may be a challenge.

Support to manage operational conditions

- Very limited capability to react safely to changing operational conditions.
- Like the single-channel architecture, the majority voting approach may not be capable of adequately handling operational modes other than the nominal one.

4.2.3 EVALUATION OF THE CHANNEL-WISE DCF ARCHITECTURE

4.2.3.1 AVAILABILITY

Availability of the system

This architecture provides a good degree of fault tolerance. There is a redundant system capable of controlling the vehicle under both nominal and failure conditions.

- CCDSS (Computer Controlled Driving Subsystem) has a redundant backup with CEHSS (Critical Event Handling Subsystem). Fault-tolerant communication channels are available, including a secondary set of AD and ODD sensors. FTDSS (Fault-Tolerant Decision Subsystem) is driving the Fault-Tolerant Actuator.
- There are no obvious single points of failure.

Degradation scheme

DCF provides for a controlled degradation of functionality under fault conditions.

- Degradation can occur in a controlled manner, depending on the nature of faults detected. There are two modes of operation, the second providing a minimum level of comfort to the passengers: Full AD mode and an emergency stop when it is no longer safe to continue operating.
- There exists full hardware and software redundancy between the primary CCDSS, the MSS (Monitoring Subsystem) and the CEHSS. CCDSS and CEHSS redundant systems are both fully capable of controlling the vehicle autonomously. MSS monitors the safety of the setpoints from the CCDSS and the CEHSS via the FTDSS and reports a failure to CCDSS and FTDSS. All subsystems would be developed to fail independently and thus diversely. A relatively low-complexity CEHSS is capable of an immediate “safety stop” or a “steady moving state” if the CCDSS fail.

4.2.3.2 RELIABILITY

Availability of nominal functionality

DCF relies on an evaluation of the outputs of CCDSS and safety evaluation of the resulting trajectory in MSS. The FTDSS needs to be designed simple and fault-tolerant.

- DCF requires a minimum mix of hardware and software. The resulting architectural footprint may result in a lower potential defect rate due to lower complexity in CEHSS and FTDSS.
- There is an active checking approach performed by the MSS to evaluate the safety of the actuator commands being issued to the CCDSS and resulting trajectory in MSS, with the intent being to detect a failure of the CCDSS, allowing a failover to occur. This checking methodology is clearly defined, but not proven. Differences between CCDSS and MSS world model are minimized by clear time synchronization of all sensors.

4.2.3.3 CYBERSECURITY

Interactions between subsystems

This architecture requires low amounts of data to be exchanged between subsystems. MSS gets the trajectory planning from CCDSS, setpoints of CEHSS via FTDSS and FTDSS gets all actuator commands from CCDSS and CEHSS and the decision about safety from MSS.

- CCDSS + MSS are independent of CEHSS, which can make it harder to corrupt additional subsystems after an attack on one of those. A security incident can only occur if there are two different attacks on two of the complex subsystems (MSS, CCDSS, CEHSS).
- FTDSS could be completely free of software or is at least a simple system with a very low attack surface. It is the only point where a single successful attack would compromise the overall system.

Interactions with external systems

It is expected that this architecture will require communication outside of the system. All subsystems beside FTDSS are likely to require regular updates. As a result, the system would likely be connected to a network to perform these updates.

- Several subsystems (CCDSS and MSS) are highly complex and likely to require frequent updates to fix defects or to install improved functionality, and are thus likely to use OTA. Use of OTA provides an exposed attack surface for gaining remote access to the system.
- The other subsystem CEHSS is simpler and likely to require less frequent updates, and may thus use more secure update mechanisms.

4.2.3.4 SCALABILITY

Scalability towards different offering levels

It may be possible to carry over components from an existing SAE L2 system.

- The CCDSS and CEHSS have similar functionality and requirements to an SAE L2 ADAS. Those components could potentially be reused in a lower-tier vehicle with only an L2 system, resulting in some cost savings.
- The MSS and FTDSS are specific to SAE L3 or higher. These components would likely be developed and manufactured only for the fully functional AD system.

4.2.3.5 SIMPLICITY

Number, complexity, and performance of subsystems

DCF includes a number of complex and high-performance subsystems, which interact in a controlled manner.

- The CCDSS is expected to have very high complexity and require very high performance. The MSS is expected to have moderate complexity and high performance, i.e., simpler than the CCDSS.
- The CEHSS is expected to have moderate complexity and low performance needs. The FTDSS is expected to be small and of low complexity, but fault-tolerant.
- Two or three disjunct sensor sets are required. Although they may be shared partly, this adds to the manufacturing cost and complexity and would require additional wiring within the vehicle.
- A fault-tolerant communication network is required. Similar to the redundant sensors, this also adds to the manufacturing complexity and wiring needed.
- Due to the high reliability requirements on multiple sets of sensor data inputs, some fault monitoring would be necessary for each of them.

Required level of diversity

This architecture has a high number of diverse subsystems fostering independent development, safety verification/validation, and integration.

- The CCDSS, MSS, and the CEHSS perform different functions within the vehicle. The CEHSS is capable of driving fully autonomously, however it makes use of a different set of input sensors than the CCDSS and is intended only for short-term use when the CCDSS or MSS are failing. The CEHSS is also expected to be a somewhat simpler control algorithm than the CCDSS. Most likely there would be little commonality between the two. Therefore, all three of these components would have very different functional requirements, likely resulting in very diverse development, verification, and validation.

Complexity of validation

This architecture has a number of diverse subsystems requiring independent verification.

- The CCDSS, MSS, CEHSS, and the FTDSS perform different functions within the vehicle. Unit test verification of each independent component within the system would require an appropriate test harness to be developed to enable full control of inputs to and access to outputs from each component.
- Multiple custom-built integration testing harnesses would be required to verify subsets of the components as they are integrated together. The different components would be developed in parallel and would require a staged integration testing approach. Each subsystem works as an independent fault containment unit.
- Assuming parallel development of all components (except perhaps the CCDSS and MSS), it is likely all would perform their certification activities as SEooC. The final functional safety concept would then need to integrate all of the different out-of-context components and validate all of the functional safety requirements for the completed system.
- The CEHSS and the FTDSS can be treated as SEooCs, allowing for parallel development and largely independent validation, verification, and certification. This is likely to significantly lower validation efforts compared to validation just on the level of the integrated system.
- The MSS is tightly coupled to the CCDSS as it performs checks on it. This makes a joint validation necessary.

4.2.3.6 SAFETY OF THE INTENDED FUNCTIONALITY

Support to accommodate functional insufficiencies

- This architecture can support a high level of coverage of triggering conditions and functional insufficiencies if diverse algorithmic implementation is applied.
- The CDSS, MSS, and CEHSS use their own sensors.

Support to manage operational conditions

- The CCDSS contains mechanisms to detect an ODD exit.
- The MSS also acts as an ODD monitor and checks the CCDSS output for violations of the ODD-related assumptions and the prospective trajectory. There is an unresolved risk that the MSS fails to recognize an unsafe trajectory, which may be reduced with a sufficiently diverse algorithmic implementation.
- The fallback channel (CEHSS) does not consider a detailed ODD, but assumes any drivable road condition without weather or geofence limitation.

4.2.4 EVALUATION OF THE LAYER-WISE DCF ARCHITECTURE

4.2.4.1 AVAILABILITY

Availability of the system

The layer-wise DCF architecture provides a good degree of fault tolerance. There is a redundant channel approach which is capable of controlling the vehicle under both nominal and failure conditions.

- A primary doer-checker pair (primary channel) controls the vehicle during normal mode, and a secondary doer-checker pair (safing [fallback] channel) provides a degraded mode of operation in case the primary pair fails. The Primary Unit output is checked for its safety by the Primary Safety Gate which fails silently and inhibits the Primary Unit output (from being sent to the Priority Selector) if the safety check is not OK. The Priority Selector will then select the output of the Safing Unit if safe (and thus not inhibited by the Safing Safety Gate), and otherwise perform an MSTOP that brings the vehicle to a stop in a low-level way (e.g., braking maneuver in planned trajectory).
- Under the consideration of the fail-silence characteristic of the Safing Safety Gate, the worst case scenario is avoided, i.e. the case in which this gate is faulty and as a result potentially sending erroneous permissive envelopes and at the same time erroneously inhibiting the Safing Unit output. Furthermore, considering the fail-operational characteristic of the Priority Selector, the Vehicle Control is the only single point of failure in the architecture.
- The channels are required to be diverse, which excludes common cause failures that could simultaneously affect the redundant channels.
- The sensor data of the safing channel is the input to both channels' safety gates, which could possibly inhibit the doer's and fallback's output simultaneously, resulting in a less safe MSTOP (coupling factor shared information input) as per the generalized example figure.
- The primary channel and safing channel seem to have joint perception inputs as per the system instantiation figure.

Degradation scheme

The layer-wise DCF architecture provides for a controlled degradation of functionality under fault conditions.

- The Safing Planner provides a trajectory that allows a safe stop, ideally at the side of the road. If the Safing Planner's output itself is checked as "not safe" (in addition to the primary planner), the Priority Selector performs an MSTOP (braking maneuver in planned trajectory).
- There exists full hardware and software redundancy between the primary channel and the safing channel. Both channels are able to control the vehicle in a safe way.
- If the safing channel's safety gate crashes (fails silently), it is stated to inhibit both channels' output to the Priority Selector, causing an MSTOP, which is a less safe option than letting the primary channel bring the car to a safe stop.
- It is left open how the Priority Selector shall trigger an MSTOP if it fails silently.

4.2.4.2 RELIABILITY

Availability of nominal functionality

The layer-wise DCF architecture provides a check for safety of the planned trajectories as well as a check of the trajectory execution (if architectural pattern also applied for this specific functionality) by safety gates at the respective stage. If a fault is detected in the primary channel, where the nominal functionality is allocated, it fails silently and is no longer considered in the arbitration algorithm of the Priority Selector.

- The safety gates have a high integrity level and false-positives are thus limited. The availability of the nominal function depends on the reliability of the primary unit itself and is therefore not diminished by the architectural concept.
- The safing channel produces trajectories designed to enable the vehicle to stop quickly (still updating its world model). These trajectories will be selected by the Priority Selector if the primary channel produces or executes unsafe trajectories.
- The introduction of a permissive envelope (which we did not encounter in other architectural candidates), qualitatively judged, might reduce reliability.

4.2.4.3 CYBERSECURITY

Interactions between subsystems

- The clearly separated and independent components make it harder to corrupt additional components when one of them is attacked.
- A safety-relevant fault caused by a cybersecurity attack in the primary or safing unit will be detected by the safety gates, which will exclude the respective channel from arbitration performed by the Priority Selector.
- However, the high number of safety gates throughout the different layers in the primary and safing channels might pose a cybersecurity risk, as each one's fail silence characteristic and output can be manipulated jointly - compromising overall system safety through a successful attack to one single safety gate.

Interactions with external systems

The layer-wise DCF architecture needs communication outside of the system in order to make regular updates possible, especially for the complex primary and safing units, which are an entry port for security attacks.

- The underlying simplex architecture provides complex subsystems (units = doers) and safety subsystems (safety gates = checkers). Only the complex units of the respective channels are expected to need regular updates that are most feasible over the air.
- The safety gates as well as the Primary Selector and Vehicle Control are simpler subsystems. They are expected to require less frequent updates, which make the usage of more secure update mechanisms feasible.

4.2.4.4 SCALABILITY

Scalability towards different offering levels

It may be possible to carry over components from an existing SAE L2 system.

- The primary units and safing units (in the respective layers) as doers have similar functionality and requirements to an SAE L2 ADAS. There is a good chance that at least some components of these units can be reused for SAE L2 ADAS or vice versa.
- The safety gates and the Primary Selector are only useful in SAE L3 and higher systems, i.e., automated driving systems.

4.2.4.5 SIMPLICITY

Number, complexity, and performance of subsystems

The layer-wise DCF architecture includes complex and high-performance subsystems as well as simple and low-performance subsystems.

- The primary units in the respective layers are expected to have high complexity and high performance needs.
- The safing units in the respective layers are expected to have moderate complexity and moderate performance needs.
- The safety gates in the respective channels are expected to have moderate complexity and moderate performance needs.
- Two disjunct sensor sets are required. This increases the unit costs of the automated driving item and its complexity.
- A fault-tolerant communication network on the sensor side is proposed, to enable sharing of sensors and help reduce cost. As this is not a decisive nor distinctive element of the architecture, we do not evaluate it further.
- The Primary Selector and the Vehicle Control are expected to have low complexity and low performance needs.
- Due to the high reliability on multiple sets of sensor data inputs, some fault monitoring would be necessary for each of them.
- The “architectural pattern” is repeated n times through the layers of a system, which increases the number of subsystems.
- The “basic architectural approach” is repeated n times through the layers of a system, which decreases the complexity of subsystems at the respective stages.

Required level of diversity

The layer-wise DCF architecture consists of several diverse subsystems requiring independent development, including verification and validation.

- The layer-wise DCF architecture is based on diverse sets of fail-silent doer-checker paired functional blocks in each architectural stage requiring independent development with independent verification and validation.
- Each stage in the respective channels consists of a low integrity level unit and a high integrity level safety gate checking the safety of the unit's output. The split into high integrity level monitoring and low integrity level intended functionality requires a technical independence between the architectural elements that includes diversity (e.g., see preconditions for ASIL decomposition in ISO 26262). The safety gates have different functionalities implemented compared to the units, which brings with it different developments. However, homogenous elements and similar design approaches must be avoided for the sake of diversity.

Complexity of validation

This architecture has a number of diverse subsystems requiring independent verification.

- The units and safety gates, as well as the Primary Selector and Vehicle Control perform different functions within the vehicle. Unit test verification of each independent sub-system within the system would require an appropriate test harness to be developed to enable full control of inputs to and access to outputs from each subsystem.
- Multiple custom-built integration testing harnesses would be required to verify subsets of the components as they are integrated together. The different components would be developed in parallel and would require a staged integration testing approach. Each subsystem works as an independent fault containment unit.
- It is assumed that the doer-checker pairs (as a single element) of the primary and safing channel as well as the Primary Selector and Vehicle Control are developed in parallel to SEooCs. The final functional safety concept would then need to integrate all of the different out-of-context components and validate all of the functional safety requirements for the completed system.
- The high number of subsystems due to the instantiation of the "architectural pattern" at the various stages results in high integration efforts, which include integration verification.
- The high number of subsystems due to the instantiation of the "architectural pattern" at the various stages means less complex subsystems compared to a single-layer approach which eases the verification efforts (review, test, analysis) for the respective subsystems and also raises the efficiency of finding anomalies.
- Due to the independence requirements between a high number of subsystems adherent to the architectural pattern applied at multiple stages, a complex and challenging dependent failure analysis must be performed.

4.2.4.6 SAFETY OF THE INTENDED FUNCTIONALITY

Support to accommodate functional insufficiencies
• This architecture supports a sufficiently diverse implementation to cover the safe handling of triggering conditions and functional insufficiencies. • The multi-channel approach not only ensures fail-degraded behavior but also promotes diverse implementation. • The layered approach showing separation of concerns facilitates analysis and implementation of measures to address functional insufficiencies (e.g., limitations of perception algorithms and outputs are handled separately from the ones related to planning). This implies that the checkers may be simpler. This specialization also has a positive impact on V&V effort, maintainability, and reusability, all of which are relevant for SOTIF. • The large number of subsystems may negatively affect the response time in critical situations. This, in turn, can be compensated for by a faster reaction, given the simpler or earlier checking.

Support to manage operational conditions
• A fallback pipeline (Safing Units and Gates of different stages) is included in the architecture. It uses an independent set of sensors. • It is assumed that at least the checkers perform the ODD monitoring functionality and other SOTIF-related checks. No details are included in the patent regarding this aspect. • The checkers of the fallback channel perform buffering of the last safe output to ensure fail-operational (degraded) vehicle behavior.

4.2.5 EVALUATION OF THE DSM ARCHITECTURE

4.2.5.1 AVAILABILITY

Availability of the system
This architecture provides a high degree of fault tolerance. There are multiple redundant subsystems capable of controlling the vehicle under both nominal and failure conditions. • Each subsystem is equipped with a backup. Redundant communication channels are available, including a secondary set of AD and ODD sensors. • In some cases, there are dual-redundant subsystems. The CSM acts as a backup for the VSM, which in turn acts as a backup for the FUN. • There are no obvious single points of failure.

Diagnostics scheme

The architecture includes multiple layers of diagnostic checking.

- Continuous diagnostic checking is performed on the functional sensors, safety sensors, as well as between various subsystems within the architecture.
- A continuous (cross-checking) challenge and response mechanism between the VSM and CSM confirms that both subsystems are within their safe operational parameters.

Degradation scheme

DSM provides for a controlled degradation of functionality under failure conditions.

- Degradation occurs in a controlled manner, depending on the nature of faults detected. There are five different modes of operation, each providing diminishing levels of comfort to the passengers: Full AD mode, a "detour" mode when immediate repair is needed, a "comfort stop" occurring at the next opportunity, an immediate "safety stop" when the fault(s) require urgent reaction, and an emergency stop when it is no longer safe to continue operating.
- There exists full hardware and software redundancy between the primary FUN and the VSM. These two redundant systems are both fully capable of controlling the vehicle autonomously. In addition, the CSM is capable of an emergency stop when necessary if both the FUN and VSM fail, acting as an additional layer in the degradation scheme.

4.2.5.2 RELIABILITY

Availability of nominal functionality

DSM provides two subsystems capable of fully autonomous operation. The CSM compares the outputs of the FUN and VSM, enabling the detection of failures.

- CSM subsystem performs continuous comparisons between the actuator outputs of the FUN and VSM. This enables rapid detection of a fault in the FUN system and immediate failover to the VSM. Such dual-redundant fully autonomous subsystems should provide a high level of reliability.
- The active checking approach performed by the CSM is unclearly defined and may be impractical if the asked-for agreement is too strict. Differences between FUN and VSM actuator commands must be expected due to differences in sensor inputs and computational algorithms. These differences may not indicate failure but rather two possible successful decisions (e.g., veer right to avoid a pothole, versus veer left to avoid a pothole). Being too strict in checking may result in false positives, whereas being too lenient may result in reactions which are too slow in a true failure situation. The CSM may thus need to resort to the MRM, which reduces the availability of the nominal functionality.
- DSM requires a complex mix of hardware and software. The resulting architectural footprint may result in a higher potential defect rate within subsystems or in the interactions between subsystems.

4.2.5.3 CYBERSECURITY

Interactions between subsystems

This architecture requires significant amounts of data to be exchanged between subsystems, including a high number of interactions between those subsystems.

- If a subsystem is compromised, the relatively large number of interfaces (e.g., due to additional monitoring mechanisms) can make it easier to corrupt additional subsystems.

Interactions with external systems

It is expected that this architecture will require communication outside of the system. All subsystems are likely to require regular updates. As a result, the system would likely be connected to a network to perform these updates.

- Several subsystems (mostly the FUN and only to a lesser extent the associated SFM) are highly complex and likely to require frequent updates to fix defects or to install improved functionality, which are thus likely to use OTA. Use of OTA provides an exposed attack surface for gaining remote access to the system.
- The other subsystems (CSM and VSM) are simpler and likely to require less frequent updates, which may thus use more secure update mechanisms.

4.2.5.4 SCALABILITY

Scalability towards higher availability

This architecture appears to be extensible to achieve higher availability, e.g., for fully driverless AD use cases.

- There is the possibility to add more VMs (adding more FUN and SFM modules) for higher availability or for load balancing if performance is an issue.
- There is only one VSM, which could prove a bottleneck for increasing availability if it is prone to failure.

Scalability towards different offering levels

It may be possible to carry over components from an existing SAE L2 system for use in the DSM.

- The FUN has similar functionality and requirements as an SAE L2 ADAS. This component could potentially be reused in a lower tier vehicle with only an L2 system, resulting in some cost savings.
- Other components of the system are specific to SAE L3 or higher (SFM, CSM, and VSM). These components would likely be developed and manufactured only for the fully functional AD system.

4.2.5.5 SIMPLICITY

Number, complexity, and performance of subsystems

DSM includes a high number of complex and high-performance subsystems, which interact in a complicated manner.

- The FUN is expected to have very high complexity and require very high performance. The VSM is expected to have slightly lower complexity and performance than the FUN.
- The CSM is expected to have moderate complexity and low performance needs. The checking approach for CSM is unclear but may turn out to be more complex than expected. The SFM is expected to have low complexity and low performance.
- Two disjunct sensor sets are required. This adds to the manufacturing cost and complexity and would require additional wiring within the vehicle.
- Two redundant high-bandwidth communication networks are required. Similar to the redundant sensors, this also adds to the manufacturing complexity and wiring needed.
- Due to the high reliability on two disjunct sets of sensor data inputs, some fault monitoring would be necessary for each of them.

Required level of diversity

This architecture has a high number of diverse subsystems requiring independent development, safety verification/validation, and integration.

- The FUN and the SFM are complementary, making it easier to ensure sufficient independence.
- The FUN, SFM, CSM, and the VSM perform different functions within the vehicle. The VSM is capable of driving fully autonomously, however it makes use of a different set of input sensors than the FUN and is intended only for short-term use when the FUN is failing. The VSM is also expected to be a somewhat simpler control algorithm than the FUN. Most likely there would be little commonality between the two. Therefore, all four of these components would have very different functional requirements, resulting in very diverse development, verification, and validation.

Complexity of validation

This architecture has a high number of diverse subsystems requiring independent verification.

- The FUN, SFM, CSM, and the VSM perform different functions within the vehicle. Unit test verification of each independent subsystem within the system would require an appropriate test harness to be developed to enable full control of inputs to and access to outputs from each component.
- Multiple custom-built integration testing harnesses would be required to verify subsets of the components as they are integrated together. The different components would be developed in parallel and would require a staged integration testing approach. For example, the SFM is tightly coupled to the FUN as it performs checks on it. They would likely be integrated, validated, and verified together initially. Next, the VSM would be added and verified with the FUN and SFM. And lastly, the CSM is tightly coupled to both the FUN and the VSM as it performs a comparison between them. It would be integrated, validated, and verified with the completed system.
- Assuming parallel development of all components (except perhaps the FUN and SFM), it is likely all would perform their certification activities separately (potentially as SEooC). The final functional safety concept would then need to integrate all of the different out-of-context components and validate all of the functional safety requirements for the completed system.
- The FUN and the VSM can be treated as SEooCs, allowing for parallel development and largely independent validation, verification, and certification. This is likely to significantly lower validation efforts compared to validation just on the level of the integrated system.

4.2.5.6 SAFETY OF THE INTENDED FUNCTIONALITY

Support to accommodate functional insufficiencies

This architecture considers several SOTIF aspects in the responsibilities of its subsystems.

- The SFM is explicitly intended to act as an ODD checker.
- The FUN, SFM, and VSM perform different functions within the vehicle. They can be implemented in a diverse way on the algorithmic level.
- A diverse set of sensors can be used.
- SFM and VSM collect and evaluate diagnostics data from the two sensor sets.

Support to manage operational conditions

- The SFM is explicitly intended to act as an ODD checker.
- Other aspects to ensure safe usage are not explicitly mentioned but could be considered in the implementation.

4.3 SPECIFIC EVALUATION IN THE CONTEXT OF THE REFERENCE AD USE CASE

4.3.1 RELEVANCE OF THE EVALUATION CRITERIA IN THE CONTEXT OF THE REFERENCE AD USE CASE

Depending on the selected use case, some KPIs may be more relevant than others. For instance, scalability (defined as a measure of an architecture's capability to be stepwise developed by extending SAE L2 systems) will likely not be relevant for urban SAE L5 robotaxis, which tend to be developed from scratch and are not intended to be sold as optional functions of standard OEM offerings to end customers.
Conversely, scalability may be highly relevant for a Highway Pilot function, which might be developed as a natural extension of highway-oriented L2 applications.

In the following, we attempt to give an assessment of the selected KPIs in the context of the reference use case of a Level 4 Highway Pilot. We employ the following ratings for the KPIs:

- Must-have
- Important
- Beneficial
- Unimportant

Readers of this document are encouraged to apply their own rating, in their specific use cases, innovation space and constraints from commercial, technical, or legacy requirements.

AVAILABILITY

In the context of an L4 Highway Pilot, availability of the system until successful completion of an MRM will become a formal safety goal with an ISO 26262 ASIL D target to be met (pending a formal HARA being conducted), as a system failure in a dense highway traffic situation will usually not be controllable by the driver and the consequences of a crash might be fatal. Therefore, availability is rated as a **must-have** for the reference use case.

RELIABILITY

In the reference use case of an SAE L4 Highway Pilot, reliability (defined as the continuous availability of the full, nominal functionality) is highly desirable from a vehicle user's perspective and shall be maximized. A switch to degraded functionality, e.g., executing an MRM, will be at least an annoyance or more likely disturbing for the passengers; frequent ones will lead to severe customer complaints, but will at least not lead to harm. Therefore, reliability is rated as **important** for the reference use case.

CYBERSECURITY

Vulnerability to cybersecurity threats impacts system safety, as an intruder might deactivate an essential safety mechanism or even maliciously manipulate essential autonomous driving functions like sensor inputs or trajectory planning. Still, an ADI by itself will not be able to fully avert cybersecurity risks, as many system functions (e.g., the sensors and actuators) are outside its scope and additional mechanisms like gateways and firewalls are needed. Therefore, although resilience to cybersecurity attacks is a must-have for the vehicle and the AD system as a whole, it is “just” considered **important** for the ADI system in the context of the reference use case.

SCALABILITY

Since the chosen reference use case of an L4 Highway Pilot may be developed as a natural extension of highway-oriented legacy L2 functions, parts of those functions (legacy sensors, ECUs, application components) might need to be incorporated into the realization of the L4 system. Conversely, SAE L2 functions might be implemented by a subset of components of the L4 system, which has been developed from scratch. In both cases, the L4 functionality might be marketed as optional equipment, and potentially a significant share of the overall vehicle volume might support L2 functions only. To implement such a concept in a commercially viable way, scalability is considered **important** for the reference use case.

SIMPLICITY

Although simpler than L5 functions or urban use cases, the algorithmic and system complexity for an L4 Highway Pilot remains high and verification and validation efforts might be prohibitive if not supported by a suitable system architecture. The established concept of “divide and conquer”, i.e., a conceptually clean, modular architecture with a well-arranged number of components of clear purpose, simple interfaces, and clear delimitations to each other, will be at least important if not a **must-have** for the reference use case.

SAFETY OF THE INTENDED FUNCTIONALITY

SOTIF is probably the key property and requirement that lay persons and the general public associate with autonomous driving functions, and technical as well as authority reports about incidents with autonomous cars mostly focus on function aspects and deficiencies (like object detection capabilities). In the context of the reference use case, SOTIF is considered a **must-have**.

4.3.2 ASSESSMENT OF THE CANDIDATE ARCHITECTURES UNDER THE EVALUATION CRITERIA

4.3.2.1 AVAILABILITY ASSESSMENT

Variant	Assessment
Single-channel	This architecture is obviously very sensitive to single points of failure. To make it somewhat resilient to such failures, several “internal” redundancy measures will likely need to be installed in a detailed architecture phase or even in an implementation phase, potentially in an ad-hoc way (making it hard to verify their sufficiency and completeness).
Majority voting	The M-oo-N voting architecture with homogeneous channels addresses random HW faults well, but is susceptible to common-cause failures, as the complex AD algorithms will not usually be suitable for full ASIL D development. Conversely, heterogeneous channels are not suitable for voting, as channels might each exhibit different (but valid) driving policies, and therefore a faulty channel might not be identifiable. This is especially true for the practical case of 1-oo-3 (TMR), where the problem is likely not solvable.
Channel-wise DCF	The channel-wise DCF architecture scores highly under the availability criterion, as it exhibits no obvious single point of failure (provided that the FTDSS subsystem is implemented in a fault-tolerant way) and due to the asymmetric approach, with its diversity of the channels, also has a high potential to rule out common cause faults.
Layer-wise DCF	The layer-wise DCF architecture scores highly on the availability criterion, due to its primary and secondary channels plus the MSTOP (blind stop) capability. However, the published description suggests potential single points of failure that would need to be avoided, e.g., using the same occupancy grid in the primary and secondary channels (whereas using that same input for the planners and safety gates of each channel can be beneficial to avoid false positives - if used correctly to restrict the planner’s decision space, not to extend it). Also, the symmetric architecture suggests sensitivity to common cause faults in the underlying implementation, which would need to be avoided.
DSM	The Distributed Safety Mechanisms architecture is intended to support the availability KPI, due to its multiple layers and redundancy mechanisms. It has some similarity to the channel-wise DCF architecture but offers additional degradation steps, giving in principle the potential for higher availability; in the concrete implementation proposed, it does not have a clear separation of the functional (FUN) and monitoring (SFM) channels, but implements both within the same SOC and virtual machine, and seems therefore highly sensitive to common cause faults.

CONCLUSION:
For the reference AD use case of an L4 Highway Pilot, the channel-wise DCF and the layer-wise DCF seem to be the architectures of choice from an availability point of view. DSM is considered problematic due to its common-cause failure sensitivity, and majority voting is considered highly problematic (if not unsuitable) in the practical case of non-deterministic channels. Single-channel is considered to be unsuitable.

4.3.2.2 RELIABILITY ASSESSMENT

Variant	Assessment
Single-channel	The resilience to functional deficiencies is not supported by any architectural measure but depends strictly on the internal implementation of its subcomponents. As such, degradation measures will likely need to be installed in a detailed architecture phase or even in an implementation phase, potentially in an ad-hoc way (making it hard to verify their sufficiency and completeness).
Majority voting	The M-oo-N voting architecture will provide high reliability, as each of its channels is conceptually capable of providing the full nominal functionality and may even provide degraded modes. In fact, a comparable level of capability for each channel is a precondition for successful voting. A practical implementation as 1-oo-3 (TMR) will still exhibit high reliability.
Channel-wise DCF	The channel-wise DCF architecture can potentially score highly under the reliability KPI but depends on the concrete capability level of its CCDSS and MSS subsystems, and on the parameterization of the MSS (which initiates the potentially degraded mode of the CEHSS), to not produce false positives. For the CCDSS, being aware of the limits that will be enforced by the MSS would be a helpful addition to the architecture.
Layer-wise DCF	The layer-wise DCF architecture, like the channel-wise DCF, can potentially score highly under the reliability criterion, but depends on the concrete capability level of its primary channel and the monitoring subsystems contained therein. To not produce false positives, it may foresee precautions like restricting the primary's decision space by the limits imposed by the monitor (although this is not detailed in the published description).
DSM	The Distributed Safety Mechanisms architecture will provide high reliability, due to its multiple and differentiated degradation steps.

CONCLUSION:
For the reference AD use case of an L4 Highway Pilot, majority voting and DSM seem to be the most capable architectures to sustain nominal functionality. Channel-wise DCF and layer-wise DCF can approximate this to some extent but will fall into degraded mode more often (due to their focus on safety). Single-channel does not support reliability at the architectural level, it is strictly implementation dependent.

4.3.2.3 CYBERSECURITY ASSESSMENT

Variant	Assessment
Single-channel	This architecture is critical from a cybersecurity point of view, as its single channel does not provide any architectural partitioning but exposes its complete functionality to a malicious intruder.
Majority voting	The M-oo-N voting architecture has beneficial properties from a cybersecurity perspective, as its channels are highly separated and exchange little (if any) information, and the voting component itself is expected to be simple and well-separated. However, if its channels are implemented homogeneously, this will be highly susceptible to exposing a common vulnerability.
Channel-wise DCF	The channel-wise DCF architecture scores highly under the cybersecurity KPI, as its clearly separated components exchange only a small amount of well-defined information and are highly diverse, potentially avoiding common vulnerabilities.
Layer-wise DCF	The layer-wise DCF architecture scores highly under the cybersecurity KPI, due to its clearly separated components. However, a successful attack to one single (of several) safety gates would compromise the overall system.
DSM	The Distributed Safety Mechanisms architecture seems to be more vulnerable from a cybersecurity point of view, as its high number of interactions between subsystems and (partly) missing separation might make it more exposed to attackers.

CONCLUSION:
For the reference AD use case of an L4 Highway Pilot, channel-wise DCF has the highest resilience against cybersecurity attacks, followed by layer-wise DCF with its higher exposure due to the larger number of safety gates. DSM seems to be more vulnerable due to its high number of interactions and tightly coupled components. Majority voting will be vulnerable in the case of homogeneous channels. Architecture-wise, single-channel does not provide any protection from cybersecurity threats.

4.3.2.4 SCALABILITY ASSESSMENT

Variant	Assessment
Single-channel	This architecture does not provide any scaling options.
Majority voting	The M-oo-N voting architecture appears to be highly scalable, as one of its channels can be used to downscale to an L2 system or could be derived by extending an existing L2 system. Likewise, it could be upscaled by adding channels.
Channel-wise DCF	The main component of the channel-wise DCF architecture (CCDSS) can be used to downscale to an L2 system or could be derived by extending an existing L2 system.
Layer-wise DCF	The primary components of the layer-wise DCF architecture can be used to downscale to an L2 system or could be derived by extending an existing L2 system.
DSM	Some of the components of the Distributed Safety Mechanisms may be derived by extending an existing L2 system. It is not obvious, however, how it could be downscaled to an L2 system by essentially just removing L4-related components. It could be upscaled to L5 by adding FUN/SFM components, but its VSM would need to be substantially extended.

CONCLUSION:

For the reference AD use case of an L4 Highway Pilot, the majority voting architecture seems to be the best option to both downscale to L2 and upscale to L5. Channel-wise and layer-wise DCF seem to provide good capabilities to downscale to an L2 system or leverage L2 system developments, whereas DSM seem to be a better fit for upscaling to an L5 system. Single-channel does not support scaling at all.

4.3.2.5 SIMPLICITY ASSESSMENT

Variant	Assessment
Single-channel	This architecture superficially seems to be simple, but its monolithic approach and lack of clearly separated subsystems will lead to high complexity and effort for implementation, verification, and validation.
Majority voting	The M-oo-N voting architecture appears to score highly under the simplicity criterion, due to its regular structure. However, each individual channel might have similar properties to the single-channel architecture, with comparable consequences for implementation and verification efforts. Relaxation on the individual channels due to the subsequent voting might be offset by efforts to identify faulty channels correctly. This might be a challenge for the system integrator, and for TMR (1-oo-3) as a practical option, it is questionable whether this can be solved at all.
Channel-wise DCF	The channel-wise DCF architecture is conceptually simple, as its clearly separated components with distinguished purposes and well-defined message exchange enable modular, separate development and simpler overall safety assessment. CCDSS will be the most complex component, but less effort than one of the majority voting's channels, due to the highly diverse MSS supervision. Efforts for the system integrator seem to be reasonable.
Layer-wise DCF	The layer-wise DCF architecture has clearly separated components with distinguished purposes and well-defined message exchange, but the higher number of components and level of information exchange (compared to the channel-wise DCF) will increase the burden on the system integrator and complicate overall safety assessment. On the other hand, the layered approach might enable a more modular, separate development.
DSM	The FUN, SFM, CSM, and the VSM modules of the Distributed Safety Mechanisms are separated, enabling separate development and verification, but interact in complex and highly diverse ways, thus putting a high burden on the system integrator and complicating the overall safety assessment. The proposed architecture also mixes functional aspects (FUN, SFM, VSM) with system integrity aspects (CSM) and implementation aspects (middleware, virtual machine) – it will thus be critical to decompose and assign the required system properties to the entities of the architecture in a clear, consistent, and complete way.

CONCLUSION:
For the reference AD use case of an L4 Highway Pilot, channel-wise DCF and layer-wise DCF seem to be preferable with respect to simplicity, i.e., reasonable integration and validation efforts. DSM is highly complex, especially for the system integrator. Majority voting seems simple, but the individual channels' complexity is high, as is their integration in the case of heterogeneous channels. Single-channel is superficially the simplest but is expected to require high verification and validation efforts.

4.3.2.6 SOTIF ASSESSMENT

Variant	Assessment
Single-channel	Due to its monolithic nature, this architecture is obviously very sensitive to functional deficiencies and to deviations from the nominal conditions, especially since it will need to rely on machine learning to achieve the performance goals and there is no visible means of supervision or diversity.
Majority voting	The M-oo-N voting architecture with homogeneous channels is susceptible to common-cause failures by functional deficiencies or deviations from the nominal conditions. Conversely, heterogeneous channels are not suitable for voting, as channels might each exhibit different (but valid) driving policies, and therefore a faulty channel might not be identifiable. This is especially true for the practical case of 1-oo-3 (TMR), where the problem is likely not solvable.
Channel-wise DCF	The channel-wise DCF architecture scores highly under the SOTIF criterion, as it exhibits a natural diversity between the CCDSS and the MSS, and the CEHSS (being explicitly foreseen for out-of-ODD operation) is likely implemented very differently than the CCDSS. This architecture also quite naturally manages changes to the nominal conditions.
Layer-wise DCF	The layer-wise DCF architecture scores highly under the SOTIF criterion, as it employs multiple checkers and safety gates both on its primary and secondary channels, promoting modularity and diversity which also has a positive impact on development and V&V. It may be capable of addressing deviations from the nominal conditions well, but this seems not to be explicitly foreseen.
DSM	The Distributed Safety Mechanisms architecture highly promotes SOTIF, due to the different functions performed by its FUN, SFM and VSM modules. Management of off-nominal conditions is also explicitly foreseen.

CONCLUSION:
For the reference AD use case of an L4 Highway Pilot, the channel-wise DCF, the layer-wise DCF and the DSM are the architectures of choice from a SOTIF point of view. Majority voting is highly problematic (if not unsuitable) in the case of heterogeneous channels, as correct voting cannot be ensured. Majority voting with homogeneous channels and the single-channel architecture are highly sensitive to functional deficiencies and off-nominal conditions, and therefore likely unsuitable.

4.3.3 EVALUATION SUMMARY

The following table gives a summarizing overview of the evaluation findings, if and how the criteria are supported by the respective architectures:

	Single-Channel	Majority Voting	Channel-wise DCF	Layer-wise DCF	DSM
Availability	Not supported	Homogeneous: yes, but common cause risk Heterogeneous: no	Concept focuses on availability	Concept focuses on availability, risk of single point failures	Sensitivity to common cause failures
Reliability	Not supported	Homogeneous: low Heterogeneous: high	Reasonable reliability, implementation dependent	Reasonable reliability, with precautions by architecture	Multiple differentiated degradation steps
Cyber-security	Not supported	Homogeneous: low Heterogeneous : high	Diverse structure with high resilience	Diverse structure, but multiple single-attack points	Diverse structure, but complexity might expose vulnerabilities
Scalability	Not supported	Omit/add channels to scale	Omit channels to downscale to L2	Omit channels to downscale to L2	Downscaling to L2 not straightforward, but potential to upscale to L5
Simplicity	High inner complexity	Simple architecture, complex channels	Simple, clear concept	Structured, medium complexity	High complexity for the integrator
SOTIF Support	Not supported	Not supported due to homogeneous architecture	Structure supports SOTIF	Structure supports SOTIF	Structure supports SOTIF

In general, we find that asymmetric architectures (Channel-wise DCF, Layer-wise DCF and DSM) are better suited than symmetric ones (Single-Channel, Majority Voting) for the complexity in Automated Driving.

Their quite naturally independently developed and complementary channels can compensate for each other's weaknesses, compared to the essentially identical or potentially even monolithic implementations of symmetric architectures.

The asymmetric architectures basically employ two design patterns and combine them in different ways:

- Doer/Checker: One subsystem performs the function, the other one monitors it.
- Active/Hot Stand-By: One subsystem is active, and the other is on stand-by; if the active is unavailable or unsafe, the stand-by takes over.

A combination of these patterns allows for a sound partitioning into modules with simple and purposeful interfaces that can be independently verified and whose integration is straightforward and readily verifiable. Also, these patterns lead to a limited, well-arranged and predictable number of system states under both nominal and off-nominal conditions. A solid and verifiable safety argumentation can then be constructed systematically based on the individually developed modules and their successful integration.

5 IMPLEMENTATION CONSIDERATIONS

5.1 HW MAPPING CONSIDERATIONS

In this section, a short introduction to some aspects of HW refinement of the conceptual system architecture is given. This is not a complete list of aspects.

5.1.1 HIGH AVAILABILITY AND VEHICLE OPERATING STATES

Architectures in the Automated Driving (AD) context in general should not limit their functionality to a dedicated vehicle operating state (like parking, standing still, driving slowly etc.), but should work in fault-free condition in all vehicle states. But the loss of the functionality can lead to a hazardous event only in specific vehicle operating states. One valid approach for degradation of Automated Driving functionality is to change the operating state to lower severity or exposure, for instance by lowering the vehicle speed in a controlled way. In this context, please check ISO 26262:2018-10 §12.

Emergency operation exposure time as reaction to a fault should be limited if the ASIL capability of the item is lower than the ASIL rating of the possible hazard. If after the occurrence of the fault, the vehicle operating states are not changed, then the ASIL is the same as that derived from the HARA and no ASIL decomposition of main path and a potential redundant path is allowed.

For redundant paths, a dependent failure analysis should be executed to find and eliminate common cause initiators.

5.1.2 COMMON CAUSE INITIATORS

ISO 26262:2018-9 §7 requires assessment of Common Cause Initiators (CCI):

a. Random HW faults
Many of the system architectures use redundant channels to mitigate random HW faults in one channel by providing the same function in the redundant channel. There is a very small chance that in the Safety Goal-relevant time interval a random HW fault is detected in the redundant channel as well. A good strategy to treat this case is to use diverse configuration of fault reaction in the first and redundant channel.

b. Development faults
Those are covered by ASIL D development process, requirement-driven development flow and tool chain qualification process. A Development Process Documentation (DPD) can provide information related to the following topics:

   i. Development Process
   ii. Development Environment

iii. Requirement Management

A Quality Process Documentation (QPD) can provide references to cover various quality management and production-related topics. Both contribute to a Quality Process Management with the goal of providing the best avoidance of systematic development faults.

c. Manufacturing faults
The manufacturer should monitor the compliance with the related standards, e.g., by Audits, Production Assessment and Process FMEA activities. Particular focus should be put on the verification by testing of characteristics determined by analogue circuitries, including in quasi-digital parts such as memory. ISO/TS16949 certificates should be provided by all manufacturing sites documenting process compliance. An ISO 9001 certificate covers all sites, locations and organizational units of a manufacturer. TS16949 certificates cover all production sites, headquarters, automotive design centers, and sales. AS 9100 certificates cover production sites in America.
The Production Part Approval Process (Produktionsteil-Abnahmeverfahren, PPAP) [AIAG] is comparable to the PPA Production Process and Product Approval (PFF Produktionsprozess- und Produkt- Freigabe) [VDA]. Both procedures are reflected in the ZVEI PPAP Guideline. The Automotive Industry Action Group (AIAG) has developed a common PPAP standard as part of the Advanced Product Quality Planning (APQP) to use a common terminology and standard forms to document project status. Companies may have their own individual requirements. PPAP is the documentation (snapshot) of the current state of the product design, functionality, and reliability as well as the production processes used.

d. Installation faults + e) Repair faults
This CCI category shall be mainly addressed by OEM and TIER1 suppliers. The guidance given by suppliers in their user manuals and safety application guidelines shall be obeyed.

e. +h) Environmental factor incl. stress
The prototypes and series components of ADS should be subject to Environmental Stress as defined by semiconductor standards and by the automotive industry, such as AEC-Q100. OEM and their TIER1 suppliers shall analyze the potential impact on their diversity claim.

f. Common external resources
The functional safety of external resources, such as power supply, debug support and communication interfaces shall be analyzed for potential common causes to redundant channels.

5.1.3 CLOCK, POWER, RESET, DEBUG AND TEST FAILURES

Infrastructure functions in Automated Driving systems are typically common cause initiators on the hardware level.

The clock configuration of an automated driving system is defined during the development phase and is usually static during runtime. Therefore, any systematic failure affecting its functionality or monitoring capability is assumed to be found during integration verification. A diverse crystal oscillator type or PCB layout for redundant and diverse paths can reduce dependent failures. Diverse configuration settings of clock upscaling and distribution can reduce dependent failure.

Systematic faults in power supply circuits can affect the voltage regulator characteristic. Extreme corner cases (which could escape system validation) are unlikely to happen identically in redundant and diverse paths for an Automated Driving system.

During the operation of an Automated Driving system, a reset of hardware components can be used as a reaction to detected faults, but also has a high impact on the availability of the system. Reset as failure reaction should be used only for failures which cannot be handled otherwise. A diverse configuration of all reset sources (especially fault reaction) can reduce dependent failures. Under certain conditions, the redundant path can be configured to ignore all reset sources (in the case of a fault detected in the main path).

Debug is meant to be used during SW development only, therefore its systematic failures do not affect the functionality of the Automated Driving system during runtime. During safety application, all debug functionality should be disabled. The only remaining systematic faults could result from SW activation with critical failure mode "unintended debug". Diverse SW implementation can reduce dependent failure (e.g., redundant path without any debug SW parts compiled).

Test functionalities based on Built-In Self-Test (BIST) are executed during startup only, therefore its systematic failures do not affect the functionality of the Automated Driving system during runtime. During safety application, all test functionality should be removed. Only remaining systematic faults could result from SW activation with critical failure mode "unintended test". Diverse SW implementation can reduce dependent failure (e.g., redundant path without any test SW parts compiled).

5.2 SW MAPPING CONSIDERATIONS

This section contains some selected topics to consider when analyzing the system architecture towards further refinement of the technical aspects at the software level.

5.2.1 SOFTWARE ARCHITECTURAL STYLES

The main aspect to consider is the software architectural design itself. An exhaustive description of software architecture styles (e.g., layered, monolithic, microkernel, pipes and filters, client-server, publisher-subscriber, event-driven) and their applications is beyond the scope of this report, but we can recommend using [45] to get a good overview. Furthermore, this report does not address how safety measures are appropriately integrated into such software architectures.

Regardless of the choice of architectural styles for individual software elements, there are common safety measures, which are listed in the following non-exhaustive list:

- Graceful degradation behavior by ensuring that there is no single point of failure, especially for middleware and service-oriented software components.
- Error detection and handling mechanisms, as described in ISO 26262 [2] Parts 6 and 10, as well as the capability to store diagnostic data.
- Use of adequate programming languages and techniques, including the application of design and coding guidelines, such as MISRA C, AUTOSAR C++, CERT.
- Performing architecture analysis, such as Failure Modes and Effects Analysis (FMEA) and Architecture Trade-off Analysis Method (ATAM) [45].
- Evaluation and optimization of metrics related to the quality aspects of the architecture (e.g., complexity, dependencies, stability of code and interfaces).

An important aspect to mention here is the shift from federated to centralized architectures in automotive systems. In such centralized architectures, the software is executed redundantly using the mechanisms of virtualization and containerization (i.e., with hypervisors coordinating resources and virtual machines processes). Consequently, the system is more flexible and hardware costs can be reduced. On the other hand, distributing processes to virtual machines brings some challenges in terms of integration and testing, as well as cybersecurity.

5.2.2 PROPERTY OF TECHNICAL INDEPENDENCE

As stated in ISO 26262 [2] Part 9, to achieve technical independence between components of the system, cascading and common cause failures that compromise a safety requirement shall be avoided. While the factors listed in 5.1.2 apply to hardware, similar classes of coupling factors shall be considered for software elements:

- Shared resources, e.g., use of identical software modules without further independence measures, use of mathematical or other software libraries.
- Shared information input, e.g., global variables, data or messages used by more than one software element.
- Systematic coupling, e.g., same software tools, same programming or modeling language, reuse of assumptions and requirements for different software implementations.
- Components of identical type, e.g., same source code generated twice.
- Communication, e.g., global variables, messaging, function calls with arguments passed.
- Unintended interface, e.g., same memory space.

5.2.3 SOFTWARE REUSE

Reusable software (e.g., third party software, libraries, FOSS) can significantly reduce the development effort for AD systems. Variant management, software configuration and the integration into different architectures are aspects that need to be done carefully to avoid dependability issues.

In addition to the requirements and recommendations for the development of SW-SEooCs and the qualification of software components contained in ISO 26262 [2] Parts 8 and 10, there are new standardization efforts that complement these and provide further guidance:

- ISO/AWI PAS 8926 Road vehicles – Functional safety – Qualification of pre-existing software products for safety-related applications (under development).
- Public initiatives such as the project Enabling Linux in Safety Applications (ELISA, see https://elisa.tech/).

5.2.4 SOFTWARE UPDATES

With the move to software-defined vehicles, architectures of AD systems must ensure regular, continuous updates of software elements in a safe manner for the long term, including over-the-air (OTA) ones. This capability is closely related to quality aspects such as modularity, modifiability, portability, extensibility, and verifiability, along with the challenge of additional safety and cybersecurity risks (e.g., risks associated with the use of cloud services).

To standardize the software update engineering process, ISO 24089 [46] has been recently published. It contains requirements and recommendations on planning, risk management, V&V, deployment, and monitoring of software updates, but does not include specific technologies or solutions.

5.2.5 REAL-TIME OPERATING SYSTEMS (RTOS) AND MIDDLEWARE

The software responsible for providing basic services and interfacing the software applications with the hardware (i.e., the electronic buses, CPUs, and ECUs) requires a high level of integrity. The well-known standard AUTOSAR (AUTomotive Open System Architecture, see www.autosar.org) has been largely used in its original version (i.e., Classic Platform) as the basis for traditional automotive functionalities such as engine control and transmission. For AD systems, however, more complex software applications and high-performance computations are to be supported. Thus, a middleware based on the new AUTOSAR Adaptive Platform includes advanced functionalities, such as:

- Runtime configuration
- OTA software updates
- Ethernet inter-ECU communication for the transmission of large data
- High-performance hardware
- Service-oriented communication
- Compatibility with other operating systems (e.g., Linux, Android)

Other capabilities that go beyond the AUTOSAR Adaptive standard might be required, for example scheduling and real-time guarantees for event chains across complex, multi-partition or multi-SOC architectures.

5.2.6 MACHINE LEARNING AND DATA-DRIVEN APPROACHES

The use of machine learning (ML) in the automotive industry was essentially introduced to address the challenges of the perception tasks (e.g., object recognition, pedestrian detection, signs recognition, road intersection detection). While the neural network model used might play a relevant role to ensure safe outputs, the performance of ML-based software depends mostly on data engineering aspects. Typical issues to avoid during development of such software are:

- Bias in data collection
- Patterns of mislabeling in training data
- Poor design of experiments for simulation validation

Another important aspect is the potentially non-deterministic behavior of ML-based software due to the inclusion of stochastic aspects in the training process or the concrete implementation. Architectures that integrate ML-based software require safety mechanisms such as redundancy and plausibility checks (e.g., safety wrappers) which makes them more complex. In general, analyzing reliability- and safety-related failure modes and mitigating them with appropriate error detection and handling mechanisms is one of the key challenges for ML-based software.

5.2.7 DATA MANAGEMENT

In addition to the safety implications of data-driven approaches in the context of ML, other aspects related to data management also require decision-making at the architecture and implementation level. Some safety aspects related to data management are:

- Data and software configuration management (e.g., to support variant management and software updates).
- Assuring safety of internal-to-vehicle data management (e.g., internal maps used for vehicle localization are uncorrupted and of a compatible version).
- Integrity of collecting, storing, and transmitting engineering field feedback data, including safety performance indicators.

5.2.8 TOOL QUALIFICATION

Tool evaluation and qualification processes are described in ISO 26262 [2] Parts 8 and 10. Due to the increasing complexity of the software development environment and the technologies used, the topic has become crucial. Continuous Integration (CI), static and dynamic code analysis, testing and simulation tools, model-based code generation, documentation generation: essentially all software engineering processes are becoming automated. While this is necessary to manage development and maintenance efforts, challenges arise from increasing reliance on the tool chain and infrastructure, with all the associated risks related to troubleshooting, cybersecurity, privacy, and safety.

To the degree that simulation is used to supplant vehicle testing, tool qualification of simulations, simulation models, and simulation orchestrators will become more critical. The same applies to the tools needed to mitigate the risk of data bias, inaccurate data labels, and data corruption in such simulation-based validations.

From a system architecture point of view, the use of different tools for redundant subsystems may be necessary to rule out common points of failure due to tool malfunctions.

5.3 SAFETY ARGUMENTATION

5.3.1 APPLICABLE SAFETY STANDARDS

With respect to safety, the relevant ISO standards are ISO 26262 (Functional Safety) and ISO 21448 (Safety of the Intended Functionality), which should be followed throughout the development of an AD system. Although compliance with those standards is not a formal (legal) requirement for vehicle homologation, they are considered “state of the art” (also in a legal sense) and therefore most OEMs adhere to them in their development processes and prescribe them to their suppliers.

5.3.2 ISO 26262 (FUNCTIONAL SAFETY) CONSIDERATIONS FOR THE ADI IMPLEMENTATION

ASIL ASSIGNMENT

It is safe to assume that an AD system for an L4 Highway Pilot will get the ASIL D level assigned, as all three relevant factors will contribute and lead to this highest classification in terms of ISO 26262:

- Severity will be S3 (highest), as a malfunction of the AD system during autonomous operation can lead to a fatal crash.
- Exposure will be E4 (highest), as the vehicle will be in a potentially hazardous situation during autonomous operation with high probability.
- Controllability will be C3 (highest), as a malfunction while in autonomous operation will not be controllable by the passengers (would require an immediate attention shift and takeover by the “driver”)[23].

For a concrete system, of course, a Hazard Analysis and Risk Assessment (HARA) needs to be conducted to derive safety goals and associated ASILs, but without doubt will lead to the classifications above for many of a Highway Pilot’s functions.

[23] Note that ISO 26262 refers to “the driver or other persons involved in the operational situation” for classifying the Controllability. Strictly speaking, there is no “driver” for an L4 function, but we apply the definition analogously, because for a Highway Pilot we still assume a person present in the driver’s seat, who needs to control the vehicle anyway until highway entry and after highway exit.

ASIL DECOMPOSITION

Note that the ASIL D assignment is valid for the AD system as a whole (formally: for its safety goals) and may be lowered for some of its constituents by appropriate ASIL decomposition. This involves partitioning the system into sub-components with lower ASIL that jointly realize the safety goal and must be integrated using ASIL D-compliant technical measures and processes. In the context of an L4 Highway Pilot, decomposition is also a practical necessity, as many components required to implement it are too complex to completely fulfill ASIL D criteria – like high-end SOCs, operating systems, or application software components.[24]

AVAILABILITY AS A SAFETY GOAL

ISO 26262 was originally for traditional powertrain, steering, braking or even ADAS (L1 or L2) functions, and those systems are usually developed with a correctness goal defined and a fail-silent system reaction, i.e., switch-off in the case of a malfunction.

For an L4 Highway Pilot with its fail-operational/fail-degraded requirement, however, the availability of the AD system becomes a safety goal with ASIL D too and needs to be met by applying appropriate technical and process measures – as any sudden non-availability of the AD system during L4 operation will not be controllable by the passengers and will likely cause harm (up to fatalities).

REDUNDANCY

On the architecture level, the availability goal is usually addressed by appropriate redundancy measures, which are installed to cope with the unavoidable failure of individual components of the system (for example, due to permanent or transient electronic faults, or due to residual SW errors). All architectures evaluated in this report exhibit such redundancy, except for the Single-Channel architecture.

SUFFICIENT INDEPENDENCE

Redundancy is not sufficient to ensure availability: sufficient independence of the redundant components must also be ensured, to avoid common cause or cascading failures which would cause redundant components to fail jointly and render the whole AD system unavailable. The same argumentation applies to ASIL decomposition: it is only allowed if the constituent components are independent of each other and cannot jointly violate the functional safety goals.

In the case of an L4 Highway Pilot, this means that decomposed / redundant system channels must be implemented in a sufficiently diverse fashion to rule out common cause failures that might impede either the functional correctness (e.g., in a doer-checker configuration) or the system availability (e.g., in a main-fallback configuration).

IS ASIL D ENOUGH?

ISO 26262 does not give any reference or consideration to the system's complexity – effectively an ASIL D SW component implemented with 5 kLOCs is considered to be equally safe as a system that contains 100 kLOCs. However, according to [10], a system with more than 10 kLOCs is likely to exhibit residual systematic SW errors, even when developed to the highest safety standards.

An L4 Highway Pilot can be considered a highly complex system and will for sure involve much more than 10 kLOCs for its implementation. Therefore, even the application of the ASIL D process to a complex system channel might not ensure system safety – instead, partitioning such a channel into smaller constituents with lower complexity, which are developed to ASIL D and can be readily (individually) verified, should be considered.

[24] Machine learning algorithms are often cited as intractable for development according to ISO 26262. Actually, their regular structure for the inference phase makes them quite easily compliant.

COMPLEX INTERACTIONS

All the investigated ADI architectures except the single-channel architecture exhibit some functionally and/or availability motivated partitioning that reflects the need for redundancy and decomposition. Some are already initially quite complex (DSM, layer-wise DCF), but even the simpler ones (channel-wise DCF, majority voting) are likely to require further partitioning within their channels, to arrive at technically tractable implementations. Such partitioning will also support the ASIL D argument, as reflected above.
In any case, one might end up with a significant number of components with a (combined) potentially enormous number of system states and complex internal interaction. To prove that the integrated system possesses the expected safety and availability properties, and does not exhibit any unintended emergent behavior, formal modelling and verification can be employed. Such a process can give mathematical correctness assurances, where human cognitive limits are exceeded, and manual verification would be too error-prone.

5.3.3 ISO 21448 (SOTIF) CONSIDERATIONS FOR THE ADI IMPLEMENTATION

The standard SOTIF-ensuring process according to ISO 21448 shall be followed when implementing the ADI, both on a system level and when implementing the channels and components of the architectures presented here.

Architecture considerations or dedicated architecture design and evaluation steps are not reflected in that standard; mentioned system modification steps to ensure SOTIF mostly focus on functional aspects and not architectural measures. Still, the chosen architecture will have a decisive impact on the efforts required to develop and validate an AD system.

However, certain architecture aspects are mentioned in ISO 21448, and closer examination shows that the architectures presented here quite naturally support the standard, and actually help to reduce the effort required for ensuring SOTIF.

SENSE-PLAN-ACT

The established Sense-Plan-Act paradigm is used in ISO to motivate a modular specification, design, and V&V approach, with individual qualitative and quantitative development goals for each layer. In the architectures described in this report, the analogous layering is explicitly foreseen only in the layer-wise DCF architecture. However, the other architectures can (and will need to) be broken down into a similar structure for their high-level building blocks, like the channels of the majority voting and channel-wise DCF architectures, the FUN layer of the DSM architecture, and in particular the single-channel architecture.

DDT FALLBACK

The "DDT fallback" is assumed as a functional entity (not necessarily an architecture element) in ISO 21448. This element is present in all the architectures discussed here, except the single-channel architecture, in an explicit manner (channel-wise DCF, layer-wise DCF, DSM) and implicitly through the multiple channels of the majority voting architecture.

ARCHITECTURE IMPACT ON V&V

ISO 21448 acknowledges that a suitable architecture can support more efficient verification and validation by enabling a modular approach and reducing the effort for V&V of individual components (compare also G3: Testing and simulation of very high safety-related availability of large monolithic system and D1: Fault Containment Units). Diversity and independence arguments (compare also D3: Diversity and redundancy for complex subsystems) are used, albeit in example form only and not as an integral part of the process to achieve SOTIF.

5.3.4 PROPOSED SUPPORTING STEPS FOR THE ADI IMPLEMENTATION

Potentially exceeding the standard ISO 26262 and ISO 21448 processes and reflecting the safety considerations from the previous sections, the following four development steps are suggested for the implementation of the described architectures of an L4 Highway Pilot:

STEP #1: SYSTEM PARTITIONING AND MAPPING

Partitioning the system into subsystems (see also D1: Fault Containment Units) that jointly implement the L4 Highway Pilot functionality is usually done as one integral step, to meet the system correctness goals, the availability goals and ensure feasibility of the ASIL D requirements. In fact, the presented architectures (except the Single-Channel architecture) largely anticipate this step, introducing redundancy to ensure system availability and checking instances to ensure system correctness (integrity).

Development step	Goal	Description
System Partitioning	System correctness	Ensure correct system outputs (e.g., trajectories) under the environmental and vehicle conditions: • Doer-Checker configuration in Layer-wise and Channel-wise DCF architectures • Channels in Majority Voting architecture • FUN/SFM/CSM in the DSM architecture
	System availability	Add redundancy to ensure availability of the system under component failures • Doer-Fallback configuration in Layer-wise and Channel-wise DCF architectures • Channels in Majority Voting architecture • VSM and Primary/Secondary Networks in the DSM architecture
	ASIL D feasibility	Further partition (modularize) the channels to enable ASIL D capable subsystems under HW and SW component constraints. This is usually not visible on the conceptual architecture level, but a necessary practical step in all architectures. Examples are: • Separate perception components per sensor • Low-level vs. high-level fusion • Trajectory planning and validation • Voting and decision components

The partitioned system architectures also serve as a basis for the formal ASIL decomposition; assignment of proper ASILs to the individual components is therefore necessary. We anticipate the following assignments:

Architecture	ASIL	Components
Single-Channel	D	Whole architecture
Majority Voter (1-oo-3)	B(D)[25]	Channels
	D	Voter (also fail-operational)
Channel-Wise DCF	B(D)	CCDSS, MSS, CEHSS
	D	FTDSS (also fail-operational)
Layer-Wise DCF	B(D)	Primary/Safing Planners, Primary/Safing Planner Safing Gates, Primary/Safing Trajectory Executor, Primary/Safing Trajectory Executor Gates (some integration functionality between these nodes may be ASIL D, but is not explicitly visible)
	D	Priority Selector, Vehicle Control (also fail-operational)
DSM	B(D)	FUN (Function), SFM (Sensor and Function Monitor), Primary Network, Secondary Network.
	D	CSM (Controller Safety Mechanism), VSM (Vehicle Safety Mechanism)

After the (logical) system partitioning and ASIL decomposition, the mapping of the components to available HW modules (SOCs) and SW partitions/operating systems needs to be performed, including selection of appropriate communication means between the components. This mapping will be largely guided by the functional demands (e.g., compute performance requirements of each logical component, network bandwidth required) and the available functional safety support by SOCs and the SW platform; it may be an iterative process until an optimum solution is found.
For the sake of generality, this report does not consider specific SOCs and SW platforms, and therefore does not dive into the mapping task further.

STEP #2: FORMAL MODELLING AND VERIFICATION

After the partitioning and mapping has been performed, a complex architecture with a significant number of components and interaction may be the result. To validate the result, formal methods can be used to deal with the potentially large, and often cognitively intractable number of system states and ensure the desired system properties with a mathematical proof.

[25] Instead of the B(D) + B(D) decompositions, other variants are also possible where multiple components work together: A(D) + C(D) or QM(D) + D.the learning phase with its non-deterministic properties that stands in the way. The main problem to solve lies in the SOTIF domain, not in the functional safety.

Development step	Goal	Description
Formal Modeling and Validation	Logical consistency, correctness, and system availability	Create formal system model and formal description of desired properties, and simulate on logical (conceptual architecture) level • Proof of desired properties • Absence of violations of such properties (e.g., absence of single points of failure) • Avoidance of unintended emergent behavior
	Logical - physical consistency, system availability	Add formal modeling of the mapping of the logical building blocks to physical components and simulate on physical (implemented architecture) level • Preservation of desired properties • Absence of violations of such properties (e.g., absence of single points of failure under the physical mapping

STEP #3: SUFFICIENT INDEPENDENCE ANALYSIS

The design goal D7: Mitigation of common-cause hazards purposefully prescribes introducing diversity on the architecture and implementation level. Nevertheless, in realistic implementation scenarios, the redundant channels of the conceptual system architectures might be implemented on homogeneous platforms (e.g., SoCs and OSs) and communication technologies. Also, the implementation of applications like AD algorithms might share a set of mathematical libraries, HW accelerator layers, etc. This might even be extended to joint perception and fusion components used by the different channels.
To achieve a sound safety argumentation, a systematic and detailed analysis of "sufficient independence" will need to be performed, to rule out common cause and cascading faults. Note that such an analysis will be necessary even if the components of each channel are sourced from different suppliers or implemented by different teams, as they might use identical third party components, derive from the same architecture specification (e.g., AUTOSAR), or use identical legacy IP blocks even within heterogeneous SOCs.

Development step	Goal	Description
Sufficient independence analysis	Ensure system availability and correctness	Systematically analyze HW and SW architecture on platform and application level, including communication network. • Prove absence of common cause faults in redundant elements, on platform and application level • Prove absence of cascading faults across the ADI architecture

STEP #4: MARKOV ANALYSIS

The overall failure rate needs to be determined and proven to be within a set target, usually around ASIL D metrics (10^{-8}/h) (compare section 2.2.7). For complex, fault-tolerant architectures with redundant paths like the ones described in this report, one cannot simply add up the failure rates of the constituents but must consider their interaction. This amounts to describing the overall system's states (like normal operation, degradation etc.) and their transition probabilities (derived from the individual component failure rates) in a Markov model and calculating the resulting overall failure rate.

Development step	Goal	Description
Markov analysis	Evaluate system failure rate	Model system states and transition probabilities from individual component failures; calculate overall failure rate. • Calculate overall failure rate, considering FuSa and SOTIF • Meet target failure rate

In addition to these supporting steps, the relevant standards like ISO 26262 and established engineering practices prescribe extensive testing and simulation for verifying the correctness, reliability and availability of complex AD systems and algorithms. For the use case of an SAE Level 4 Highway Pilot, these shall only be intensified compared to systems of lesser criticality. In this context, fault injection campaigns are of particular value and can challenge many design properties, such as sufficient independence of redundant channels, fault tolerance of arbiters and resilience of the overall systems against arbitrary faults in its components.

OUTLOOK

The further direction of the Safety & Architecture Working Group is still under discussion among the member companies of The Autonomous, and some of the potential work packages could be as follows:

- Extend the analysis to further use cases, such as SAE L5 or an Urban Pilot function, and see if and to what extent the evaluation of the relative merits and weaknesses of each architecture changes.
- Extend the analysis to other architectures that recently appeared in the market and literature, such as self-checking pairs and the "Safety Shell" architecture [30].
- Deepen the analysis to include more implementation aspects, e.g., to propose concrete HW and SW mappings and suitable ECU and networking architectures.
- Extend the report with practical guidelines for implementation tasks, such as how to check on and ensure logical completeness and consistency of an architecture, or how to evaluate and quantify the independence of computation channels.
- Work out an "architecture evaluation guideline" from the experiences gained throughout the presented work, to help the industry community apply a similar framework in their concrete development projects.

Whatever the future direction, working on this report has been a hugely gratifying experience for the team, and we are confident that it provides value to the community of industry players and academic institutions working on automated driving systems.

TERMINOLOGY

TERMINOLOGY FROM STANDARDS AND LITERATURE

The “Safety & Architecture” WG makes use of the terminology laid out in different industry standards and literature. Please refer to the listed standards for all terms not specifically defined in the following.

A small number of terms have been added as new definitions to clarify the scope of the “Safety & Architecture” WG.

For terms related to systems, faults, and failures, we use the following (in order of preference):

- ISO 26262:2018 “Road vehicles – Functional safety” [2]
- IEC 61508:2010 “Functional safety of electrical/electronic/programmable safety-related systems” [47]
- ISO 21448:2022 “Road vehicles – Safety of the Intended Functionality” [3]
- Algirdas Avizienis, J-C. Laprie, Brian Randell, and Carl Landwehr. “Basic concepts and taxonomy of dependable and secure computing.” IEEE transactions on dependable and secure computing 1, no. 1 (2004) [1]

For terms related to AD, we use the following (in order of preference):

- ISO/SAE PAS 22736 “Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles” [48] (based on SAE J3016_202104 [7])
- BSI PAS 1883:2020 “Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) - Specification” [49]

Please also refer to the relevant databases maintained by ISO and IEC:

- https://www.iso.org/obp/ui
- http://www.electropedia.org/

Term	Reference or definition	Notes
AD Intelligence	computational unit between the sensors and actuators	
Architecture	ISO 26262:2018-1	
Automated Driving System (ADS)	ISO/SAE PAS 22736	
Availability	ISO 26262:2018-1	
Cascading failures	ISO 26262:2018-1	• In literature, one will often find the following differentiation: Failures of redundant systems due to systematic weaknesses of the architecture are caused by common cause initiators (CCI) and coupling faults (= ISO 26262 cascading faults). In many standards this differentiation is not done, e.g., IEC 61508-6:2010, Annex D: The term CCF is often used to cover all kinds of dependent failures as it is done in this annex. According to an Exida 2010 Safetronic paper, ISO 26262 is the first standard to distinguish the two distinct phenomena.
Channel	IEC 61508:2010-4	
Common cause failure (CCF)	ISO 26262:2018-1	
Common mode failure (CMF)	ISO 26262:2018-1	
Conceptual architecture	An abstract, high-level architecture that does not specify technical (e.g., HW, SW) components.	• This is similar to the "system architectural design" required as an external input in ISO 26262:2018-3.
Controllability	ISO 26262:2018-1	
Coupling factors	ISO 26262:2018-1	
Dependability	Avizienis, TR 2004-47	The paper defines this as encompassing the following attributes (quote): • availability: readiness for correct service [see also ISO 26262:2018-1] • reliability: continuity of correct service • safety: absence of catastrophic consequences on the user(s) and the environment [see also ISO 26262:2018-1] • integrity: absence of improper system alterations • maintainability: ability to undergo modifications, and repairs
Dependent failures	ISO 26262:2018-1	
Dependent failure initiator (DFI)	ISO 26262:2018-1	

Term	Reference or definition	Notes
Diagnostic coverage (DC)	ISO 26262:2018-1	- In the context of ISO 26262, this only covers HW faults. For our purposes, we use the same term to cover SW faults as well, which goes hand in hand with the decision to quantify (systematic) SW faults.
Diversity	ISO 26262:2018-1	
Dual-point failure	ISO 26262:2018-1	
Dual-point fault	ISO 26262:2018-1	
Dynamic Driving Task (DDT)	ISO/SAE PAS 22736	
DDT fallback	ISO/SAE PAS 22736	
Dynamic elements	BSI PAS 1883	
Ego vehicle	BSI PAS 1883	- Used instead of "subject vehicle".
Element	ISO 26262:2018-1	
Emergent behavior		Behavior that cannot be attributed to one individual system alone, but arises in the interplay of various systems, components etc.
Environmental conditions	BSI PAS 1883	
Error	ISO 26262:2018-1	
Failure	ISO 26262:2018-1	
Fault	ISO 26262:2018-1	
Fault-Containment Unit (FCU)	*A Fault Containment Unit is a set of subsystems that shares one or more common resources that can be affected by a single fault and is assumed to fail independently from other FCUs. [50]*	- The definition is very strict in the sense that any potential dependent failure (detected during the analysis of dependent failures) between supposed FCUs would prove that they (by definition) are actually not FCUs. - The definition is interpreted to mean that potential dependent failures may exist between FCUs defined in the architecture, but have to be mitigated by either reducing the probability of the root cause, reducing the coupling factors, or controlling their effects.
Formal verification	ISO 26262:2018-1	
Functional insufficiency	ISO 21448:2022	Insufficiency of specification or performance insufficiency. Both terms are also defined in ISO 21448:2022.
Interface	Avizienis, TR 2004-47 [1]	- The paper distinguishes between the "service interface" and the "use interface".
Item	ISO 26262:2018-1	
Hazard	ISO 26262:2018-1	

Term	Reference or definition	Notes
Malfunction	ISO 26262:2018-1	- Malfunctions can arise due to multiple causes, e.g., faults, performance limitations [see also ISO 21448:2022], or unexpected behavior [see also ISO 21448:2022].
Mapping	The process of transforming a conceptual architecture into a technical HW and/or SW architecture.	- The same conceptual architecture can be mapped to many different HW/SW solutions. - Certain considerations need to be applied during the mapping to ensure properties of the conceptual architecture are not lost.
Minimal Risk Condition (MRC)	ISO/SAE PAS 22736	
Minimal Risk Maneuver (MRM)	BSI PAS 1883	- In a highway ODD, there are multiple possible MRMs, e.g., reducing speed and continuing to the next rest stop, pulling over to the emergency lane, or coming to a controlled stop in the current lane. These differ by their inherent safety and the capability and timeframe necessary to execute them.
Misuse	ISO 21448:2022	An example of a direct misuse is the activation of a highway pilot in an urban setting. An example of an indirect misuse is a driver falling asleep and not monitoring an L2 system during operation.
Monitor	ISO/SAE PAS 22736	
Object and Event Detection and Response (OEDR)	ISO/SAE PAS 22736	
Operational Design Domain (ODD)	ISO/SAE PAS 22736	
Output insufficiency	ISO 21448:2022	
Passenger car	ISO 26262:2018-1	
Performance limitation	ISO 21448:2022	
Random hardware fault	ISO 26262:2018-1	
Request to intervene	ISO/SAE PAS 22736	
Routine/normal operation	ISO/SAE PAS 22736	
Safety	ISO 26262:2018-1	
Safety architecture	ISO 26262:2018-1	
Safety case	ISO 26262:2018-1	
Safety Element out of Context (SEooC)	ISO 26262:2018-1	

Term	Reference or definition	Notes
Safety goal	ISO 26262:2018-1	
Scenery	BSI PAS 1883	
Service	Avizienis, TR 2004-47	
Severity	ISO 26262:2018-1	
System	ISO 26262:2018-1	
Systematic fault	ISO 26262:2018-1	
Triggering condition	ISO 21448:2022	
Validation	ISO 26262:2018-1	
Verification	ISO 26262:2018-1	
Vulnerable Road User (VRU)	BSI PAS 1883	

REFERENCES

[1] A. Avizienis, J.-C. Laprie, B. Randell and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, 2004.

[2] ISO, "ISO 26262:2018 Road vehicles - Functional safety," 2018.

[3] ISO, "ISO 21448:2022 Road vehicles - Safety of the Intended Functionality," 2022.

[4] Aptiv; Audi; Baidu; BMW; Continental; Daimler; FCA; Here; Infineon; Intel; Volkswagen, "Safety First for Automated Driving," 2019.

[5] H.-P. Schoener and J. Antona-Makoshi, "Testing for Tactical Safety of Autonomous Vehicles," in 30th Aachen Colloquium Sustainable Mobility, Aachen, Germany, 2021.

[6] ISO, "ISO/SAE 21434:2021 Road vehicles - Cybersecurity engineering," 2021.

[7] SAE, "SAE J3016 Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems," 2021.

[8] Economic Commission for Europe - Inland Transport Committee, "Proposal for a new UN Regulation on uniform provisions concerning the approval of vehicles with regards to Automated Lane Keeping System," 2020.

[9] H. Egerth and S. Yantis, "Visual Attention: Control, Representation and Time Course," Annual Review of Psychology, vol. 48, pp. 269-297, 1997.

[10] D. Dvorak, "NASA Study on Flight Software Complexity," Jet Propulsion Laboratory, California Institute of Technology, 2009.

[11] J. McDermid and T. Kelly, "Software in safety critical systems: achievement and prediction," Nuclear Future, vol. 2, no. 3, pp. 140-146, 2006.

[12] J. Gray, "Why do computers fail and what can be done about it?," Tandem Computer Corporation, 1985.

[13] Wikipedia, "Wikipedia entry on Heisenbug," [Online]. Available: https://en.wikipedia.org/wiki/Heisenbug. [Accessed 31 07 2023].

[14] G. Li, S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer and S. Keckler, "Understanding error propagation in deep learning neural networks (DNN) accelerators and applications," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2017.

[15] N. Kalra and S. Paddock, "Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?," Transportation Research Part A: Policy and Practice, vol. 94, pp. 182- 193, 2016.

[16] R. Young, "Automated driving system safety: Miles for 95% confidence in "vision zero"," SAE International Journal of Advances and Current Practices in Mobility, vol. 2, no. 6, pp. 3454-3480, 2020.

[17] X. Zhang, J. Tao, M. Törngren, J. Gaspar Sanchez, M. Rusyadi Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan, M. Nica and H. Felbinger, "Finding Critical Scenarios for Automated Driving Systems: A Systematic Mapping Study," IEEE Transactions on Software Engineering, vol. 49, no. 3, pp. 991-1026, 2023.

[18] Statistisches Bundesamt (Destatis), "Verkehrsunfälle 2018," 2020.

[19] P. Koushki and F. Balghunaim, "Determination and Analysis of Unreported Road Accidents in Riyadh, Saudi Arabia," Journal of King Saud University - Engineering Sciences, vol. 3, pp. 101-118, 1991.

[20] SAE, "SAE ARP4754A Guidelines for Development of Civil Aircraft and Systems," 2010.

[21] SAE, "SAE ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment," 1996.

[22] Y. Fu, J. Seemann, C. Hanselaar, T. Beurskens, A. Terechko, E. Silvas and W. Heemels, "Characterization and Mitigation of Insufficiencies in Automated Driving Systems," in The 27th International Conference on the Enhanced Safety of Vehicles (ESV), Yokohama, Japan, 2023.

[23] UL, "ANSI/UL 4600 (Ed. 3) Evaluation of Autonomous Products," 2023.

[24] SAE, "SAE J3018 Safety-Relevant Guidance for On-Road Testing of Prototype Automated Driving System (ADS)-Operated Vehicles," 2020.

[25] S. Shalev-Shwartz, S. Shammah and A. Shashua, "On a Formal Model of Safe and Scalable Self-driving Cars," Mobileye, 2017.

[26] H. Kopetz, "An Architecture for Driving Automation," 2020. [Online]. Available: http://www.the- autonomous.com.

[27] J. Lala, in First IFIP Workshop on Intelligent Vehicle Dependability & Security, 2021.

[28] Statistik Austria, "Straßenverkehrsunfälle mit Personenschaden, Jahresergebnisse 2018," 2019.

[29] P. Liu, R. Yang and Z. Xu, "How Safe Is Safe Enough for Self-Driving Vehicles?," Risk Analysis, vol. 39, no. 2, pp. 315-325, 2018.

[30] C. Hanselaar, E. Silvas, A. Terechko and W. Heemels, "Detection and Mitigation of Functional Insufficiencies in Autonomous Vehicles: The Safety Shell," in IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), 2022.

[31] A. Armoush, "Design patterns for safety-critical embedded systems," PhD thesis, RWTH Aachen University, 2010.

[32] Audi, "zFAS the Brain of piloted Driving and Parking (nVIDIA GPU Technology Conference)," 2015. [Online]. Available: https://on-demand.gputechconf.com/gtc/2015/presentation/S5637-Matthias- Rudolph.pdf.

[33] Audi, "Audi A8 - Central driver assistance controller (zFAS)," 7 2017. [Online]. Available: https://www.audi-technology-portal.de/en/electrics-electronics/driver-assistant-systems/audi-a8- central-driver-assistance-controller-zfas. [Accessed 21 11 2023].

[34] Tesla Inc., "Tesla AI Day 2022," 01 10 2022. [Online]. Available: https://www.youtube.com/watch?v=ODSJsviD_SU&ab_channel=Tesla.

[35] AutoPilot Review, "Tesla Hardware 4 – Full Details and Latest News," 2023. [Online]. Availa https://www.autopilotreview.com/tesla-hardware-4-rolling-out-to-new-vehicles/.

[36] M. L. Shooman, "Reliability of computer systems and networks: fault tolerance, analysis and design," in N-Modular Redundancy, Wiley-Interscience, 2002, pp. 145-201.

[37] H. Kopetz, Real Time Systems - Design Principles for Distributed Embedded Applications, Springer Verlag, 2012.

[38] P. S. Shalev-Shwartz. [Online]. Available: https://www.youtube.com/watch?v=ViGL0z1BULs. [Accessed 22 08 2022].

[39] BMW Group, "Safety Assessment Report," 2020.

[40] J. Yoshida, "EE Times," 29 04 2020. [Online]. Available: https://www.eetimes.com/unveiled-bmws- scalable-av-architecture/.

[41] M. Wagner, J. Ray, A. Kane and P. Koopman, "A safety architecture for autonomous vehicles". Patent EP3400676B1, 2017.

[42] T. Bijlsma, A. Buriachevskyi, A. Frigerio, Y. Fu, K. Goossens, A. O. Örs, P. J. van der Perk, A. Terechko and B. Vermeulen, "A Distributed Safety Mechanism using Middleware and Hypervisors for Autonomous Vehicles," in 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, 2020.

[43] Y. Fu, A. Terechko, J. Groote and A. Saberi, "A formally verified fail-operational safety concept

for automated driving," SAE Intl., pp. 7-21, 17 Jan 2022.

[44] Audi AG; BMW AG; Daimler AG; Porsche AG; VW AG, "Standardized E-GAS Monitoring Concept for Gasoline and Diesel Engine Control Units," 2013.

[45] M. Staron, Automotive Software Architectures: An Introduction, Springer Cham, 2021.

[46] ISO, "ISO 24089 Road vehicles - Software update engineering," 2023.

[47] IEC, "IEC 61508:2010 Functional safety of electrical/electronic/programmable safety-related systems," 2010.

[48] ISO, "ISO/SAE PAS 22736:2021 Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," 2021.

[49] BSI, "BSI PAS 1883:2020 Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) - Specification," 2020.

[50] B. Frömel, "Fault Tolerance (lecture notes)," [Online]. Available: https://ti.tuwien.ac.at/cps/teaching/courses/cpsesfvo/slides-ws14/04_ft.pdf#:~:text=A%20Fault%20Containment%20Unit%20%28FCU%29%20is%20a%20set,and%20is%20assumed%20to%20fail%20independently%20from%20other.

[51] Projekt Pegasus, "Projekt Pegasus," [Online]. Available: https://www.pegasusprojekt.de/en/pegasus- method.

[52] M. Scholtes, L. Westhofen, L. Turner, K. Lotto, M. Schuldes, H. Weber, N. Wagener, C. Neurohr, M. Bollmann, F. Körtke, J. Hiller, M. Hoss, J. Bock and L. Eckstein, "6-Layer Model for a Structured Description and Categorization of Urban Traffic and Environment," IEEE Access, vol. 9, pp. 59131 - 59147, 2021.

[53] D. Powell, "Failure Mode Assumptions and Assumption Coverage," in Predictably Dependable Computing Systems, Springer, 1995, pp. 123-140.

[54] F. Cristian, "Understanding Fault-Tolerant Distributed Systems," Communications of the ACM, vol. 34, pp. 56-78, 1993.

[55] N. Mohan, M. Törngren and V. Izosimov, "Challenges in architecting fully automated driving; With an emphasis on heavy commercial vehicles," in Workshop on Automotive Systems/Software Architectures, 2016.

[56] B. Littlewood and L. Strigini, "Validation of Ultrahigh Dependability for Software-Based Systems," Comm. ACM, vol. 36, pp. 69-80, 1993.

[57] H. Kopetz, Simplicity is Complex - Foundations of Cyber-Physical System Design, Springer Verlag, 2019. [58] H. Kopetz, "Emergence in Cyber-Physical System-of-Systems," in Cyber-Physical System-of-Systems,

[58] H. Kopetz, "Emergence in Cyber-Physical System-of-Systems," in Cyber-Physical System-of-Systems, Springer Verlag, 2016, pp. 73-96.

[59] A. Chou, J. Yang, B. Chelf, S. Hallem and D. Engler, "An Empirical Study of Operating System Errors," in Proceedings of the ACM SOPS 2001, 2001.

[60] Waymo, "Waymo Public Road Safety Performance Data," 2020.

[61] NHTSA, "Pre-Crash Scenario Typology for Crash Avoidance Research," 2007.

[62] American Psychological Association, "Multitasking Switching Cost," [Online]. Available: https://www.apa.org/research/action/multitask. [Accessed 22 May 2006].

[63] Automotive Electronics Council, "AEC-Q100:Rev-H Failure Mechanism Based Stress Test Qualification For Integrated Circuits," 2014.

[64] P. Koopman, "Simplified Proposal for Vehicle Automation Modes," 31 January 2022. [Online]. Available: https://safeautonomy.blogspot.com/2022/01/simplified-proposal-for-vehicle.html.

[65] California DMV, "Disengagement Reports," 2021. [Online]. Available: https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/disengagement- reports/. [Accessed 14 07 2023].

[66] C. Hanselaar, E. Silvas, A. Terechko and W. Heemels, "Detection and Mitigation of Functional Insufficiencies in Autonomous Vehicles: The Safety Shell," in 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), Macau, 2023.

[67] B. Kaiser, B. Monajemi Nejad, D. Kusche and H. Schulte, "Systematic design and validation of degradation cascades for safety-relevant systems," in The 2nd International Conference on Engineering Sciences and Technologies, 2017.

[68] S. Poledna, "Course: Dependable Computer Systems Part 5: Failure modes and models," 2007. [Online]. Available: https://ti.tuwien.ac.at/cps/teaching/courses/dependable_systems-ss08/dcs_slides/dcs-2007- p5.pdf. [Accessed 17 10 2023].

LIST OF ABBREVIATIONS

Abbreviation	Meaning
ACM	Association for Computing Machinery
AD	Automated/Autonomous Driving
ADI	Automated Driving Intelligence
ADS	Automated Driving System
AV	Automated Vehicle
BIST	Built-In Self-Test
CCDSS	Computer-Controlled Driving Sub-System (in channel-wise DCF architecture)
CCF	Common Cause Failure
CCI	Common Cause Initiator
CEHSS	Critical Event-Handling Sub-System (in channel-wise DCF architecture)
CSM	Controller Safety Mechanism (in DSM architecture)
DC	Diagnostic Coverage
DCF	Doer/Checker/Fallback
DDT	Dynamic Driving Task
DFA	Dependent Failure Analysis
DFI	Dependent Failure Initiator
DSM	Distributed Safety Mechanism
ECU	Electronic Control Unit
EOTI	Emergency Operation Time Interval
FCU	Fault-Containment Unit
FMEA	Failure Mode and Effects Analysis
FSM	Function Safety Monitor (in DSM architecture)
FTA	Fault Tree Analysis
FTDSS	Fault-Tolerant Decision Sub-System (in channel-wise DCF architecture)
FUN	Function (in the DSM architecture)
HARA	Hazard Analysis and Risk Assessment
HUD	Heads-Up Display
HW	Hardware
HWP	Highway Pilot
IEC	International Electrotechnical Commission
IEEE	Institute for Electrical and Electronics Engineers
IMU	Inertial measurement Unit
ISO	International Organization for Standardization

Abbreviation	Meaning
KPI	Key Performance Indicator
MRC	Minimal Risk Condition
MRM	Minimal Risk Maneuver
MSS	Monitoring Sub-System (in channel-wise DCF architecture)
MTTF	Mean Time to Failure
NHTSA	National Highway Traffic Safety Administration
ODD	Operational Design Domain
OEDR	Object and Event Detection and Response
OEM	Original Equipment Manufacturer
OS	Operating System
SAE	Society of Automotive Engineers
SaRA	Safety-Related Availability
SEooC	Safety Element out of Context
SEU	Single-Event Upset
SoC	System-on-Chip
SOTIF	Safety of the Intended Functionality
SUV	Sports Utility Vehicle
SW	Software
VRU	Vulnerable Road User
VSM	Vehicle Safety Mechanism (in the DSM architecture)
V2X	Vehicle-to-anything (vehicle, infrastructure)
WG	Working Group

APPENDICES

APPENDIX A: ODD OUTLINE OF REFERENCE AD USE CASE

Followed taxonomy

The HWP feature should only be active inside its defined Operational Design Domain (ODD), which is given by a set of conditions regarding its environment. We follow BSI PAS 1883:2020 [49], which mostly covers the same attributes as the formalism (6 layers) of Project Pegasus [51] [52]. The HWP feature should refuse to activate if these are not met and should, if these are no longer met, prompt the driver to take back control within a convenient time span and in the meantime ensure safety, e.g., by bringing the vehicle to a safe stop. The ability to execute an MRM must be maintained even outside the ODD.

Scenery

ZONES

Attribute	Sub-attribute (1)	Sub-attribute (2)	Capability
Zones	Geo-fenced areas		Yes, as designated by OEM
	Traffic management zones		No
	School zones		No
	Regions or states		Yes, as designated by OEM
	Interference zones	Dense foliage	Yes (not close to driving path)
		Tall buildings	Yes

DRIVABLE AREA

Attribute	Sub-attribute (1)	Sub-attribute (2)	Sub-attribute (3)	Capability
Drivable area	Drivable area type	Motorways (highways)		Yes, maximum 130 km/h
		Radial roads		No
		Distributor roads		No
		Minor roads		No
		Slip roads		No
		Parking		No
		Shared space		No
	Drivable area geometry	Horizontal plane	Straight roads	Yes
			Curves	Yes, maximum 1/100 m
		Transverse plane (cross-section)	Divided/undivided	Divided
			Pavement	No
			Barrier on the edge	
			Types of lanes together	
		Longitudinal plane (vertical)	Up-slope	Yes, maximum +4%
			Down-slope	Yes, maximum -4%
			Level plane	Yes
	Lane specification	Lane dimensions		Minimum 3.5 m
		Lane marking		Yes, in good condition
		Lane type	Bus lane	No (may be present, but must not be used during normal operation)
			Traffic lane	Yes
			Cycle lane	No
			Tram lane	No
			Emergency lane	No (may be present, but must not be used during normal operation)
			Other special purpose lane	Yes, carpool lanes
		Number of lanes		Yes, minimum 2 lanes per direction
		Direction of travel	Right-hand traffic	Yes
			Left-hand traffic	No

Attribute	Sub-attribute (1)	Sub-attribute (2)	Sub-attribute (3)	Capability
Drivable area	Drivable area signs	Information	Variable	Yes, full-time and temporary
			Uniform	Yes, full-time and temporary
		Regulatory	Variable	Yes, full-time and temporary
			Uniform	Yes, full-time and temporary
		Warning	Variable	Yes, full-time and temporary
			Uniform	Yes, full-time and temporary
	Drivable area edge	Line markers		Yes
		Shoulder (paved or gravel)		Yes
		Shoulder (grass)		Yes
		Solid barriers		Yes, obligatory on left side
		Temporary line markers		No
		None		No
	Drivable area surface	Surface type	Asphalt	Yes
			Concrete	Yes
			Cobblestone	No
			Gravel	No
			Granite setts	No
		Surface features	Cracks	Yes, minor only
			Potholes	No, not in significant density
			Ruts or swells	Yes, minor only
			Damage caused by weather	Yes, minor only
			Damage caused by traffic	Yes, minor only
		Induced conditions	Icy	No, not to a significant extent
			Flooded	No
			Mirage	Yes
			Snow	No
			Standing water	No
			Wet	Yes
			Contaminated	Yes, minor only

Additional assumptions:

- Changed road markings or reduced lane width are not supported.
- The speed limit is appropriate for the curve radius and slope of the road such that the entire stopping distance is visible without occlusions (in the absence of other vehicles).

JUNCTIONS

Attribute	Sub-attribute (1)	Sub-attribute (2)	Sub-attribute (3)	Capability
Junctions	Roundabout			No
	Intersection	T-junction		No
		Staggered		No
		Y-junction	On-ramp and off-ramp	No (except driving by)
			Other	No
		Crossroads		No
		Grade-separated	Interchange	No
			Other	No

ROAD STRUCTURES

Attribute	Sub-attribute (1)	Capability
Special structures	Automatic access control	No
	Bridges	Yes
	Pedestrian crossings	No
	Rail crossings	No
	Tunnels	Yes, with separate driving directions
	Toll plaza	No
Fixed road structures	Buildings	No
	Streetlights	Yes, but not required
	Street furniture	No
	Vegetation	No
Temporary road structures	Construction site detours	No
	Refuse collection	No
	Road works	No
	Road signage	No

ENVIRONMENTAL CONDITIONS

Attribute	Sub-attribute (1)	Sub-attribute (2)	Capability
Weather	Wind	Calm - fresh breeze (<10.7 m/s)	Yes
		Strong breeze (>10.7 m/s) - hurricane force	No
	Rainfall	Light rain (<2.5 mm/h)	Yes
		Moderate rain (>2.5 mm/h) - cloudburst	No
	Snowfall	Light snow (>1 km visibility)	Yes
		Moderate snow (<1 km visibility) - heavy snow	No
Particulates	Marine		No, not to significantly reduced visibility
	Mist and fog		No, not to significantly reduced visibility
	Sand and dust		No, not to significantly reduced visibility
	Smoke and pollution		No, not to significantly reduced visibility
	Volcanic ash		No, not to significantly reduced visibility
Illumination	Day		Yes
	Night or low-ambient		No, not to significantly reduced illumination
	Cloudiness	Clear - overcast	Yes
	Artificial illumination		Yes
Connectivity	Communication	V2V, V2I	Yes, at least intermittently
		Cellular	Yes, at least intermittently
		Satellite	No
		DSRC and ITS-G5	No
	Positioning	Galileo	Yes, at least intermittently
		GLONASS	Yes, at least intermittently
		GPS	Yes, at least intermittently

Additional assumptions:

- Not being warned of major road or traffic conditions is uncommon. We assume that the road layout is known ahead of time and that unexpectedly encountering challenging road or traffic conditions is uncommon as authorities are in charge of keeping the road in an acceptable state of repair and/or informing traffic participants (via signs, map data, and/or V2X) if this is not the case.
- HD Maps are available for all supported highway segments.

DYNAMIC ELEMENTS

Attribute	Sub-attribute (1)	Sub-attribute (2)	Capability
Traffic	Density of agents	Dense traffic (including stop & go)	Yes
		Free-flow traffic (including no lead vehicle)	Yes
	Volume of traffic		
	Flow rate		
	Agent type	Cars	Yes
		Buses and trucks	Yes
		Motorbikes	Yes
		VRUs (pedestrians, bicyclists)	Yes, to a very limited degree
		Animals	Yes, to a very limited degree
		Minor static obstacles (lost load, debris, etc.)	Yes
		Major static obstacles (lost load, trees, rocks, etc.)	Yes, to a very limited degree
		Special vehicles	Yes
Subject vehicle (ego vehicle)	Behavior capabilities	Ego vehicle speed	0-130 km/h
		Lane change	Yes
		Lane merge	Yes
	Vehicle	All sensors and actuators in working condition	Yes
		Sensor or actuator non-operational	No, except during MRM
		Superficial body damage	Yes
		Moderate - major body damage	No
		Door or window open	No
		Low fuel or charge level	No
	Passengers	Driver not in driver seat	No
		Unbelted passenger	No
		Driver asleep or incapacitated	No

Additional assumptions:

- All human traffic participants are aware that the highway is a restricted environment and act accordingly (responsibly).

APPENDIX B: DETAILED DESCRIPTION OF THE CHANNEL-WISE DCF ARCHITECTURE

This appendix provides additional details on the channel-wise Doer / Checker / Fallback conceptual system architecture as described in section 3.5.1.

STRUCTURAL DETAILS

The subsystems in the channel-wise DCF architecture differ in their assumptions, failure modes, and estimated necessary Mean Time to Failure (MTTF, see Table 6). The interfaces between subsystems are described in Table 7 and data types in Table 8.

TABLE 6: COMPARISON OF SUBSYSTEMS IN THE CHANNEL-WISE DCF CONCEPTUAL SYSTEM ARCHITECTURE.

Subsystem	Assumptions applying to failure mode	Failure mode assumption	Estimated necessary MTTF
Doer / CCDSS	• It is assumed that the vehicle is in operating condition. • It is assumed that the ODD of the item is respected.	Authentication-detectable Byzantine [68][26]	1000 h
Checker / MSS		Authentication-detectable Byzantine	1000 h
Fallback / CEHSS		Authentication-detectable Byzantine	1000 demands[27]
Redundancy management / FTDSS	• It is assumed that the HW of the FTDSS is correct[28]. • It is assumed that the SW of the FTDSS is correct.		

[26] Failure mode assumptions range (from most to least restrictive) [68]: fail-stop, crash, omission (fail-silent), performance, authentication-detectable Byzantine, Byzantine (fail-arbitrary). Byzantine failures are fully arbitrary and can appear different to different receivers. Authentication-detectable Byzantine failures have the restriction that they cannot spoof other systems' messages.

[27] The Fallback is not continuously in control of the vehicle and only acts on demand. Therefore, its MTTF is not given per time, but per demands.

[28] To achieve fault tolerance, the FTDSS consists of two instances: FTDSS A and FTDSS B.

TABLE 7: INTERFACES IN THE CHANNEL-WISE DCF CONCEPTUAL SYSTEM ARCHITECTURE. EXTERNAL INTERFACES OF THE AD INTELLIGENCE ARE SHADED ORANGE.

#	Sender	Receiver	Data type	Periodicity
1	Sensor System	CCDSS	SensorData	Sensor-dependent, event-driven (~10-100 ms)
2	Sensor System	MSS	SensorData	Sensor-dependent
3	Sensor System	CEHSS	SensorData	Sensor-dependent
4	Diagnostics System	CCDSS	SystemStatus	Main cycle, time-driven (~50 ms)
5	UI System	CCDSS	UserInput	Event-driven
6	CCDSS	MSS	ActuatorData	Main cycle
7	CCDSS	FTDSS A	ActuatorData	Main cycle
8	CCDSS	FTDSS B	ActuatorData	Main cycle
9	CCDSS	UI System	UserInformation	Main cycle
10	CCDSS	Sensor System	SensorControl	Main cycle
11	CCDSS	Diagnostics System	DiagnosticsData	Main cycle
12	MSS	FTDSS A	ValidationResults	Main cycle
13	MSS	FTDSS B	ValidationResults	Main cycle
14	MSS	CCDSS	ValidationResults	Main cycle
15	MSS	Diagnostics System	DiagnosticsData	Main cycle
16	CEHSS	FTDSS A	ActuatorData	Main cycle
17	CEHSS	FTDSS B	ActuatorData	Main cycle
18	CEHSS	Sensor System	SensorControl	Main cycle
19	CEHSS	Diagnostics System	DiagnosticsData	Main cycle
20	FTDSS A	Actuator System	ActuatorData	Main cycle
21	FTDSS A	MSS	ActuatorData	Main cycle
22	FTDSS A	MSS	ActuatorData	Main cycle
23	FTDSS A	Diagnostics System	DiagnosticsData	Main cycle
24	FTDSS B	Actuator System	ActuatorData	Main cycle
25	FTDSS B	MSS	ActuatorData	Main cycle
26	FTDSS B	MSS	ActuatorData	Main cycle
27	FTDSS B	Diagnostics System	DiagnosticsData	Main cycle

TABLE 8: DATA TYPES IN THE CHANNEL-WISE DCF CONCEPTUAL SYSTEM ARCHITECTURE.

#	Data type	Interfaces	Periodicity
1	SensorData	1, 2, 3	
2	SystemStatus	4	
3	UserInput	5	
4	ActuatorData	6, 7, 8, 16, 17, 20, 21, 22, 24, 25, 26	• Trajectory (timed sequence of waypoints for next ~1-3 sec) • Actuator setpoints (timed sequence of desired accelerations / decelerations and curvatures for next ~1-3 sec) • Priority of producing subsystem (CCDSS > CEHSS) • Incremental iteration counter
5	UserInformation	9	
6	SensorControl	10, 18	
7	DiagnosticsData	11, 15, 19, 23, 27	
8	ValidationResults	12, 13, 14	• CCDSS validation result (true / false) • CEHSS validation result (true / false) • Incremental iteration counter

BEHAVIORAL DETAILS

The rough function of the different subsystems in the channel-wise DCF architecture is described in pseudo-code in Table 9.

TABLE 9: PSEUDO-CODE FOR THE CHANNEL-WISE DCF CONCEPTUAL SYSTEM ARCHITECTURE WITH INTERFACE IDS IN PARENTHESES.

Subsystem	States	Behavior
CCDSS	• Nominal • Degraded • Internal fault	Receive sensor data (#1) Receive system status (#4) Receive validation results (#14) from last cycle IF (more than N out of last M CEHSS validation results are FALSE or not received) OR (system status NOK) # too frequent transient faults are treated as a permanent fault Go to degraded state IF (nominal state) Plan nominal trajectory ELIF (degraded state) Plan degraded trajectory IF (NOT internal fault state) Generate corresponding actuator setpoints Send trajectory and setpoints (#6, #7, #8) Report subsystem status (#11) IF (internal fault in execution or communication detected) Go to internal fault state (remain silent)

Subsystem	States	Behavior

MSS	• Functional
	• Internal fault
		Receive sensor data (#2)
		Generate environment model
		Receive actuator data for CCDSS (#6, #21, #25) and CEHSS (#22, #26)
		IF (functional state)
		    IF (all CCDSS sets are received) AND (all CCDSS sets are identical)
		        # this can catch some Byzantine faults
		        Run CCDSS evaluation (TRUE/FALSE)
		    ELSE
		        CCDSS evaluation is FALSE
		    IF (all CEHSS sets are received) AND (all CEHSS sets are identical)
		        Run CEHSS evaluation (TRUE/FALSE)
		    ELSE
		        CEHSS evaluation is FALSE
		    Send validation results (#12, #13)
		Report subsystem status (#15)
		IF (internal fault in execution or communication detected)
		    Go to internal fault state (remain silent)

CEHSS	• Functional • Internal fault	Receive sensor data (#3) IF (functional state) Plan degraded trajectory Generate corresponding actuator setpoints Send trajectory and setpoints (#16, #17) Report subsystem status (#19) IF (internal fault in execution or communication detected) Go to internal fault state (remain silent)

FTDSS A	• Functional
	• Internal fault
		Receive CCDSS actuator data (#7)
		Receive CEHSS actuator data (#16)
		Forward CCDSS actuator data (#21)
		Forward CEHSS actuator data (#22)
		Receive validation results (#12)
		IF (CCDSS set received)
		    IF (CEHSS set received)
		        IF (CCDSS set valid)
		            Select CCDSS set
		        ELSE
		            # transient faults can be masked
		            Select CEHSS set
		    ELSE
		        Select CCDSS set
		ELSE
		    IF (CEHSS set received)
		        Select CEHSS set
		Forward selected actuator data (#20)
		Report subsystem status (#23)
		IF (internal fault in execution or communication detected)
		    Go to internal fault state (remain silent)

FTDSS B	• Functional • Internal fault	Same as for FTDSS B, except using interfaces (#8, #13, #17, #24, #25, #26, #27) instead of (#7, #12, #16, #20, #21, #22, #23).

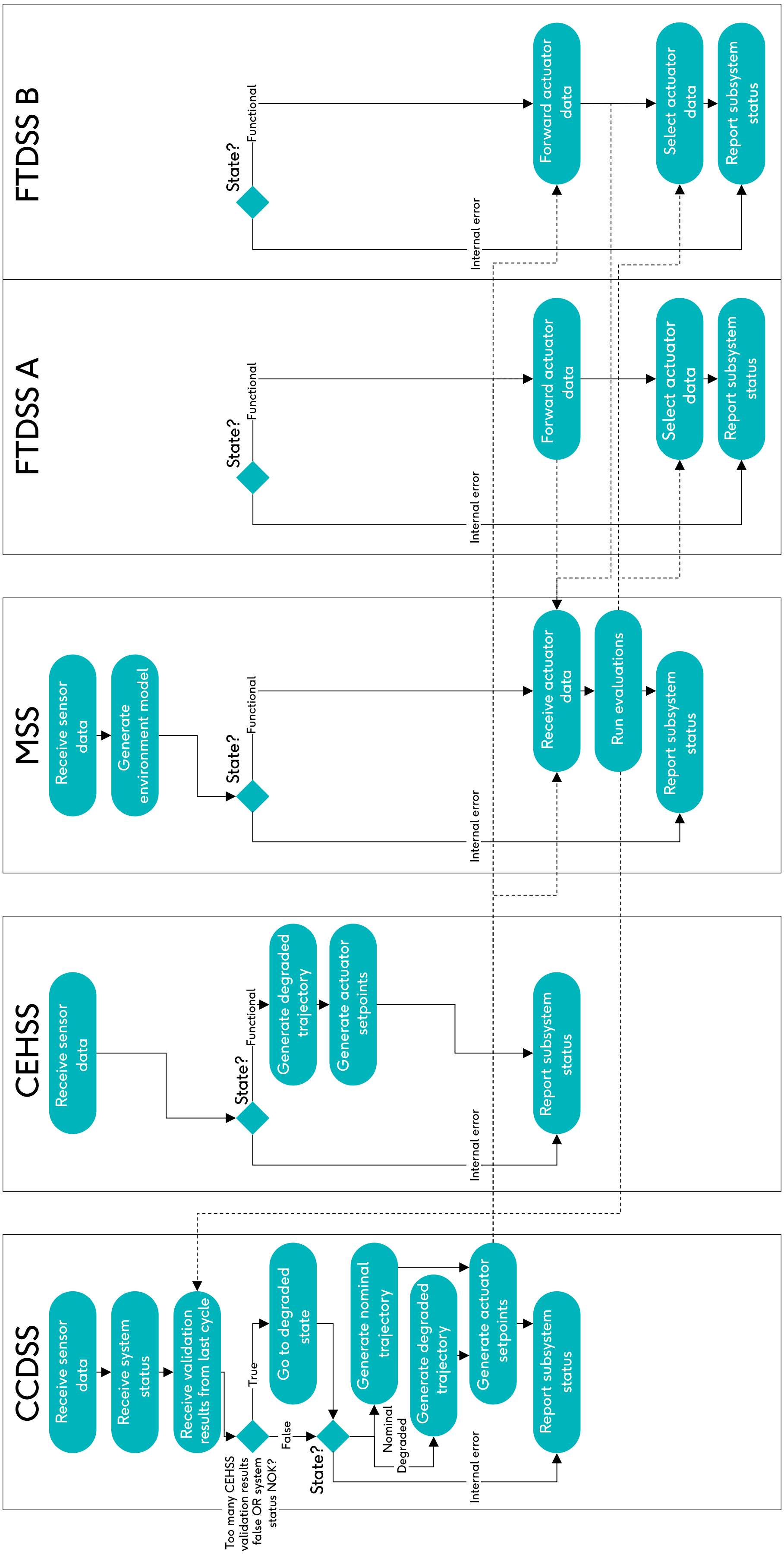


Figure 22: Activity diagram for the channel-wise Doer/Checker/Fallback conceptual system architecture.

APPENDIX C: SAMPLE ANALYSIS POINTS REGARDING DIFFERENT CONCEPTUAL ARCHITECTURE PATTERNS

This appendix provides additional details on the design principle D7: Mitigation of common-cause hazards, section 1.5.2.

The dimensions introduced in Figure 6 can be exemplified by conceptually applying them to a selection of architecture patterns. Functional complexity is indicated by the depth of the slice, while the implemented capabilities' coverage of an operational domain is indicated by the surface area. Holes are therefore representative of an absence of capability.

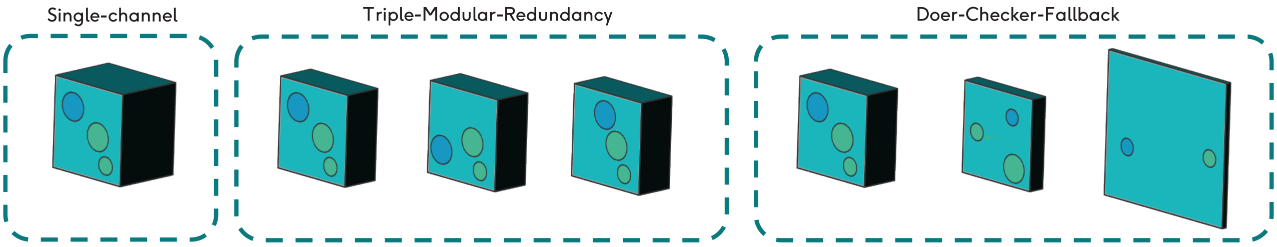


Figure 23: Channel dimensioning as per conceptual pattern of selected architecture candidates

Given the selected architecture patterns in Figure 23, the following relative observations can be made:

- Single-channel architectures can be expected to have significantly complex implementations to meet the functional requirements of advanced use cases.
  - Errors and output insufficiency within the implementation is not complemented or offset by the capability of another channel; a fallback capability is not available.
- TMR (triple-modular-redundancy) architectures typically offer redundancy of the same functionality, hence the equal depth of each slice and diverse location of errors to prevent unavailability of the function.
  - Nevertheless, the redundant functions may all contain the same output insufficiencies and therefore offer no prevention of common-cause functional insufficiency hazards.
  - Even the diverse implementation of the functionality aiming to achieve non-common cause output insufficiencies could struggle with inexact agreement when handling the individual channel outputs.
  - A fallback capability beyond the nominal functionality within the intended ODD is not available, as indicated by the equal areas of each slice.
- Doer-checker-fallback architectures typically propose the diverse implementation of a complex performance channel and its complementary checker channel, hence the unequal depth of the slices and diverse location of holes.
  - A fallback capability beyond the nominal functionality is offered by a basic channel capable of offering minimum-risk-maneuver in conditions beyond the ODD; the area of which could be considered analogous to something like the Target Operational Domain (TOD).

THE AUTONOMOUS