Multi-Objective Optimization for Safety-Related Available E/E Architectures Scoping Highly Automated Driving Vehicles

RICARDO GONZALEZ DE OLIVEIRA, University of Luxembourg and Robert Bosch GmbH
NICOLAS NAVET, University of Luxembourg and Cognifyer S.à.R.L.
ACHIM HENKEL, Robert Bosch GmbH

Megatrends such as Highly Automated Driving (HAD) (SAE ≥ Level 3), electrification, and connectivity are reshaping the automotive industry. Together with the new technologies, the business models will also evolve, opening up new possibilities and new fields of competition. To cope with the ongoing advances, new Electric/Electronic (E/E) architecture patterns are emerging in the sector, distributing the vehicle functions across several processing devices and enhancing the connectivity between them via Ethernet-based networks. Upcoming systems will demand Safety-Related Availability (SaRA) requirements in mixed-critical E/E architectures that challenge the concept of freedom from interference defined in ISO 26262. This work explores the concepts of SaRA system development according to ISO 26262, building a framework based on model-based systems engineering to evaluate feasible next-generation automotive E/E architecture designs with a multi-objective analysis. Additionally, we propose a pattern template for SaRA systems to automate the architecture synthesis. To illustrate the framework created, we evaluate a set of automotive E/E architectures synthesized to support mixed-critical vehicle features, including SaRA SAE Level-3 functions, considering the communication networks' performance as well as hardware and safety-related development costs. This work presents a methodology for original equipment manufacturers and Tier-1 suppliers that enables them to make the trade-offs arising in the design of E/E architectures based on quantified information.

CCS Concepts: • **Computer systems organization → Dependable and fault-tolerant systems and networks**; • **Computing methodologies → Model development and analysis;**

Additional Key Words and Phrases: Automated driving, model based systems engineering, design patterns, automotive, E/E architecture, safety, availability, time-sensitive networking, design space exploration

ACM Reference format:
Ricardo Gonzalez de Oliveira, Nicolas Navet, and Achim Henkel. 2023. Multi-Objective Optimization for Safety-Related Available E/E Architectures Scoping Highly Automated Driving Vehicles. *ACM Trans. Des. Autom. Electron. Syst.* 28, 3, Article 41 (March 2023), 37 pages.
https://doi.org/10.1145/3582004

Authors' addresses: R. Gonzalez de Oliveira, University of Luxembourg, Avenue de l'Université 2, Esch-sur-Alzette, Luxembourg and Wilhelm-Fein-Straße 6, Ludwigsburg, Germany; email: ricardo.gonzalezdeoliveira@de.bosch.com; N. Navet, University of Luxembourg, Avenue de l'Université 2, Esch-sur-Alzette, Luxembourg and Cognifyer S.à.R.L., Esch-sur-Alzette, Luxembourg; email: nicolas.navet@uni.lu; A. Henkel, Robert Bosch GmbH, Ludwigsburg, Germany; email: achim.henkel@de.bosch.com.

1084-4309/2023/03-ART41 $15.00
https://doi.org/10.1145/3582004

1 INTRODUCTION

1.1 Context of the Work

The mobility of tomorrow will be different. Megatrends such as autonomous driving, connectivity, electrification, and personalization [40, 53] are reshaping the automotive industry, leading to shifts in the business models of the mobility sector, and opening up new possibilities and competitions. Today's automotive **Electric/Electronic (E/E)** architectures do not offer the flexibility to support the shorter development and technology lifecycles demanded by the megatrends [30]. Today's vehicles can include up to 120 **Electrical Control Units (ECUs)** and 100 million lines of code, with the anticipation that **Software (SW)** will further increase in size to implement new functionalities [36].

New E/E architecture patterns are emerging in the automotive sector to address the flexibility and scalability requirements while maintaining the complexity under control cost-effectively. As it has been done in the aerospace domain with **Integrated Modular Avionics (IMA)** (see [57]), these new solutions are *integrated architectures* instead of *federated architectures*, relying on state-of-the-art technologies such as high-performance multi-core processors, system-on-chips, and high-speed Ethernet networks [52]. More precisely, such integrated automotive architectures are characterized by possessing several microprocessor (μP)-based **Vehicle Computers (VCs)** running cross-domain functions communicating through a high-speed Ethernet backbone. The VCs have more memory and computation power than the classical decentralized microcontroller (μC)-based ECUs. The communication backbone will support **Time-Sensitive Networking (TSN)** standards (see the work of Nasrallah et al. [41] for a survey) to guarantee the **Quality-of-Service (QoS)** requirements demanded by the vehicle function.

Nevertheless, developing such integrated architectures is a non-trivial task, as the VCs will host functionalities ranging from critical real-time functions pertaining to the vehicle's dynamics to non-critical interactive apps. Indeed, ISO 26262 [17] requires *freedom from interference*, which guarantees that mixed-criticality applications can coexist with no cascading failures jeopardizing the safety requirements. This work contributes to the development of systems with **Safety-Related Availability (SaRA)** requirements according to ISO 26262:2018 part 10, focusing on synthesizing automotive E/E integrated architectures hosting safety mixed-criticality applications including SAE Level-3 [48] functions.

1.2 Definition of the Problem

As explained in the previous section, new automotive E/E architectures are emerging, moving from federated to integrated designs. In previous work [23], we presented a set of functional E/E architecture patterns, distributing vehicle domain functions (e.g., chassis, powertrain, infotainment, driving assistance) to VCs. In particular, we highlight that **Original Equipment Manufacturers (OEMs)** tend to select one type of architecture pattern according to their vehicle portfolio. The three major types of architecture patterns are *domain concentration*, *domain fusion*, and *vehicle centralized*, with the latter one being an instance of the integrated architectures. The design of integrated architectures increasingly follows the **Service-Oriented Architecture (SOA)** paradigm, where the SW is decoupled from the **Hardware (HW)**, providing more freedom in the SW functions allocation. Further, having Ethernet as the network backbone gives flexibility with respect to the network topology, like *ring*, *star*, *tree*, or a mix of them. Figure 1 depicts a potential automotive integrated architecture, separating the architecture into layers according to the processing devices and summarizing possible communication network topologies and technologies. As can be seen, there are several design options for network topologies and the deployment of SW functions. The objective is to identify the solutions that meet all system QoS requirements, such as performance and dependability, while being compatible with the automotive cost pressure.

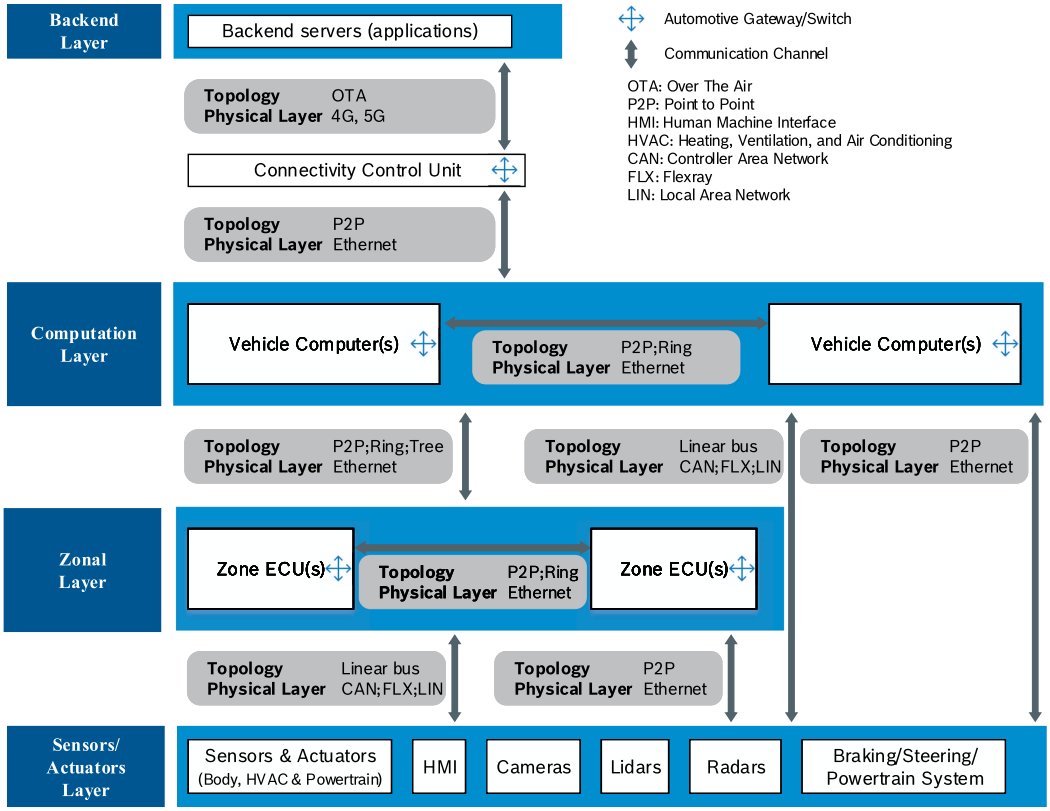


Fig. 1. Example of an automotive integrated E/E architecture pattern, showing the structure layers and possible communication network topologies and technologies.

Concerning the QoS requirements, passenger vehicles are safety-critical systems with a set of guarantees to meet. Some automotive E/E functions can achieve a safe state after a system failure by turning off the functionality. However, some applications have a fail-operational safe state, meaning that simply turning off the functionality could lead to a hazardous event. Examples are steer-by-wire systems or SAE $\geq$ Level-3 functions. For those systems with SaRA requirements, meaning that only turning them off is not sufficient, the 2018 version of ISO 26262 in part 10 provides development guidance, specifying safety goals with SaRA requirements.

Designing automotive integrated E/E architectures capable of performing mixed-safety functions with SaRA requirements involves making design choices on the E/E architecture topology, network protocols, and function allocation. In this work, we approach the problem as a multi-objective optimization analysis to identify solutions that provide meaningful performance versus cost trade-offs.

Precisely, this work explores the following research questions:

RQ1: How can we automate the **Design Space Exploration (DSE)** of integrated E/E automotive architectures supporting mixed-safety criticality functions with SaRA requirements (e.g., SAE $\geq$ Level 3) according to ISO 26262:2018 part 10, chapter 12?

RQ2: Can we enhance the design patterns in the literature to meet fail-operational requirements according to ISO SaRA guidelines, as required for next-generation automotive platforms?

RQ3: How should meta-models be augmented to SaRA design patterns and integrated E/E architectures? And how should it be structured so as to facilitate multi-objective DSE?

1.3 Limitation of Existing Solutions

Model-Based Systems Engineering (MBSE) techniques enhanced by design patterns are the state of the art to conceive and synthesize feasible automotive E/E architectures (see elsewhere [32, 59] and Section 8 for a review of related works). Commercial off-the-shelf tools (e.g., PREEvision, Siemens Capital, Enterprise Architects) using languages like SysML can support the design at different stages of the vehicle development, possibly with quantified metrics for evaluation. However, the user needs to manually specify the architecture details, with no support from design exploration. Some exploration capabilities are offered by the research tool AutoFOCUS3 [50], with the aid of plugins, to analyze the design and implementation alternatives (e.g., design patterns, task allocation). Nevertheless, to the best of our knowledge, no MBSE solution is capable of synthesizing SaRA automotive integrated E/E architectures, including an analysis to verify the timing QoS requirements.

Some works [11, 56] explore how to meet the QoS guarantees in automotive E/E architectures relying on service-oriented communications over Ethernet TSN protocols. Except for one experiment in the work of Creighton et al. [11], both studies evaluate the network performance on manually constructed architecture designs. However, as done in this article, these works could be enhanced with the use of MBSE techniques and design patterns to extend the search space and further automate the design of Ethernet-based E/E architectures.

It should be noted that the two categories of works cited previously can complement each other, meaning that DSE can be tailored to explore integrated automotive E/E architectures with SaRA and timing QoS requirements.

1.4 Contributions of the Article

This article presents a framework that enables the creation and multi-objective evaluation of integrated automotive E/E architectures with SaRA requirements in the early phases of the vehicle development process. The goal is to assist OEMs in quantifying the impact of architectural choices on the cost versus performance trade-off. This work provides the following main contributions:

(1) An MBSE framework based on viewpoints and model transformation that can be integrated into early automotive system development processes to generate fail-operational automotive E/E architectures.
(2) The definition of meta-models to describe integrated automotive HW topologies and fail-operational functional cause-effect chains. The former enables the exploration of zone-based layered architectures, including computation, zone, and sensor/actuator layers. The latter includes a domain-specific modeling language defining constraints and objectives, enabling the synthesis (i.e., the mapping of SW tasks to processors) with SaRA requirements.
(3) The elaboration of a design pattern template for SaRA topologies integrated into the MBSE framework to generate fail-operational architectures.
(4) The formal definition of the **Multi-Objective Analysis (MOA)** and exploration problem, as well as a set of techniques to solve it. Precisely, the multi-objective problem encompasses the following attributes: fail-operational E/E architecture validity, network performance, and costs (i.e., wiring harness, processors, and safety development costs).
(5) An evaluation of the approach on five different fail-operational integrated automotive E/E architectures varying in terms of HW topology and vehicle domains distribution, supporting a set of mixed-safety criticality automotive features including SAE Level-3 functions.

1.5 Organization of the Article

The rest of this article is structured as follows. Section 2 introduces the concepts of SaRA requirements according to ISO 26262 and shows the state of the art for automotive fail-operational E/E architectures. Section 3 presents the concepts and assumptions for integrated automotive E/E architectures and the approach to depict fail-operational design patterns. Section 4 presents the methodology for the system modeling using viewpoints, and Section 5 defines the design-space exploration with the MOA evaluation. Section 6 depicts the experimental setup to demonstrate the framework capabilities using as a use case a set of integrated E/E architectures supporting mixed-critical tasks, including SAE Level-3 functions with SaRA requirements. Section 7 details the results of the experiments through MOA. Section 8 recaps related works about the evolution of automotive E/E architectures, automotive safety standards and their relation to autonomous driving, modeling and design patterns for cyber-physical systems, and architecture exploration in a multi-objective optimization framework. Finally, Section 9 concludes and identifies future research directions.

2 AUTOMOTIVE SAFETY-RELATED AVAILABLE ARCHITECTURES

ISO 26262 defines a *safe state* as an operating mode without "unreasonable risk levels" in case of a system failure. For many E/E systems, the safe state can be achieved by turning off the application. For example, an SAE Level-2 **Lane-Keeping Assistance (LKA)** function will be shut off in case of malfunction, and the driver will take over the full control the vehicle. However, in some cases, the **Hazard Analysis and Risk Assessment (HARA)** shows that just turning off a certain functionality could lead to a hazard incident (e.g., steer-by-wire or SAE $\geq$ Level-3 functions), requiring additional SaRA requirements. This section provides details about the ISO 26262 approach for SaRA systems and architectural strategies for the implementation.

2.1 SaRA Requirements According to ISO 26262

The latest version of ISO 26262, published in 2018, gives guidance for developing automotive E/E systems with SaRA requirements in part 10, chapter 12. This chapter is supplementary content compared to the previous ISO 26262 version from 2011, focusing mainly on fail-safe systems, meaning that in case of failure, the system deactivates itself. This work will consider the latest version of the standard.

The standard notes that there are various measures to guarantee sufficient availability, including fault tolerance, fault avoidance, and fault forecasting. The term *fault tolerance* is related to fail-operational systems, meaning that after one or more specified faults, the system can still deliver the intended functionality for a given time. *Fault avoidance* is related to measures to reduce the likelihood of a fault, and *fault forecasting* is the ability to predict a fault before it can trigger a failure. Chapter 10 of the standard concentrates on fault tolerance, giving recommendations for the specification in the concept phase (i.e., early functional safety analysis), considerations for the availability during the HW design phase, and briefly covering SW fault avoidance and tolerance during the SW development phase.

According to the norm, evaluating whether the loss of the availability of an item's functionality can trigger a dangerous event or not depends on the vehicle operating state. Part 1 of the ISO defines an *item* as a system or combination of systems that execute a function at the vehicle level, and the *vehicle operating state* indicates the currently provided performance for a specified functionality. That means, for example, that the loss of a SAE Level-3 function during a high-speed driving phase will have a different HARA compared to a low-speed driving maneuver. Alternatively, it will have no safety issue if the sudden loss of functionality occurs when the system is not

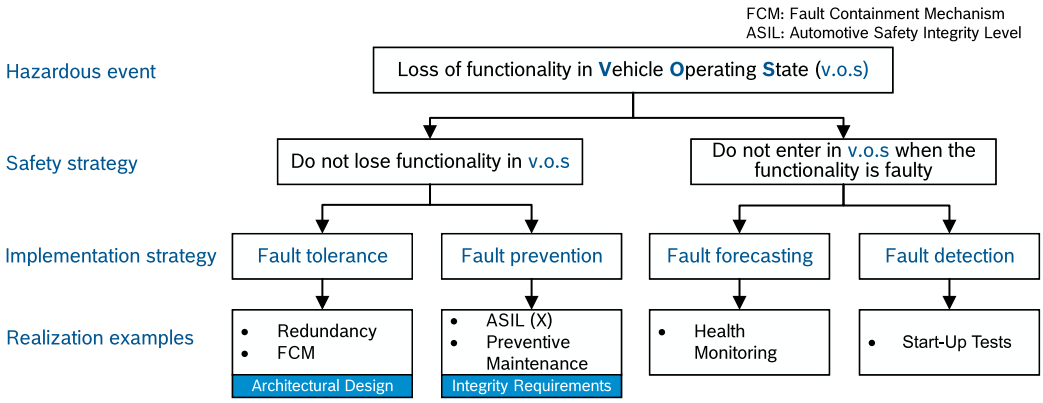


Fig. 2. General topics to be considered during the concept phase for fault-tolerant systems.

in operation. As a summary, Figure 2 illustrates the general topics to be considered during the concept phase to specify fault tolerance items and some realization examples. The picture separates the safety strategy for fault-tolerant systems into two groups: "do not lose functionality in vehicle operation state" and "do not enter vehicle operation state when the functionality is faulty." The former has two implementation strategies (fault tolerance and fault prevention), whereas the latter has two implementation strategies (fault forecasting and fault detection). This work focuses on fault tolerance and fault prevention implementation strategies—that is, designing automotive E/E architectures capable of not losing given functionalities during specific vehicle operation states.

Implementation strategies related to fault tolerance are associated with the architectural design—for example, having a redundant channel[1] to perform a function after a failure. Fault prevention is related to the integrity requirements of the ISO, where the HARA of a given item will have a set of safety goals with a specific **Automotive Safety Integrity Level (ASIL)** related. The norm specifies safety goals as top-level safety requirements (e.g., avoid insufficient brake torque), and the ASIL as the requirements and safety measures to apply for avoiding an unreasonable risk, with D representing the most stringent and A the least stringent level.

The creation and evaluation of the SaRA requirements come from the interaction with other items—that is, the system architecture, the fault tolerance measures, and the fault reaction operation and prevention of hazardous events. After a fault occurs, the system shall react inside an allowable time span to enter an emergency operation state or to a safe state, with a clear definition of the new functions to be executed and their associated performance constraints. The new vehicle operation state requires a re-evaluation of the HARA, possibly leading to new safety goals and ASIL requirements.

As an example of a fault-tolerant item with SaRA requirements, here we consider a steer-by-wire system with the safety goal of preventing a total loss of the steering capabilities with an ASIL D requirement. After a failure, the system shall move to a safe state that consists of a restricted maximal vehicle speed reached within a given time span. The new vehicle operational state has ASIL A because of the speed restriction. The system architecture consists of two independent channels A and B, where A provides the nominal function and B is the backup system. According to the ISO, the architecture described has the following safety requirements regarding systematic faults, random HW faults, and restriction of vehicle operational state after failure:

[1]For the safety domain, the concept of channel represents a system or set of systems that will perform a functionality. For example, if a functionality must have fail-operational capabilities, it can be constructed with two redundant channels.

Systematic faults:
- Channels A and B have to fulfill ASIL D level.

Random HW faults:
- The combination of the channels has to fulfill ASIL D level.[2]
- Channel A has to fulfill ASIL D level for the nominal functioning mode.
- Channel B can fulfill ASIL A level to match the restricted vehicle operational state safety goal.

Restriction of vehicle operational state:
- The item responsible for restricting the vehicle speed has to fulfill ASIL D level.

We note that the usage of ASIL decomposition on fault-tolerant items could lead to wrong designs if not correctly applied, and this has been a common misunderstanding of the norm since the first issue, as observed in the work of D'Ambrosio and Debouk [12]. The norm states that the ASIL decomposition applies to redundant elements of a fault-tolerant item, where the decomposed components have at least an ASIL greater than or equal to the HARA after the loss of redundancy. The decomposition is specific for the redundant channels considering the new vehicle operating state HARA after the fault.

Moreover, ISO 26262 [17] (part 10, chapter 5, note 2) observes that the usage of homogeneous redundancy (i.e., duplication of identical HW or SW components for the redundant channels) does not address systematic failures (i.e., failures that occur in a deterministic manner). Therefore, it is necessary to implement heterogeneous redundancy to protect against common sources of failures.

2.2 Fail-Operational Safety-Related Automotive Architectures

With increasingly automated driving functions, the automotive industry needs to implement fail-operational architectures that match the SaRA requirements. Even high-end cars do not have such architectures because the SAE Level-2 systems in use today can be fail passive, relying on the driver to take over the control in case of faults. However, that will change when SAE $\geq$ Level-3 systems are introduced, as they give the responsibility for fault tolerance to the system itself. An essential factor for the development of fault-tolerant systems is the number of faults that the system must tolerate while being able to reach a safe state. This redundancy degree influences the number of redundant elements the architecture should have. For example, an SAE Level 3 with degree 1 needs one redundant channel. In case of fault, the remaining channel must perform a safe maneuver within a given time span after the first fault to reach a safe state. Another example could be a degree 2 SAE Level-4 system with two redundant channels triggering a limp-home mode after the first fault and moving to a safe state after the second fault.

The automotive industry can get design inspiration from other sectors to develop fail-operational architectures. However, adapting solutions from one sector to another is not simple, as different industries have different cost pressure, production volumes, hazard consequences, and other specificities. The work of Schnellbach [49] provides an analysis of the state of the art for fail-operational architectures in the avionics, railway, agricultural, and industrial sectors. Of all sectors, the avionics industry has the most experience. The fail-safe behavior is usually not an option, and, for instance, airborne avionic functions need to be fail-operational. Examples of redundancy concepts from avionics are *duo-duplex* used by Airbus [26], *quadruplex* used for the NASA Space Shuttle [7], and *triple-triple redundancy* architecture used for the Boeing 777 [58].

The automotive industry has higher financial pressure and production volumes than the aerospace domain, making it impossible to re-use the solutions proven in use over decades in the aerospace domain. OEMs will have to find a trade-off between costs and safety performance

[2]The combination of the channels have, for example, 99% or greater HW fault coverage.

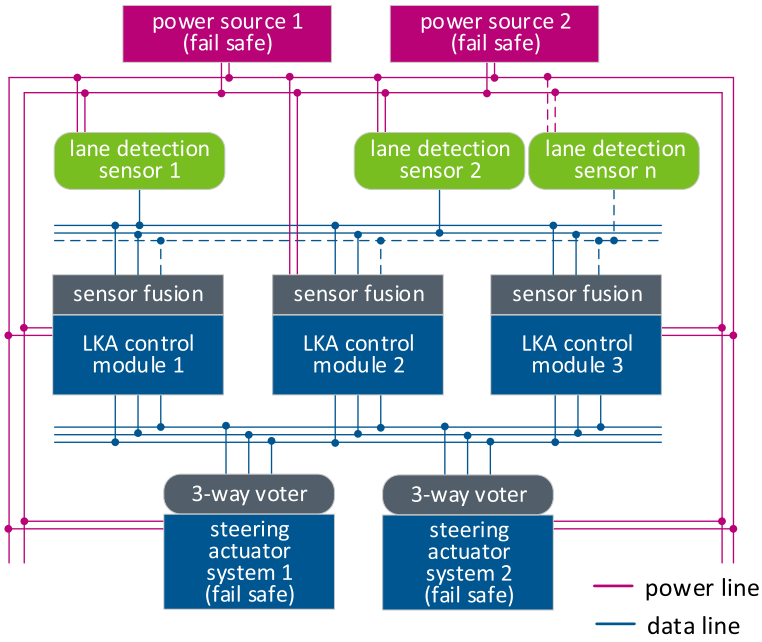


Fig. 3. Example of a fail-operational LKA system, showing the architectural components, the power lines, and data lines connections.

for next-generation vehicles, offering the desired vehicle features at a price that the end consumers can afford. Research and books already propose fail-operational architectures for specific automotive systems, such as a design concept for fail-operational LKA in the work of Becker et al. [6] or a similar concept for fail-operation steer-by-wire in another work by Becker et al. [5]. Figure 3 depicts an architecture from their work [6] that implements a combination of **Triple Modular Redundancy (TMR)** for the LKA control modules and a duplex for the steering actuator system. The power supply and the actuator systems have a fail-safe behavior with a safe containment mechanism, turning off after a fault and not disturbing the other functional channels. The TMR has the objective to mask faults on the LKA control channels, assuming that at least two channels have similar outputs according to the voter logic at any given time. This architecture has degree 1, requiring the system to move to a safe state after a single system fault.

It is relevant to mention that a way to avoid additional costs for fail-operational systems is to use another vehicle item possessing the capability to perform the redundancy. For example, in the case of a fault in the steering system, it is possible to use the brakes or an all-wheel electric drive to apply a specific yaw rate to the vehicle within the limit of the physic dynamics. For a specific vehicle operating state, a system can be a redundant channel for another system. Since this approach is strongly related to the vehicle dynamics and vehicle operating state, this work will not consider such fallback strategies. Instead, it will explore the deployment of redundant channels using duplication and the impacts on the vehicle's E/E architecture. The following section will detail the template used in this work for integrated automotive E/E architectures and the definition of design patterns for fail-operational systems.

3 INTEGRATED AUTOMOTIVE E/E ARCHITECTURES AND SARA TOPOLOGIES

In this section, we propose the integrated vehicle's E/E architecture template to design fail-tolerant automotive systems. Section 3.1 focuses on the HW template presenting the resources' description

and the communication topology. Section 3.2 proposes design patterns for fail-operational topologies and the method to integrate them into the system model.

3.1 Integrated Vehicle's E/E Architecture Template and Assumptions

For the generation of integrated E/E architectures, we consider the template presented in Figure 1 adapted from a Bosch internal analysis. The figure depicts a layered automotive architecture, separating the design space into two dimensions: HW resources and network topology.

3.1.1 HW Resources. The HW resources are the processing and data forwarding units. According to the components' role in the E/E architecture, the template separates the HW resources into four layers:

- *Sensor/Actuators layer*: This layer represents the low-level embedded elements from the E/E architecture. It contains the ECUs, sensors, and actuators. In this work, they are either connected to an automotive switched Ethernet network or to a **Controller Area Network Flexible Data-Rate (CAN-FD)** bus.
- *Zonal layer*: This layer represents the automotive gateways in a specific physical position in the vehicle (e.g., front, cabin, or rear compartment). The zone gateways operate as a multi-protocol hub capable of transferring data across different network technologies. In this work, each zone gateway contains an Ethernet switch and a CAN-FD end-node embedded in the gateway.
- *Computation layer*: This layer represents the VCs, which are multi-functional domain integration platforms for most of them. Each VC can host multi-criticality safety tasks following ISO 26262 guidelines to ensure freedom from interference. Further, each VC possesses an automotive Ethernet switch built in and a middleware to support the SW services.
- *Backend layer*: This layer represents the services that run on the cloud. This work only considers the *connectivity control unit* providing the communication interface to the cloud. The cloud architecture and its performance are outside the scope of this study.

3.1.2 Network Topology. The network topology describes how the HW resources are connected: layout of the networks and technologies used (protocols, data rates, etc.). This work considers automotive Ethernet links and CAN-FD buses for the connectivity.

3.2 Design Patterns for SaRA Architectures

Design patterns in the SW domain describe commonly used algorithms and architectural structures to solve general design problems in particular contexts.[3] In an effort to catalog and improve the patterns' re-usability for safety-critical embedded systems, many works have been devoted to enhance the description of the design patterns, such as in a formal manner, as well as their support in industrial tools through plugins (see elsewhere [23, 41] and Section 6 for a review of related works). As an example of design patterns. Figure 4 shows a duplex redundancy pattern from the work of Armoush [1] with the flow of information (both data and control flow) between the components and the safety mechanisms (i.e., comparator and switch). The data processing and sensor blocks have two channels to tolerate faults in this pattern. Still, the comparator, the switch, and the actuator are single points of failure, making it non-fault-tolerant if any of those elements fails. A simple solution could be to have additional channels for those elements. However, in case the system designer needs a chain of patterns, like the LKA architecture from Figure 3 with a TMR for

[3]In this work, we use a broad definition of design patterns that encompasses architectural patterns.

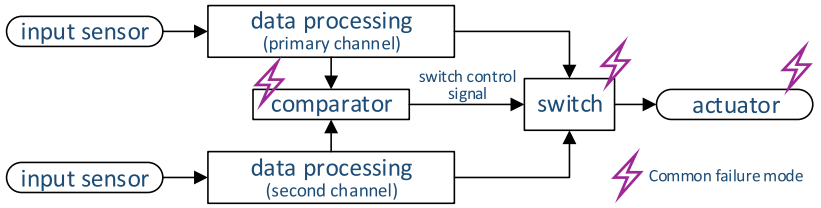


Fig. 4. Duplex pattern and potential elements with common failure mode.

Table 1. Design Patterns Template for Safety-Critical Embedded Systems [32]

Attribute	Meaning
Name	Name of the design pattern
Intent	Purpose of the design pattern
Motivation	Motivation to use the design pattern
Application	Situations to apply the design pattern
Structure	Representation of the design pattern and their connections
Participants	Description for each structure elements
Collaborations	Explanation on how the design pattern elements collaborate to achieve intent
Consequences	Design pattern effects on the system's functional aspects
Implementation	Guidelines for the design pattern development
Implications	Design pattern effects on the system's non-functional aspects
Usage classification	Category of the design pattern
Example	Use case example
Related patterns	Synergy to other design patterns or derivations

the control modules connecting to a duplex for the actuators, additional information beyond the pattern template, such as channels interconnections, is necessary to couple the patterns correctly.

Table 1, proposed in the work of Khalil [32], represents a possible design pattern structure template for safety-critical embedded systems. Some attributes in the table are highly relevant for the design of fault-tolerant systems. The *structure* and *participants* attributes describe how the channels are designed and connected. The *collaboration* and *consequences* attributes explain how the design pattern elements interact to reach a safe state. The *implications* attribute defines the impact of the design pattern on the non-functional requirements (e.g., reliability, availability, execution time, and costs) considered in the multi-objective DSE. Finally, the *related patterns* have importance in identifying possible pattern chains, as explained before.

In automotive systems, tasks with diverse dependability requirements need to exchange data—for example, a set of heterogeneous fail-safe sensors sending data to a fault-tolerant computation system that commands a fault-tolerant actuator. To define the intraconnection between elements of a system, and interconnections between systems, we introduce the concept of *outer layer* and *inner layer* pattern, as illustrated in Figure 5. The outer layer pattern works as a multiplexer defining how many channels are input and how many channels are output. This pattern guides the interconnections to the following pattern, requiring that the number of output channels from the first pattern matches the number of input channels of the following pattern. The outer layer pattern uses a black-box concept and fits in the *related patterns* attribute from Table 1. The inner layer depicts the intraconnections of the design pattern, showing the participants of the pattern and the data exchanged and processed.

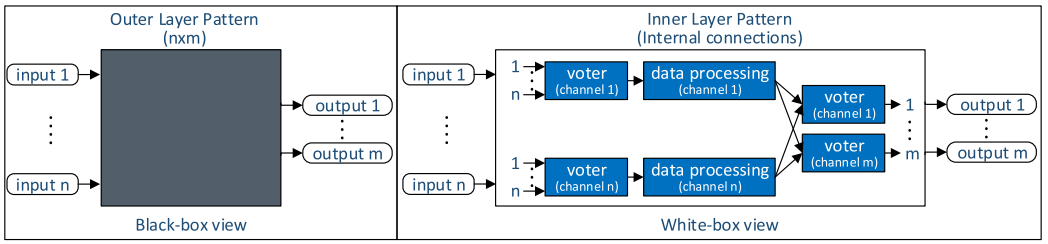


Fig. 5. Example of the outer layer and inner layer description approach to describe the intra- and interconnections in design pattern chains for fail-operational architectures.

Each element of the inner layer has a channel tag that can support the deployment of the components to redundant resources. The inner layer uses a white-box approach showing the design details, which is related to the *structure*, and *implication* attributes from Table 1.

3.3 Diverse Redundancy Requirement

As described in Section 2.1, ISO 26262 stresses the need for heterogeneous redundancy, also called *diverse redundancy*, to avoid common sources of failures on redundant channels. The design patterns for SaRA architectures presented in Section 3.2 give guidelines for the system designers to decide where to implement heterogeneous redundancy. For example, the same functions with different channel tags will require diverse HW and SW redundancy. This typically implies the need for different HW suppliers, and different teams developing the same SW components.

However, implementing diverse redundancy increases the architecture costs, as it is necessary to develop and validate the different solutions independently. In this work, we do not consider the additional costs of diversity, and the costs are calculated as if channels were homogeneous. This assumption could be lifted with a more precise cost model.

In the rest of the article, we assume that when the architect applies a design pattern on the functional architecture, as explained in Sections 3.2 and 4, diversity is taken into consideration. The following section presents the integrated E/E architecture system modeling used in this work, formalizing meta-models for the systems and fault-tolerant design patterns.

4 SYSTEM MODELING AND VIEWPOINTS

For the system modeling, we use the concept of model viewpoints according to the SPES2020 methodology [46] to describe the E/E architecture's structure and specifications. Each viewpoint represents implementation-specific aspects regarding the SW or HW architecture. The viewpoint elements are modeled in SysML. As is classically done in SPES2020, we use the functional, logical, and technical viewpoints to describe the E/E architecture, with the requirements included in the functional viewpoint. Each viewpoint and the meta-models are introduced in more detail in the following sections. To support the explanation, we use an SAE Level-3 highway-pilot feature[4] as an example.

4.1 Functional Viewpoint

The functional viewpoint represents the system's functional dependencies and feature interactions by functional building blocks. We use the functional viewpoint to describe the information flow

[4]In this work, a feature is defined as a characteristic behavior of a system that the customer experiences. A feature is typically a composition of functions and logical interactions between those functions described by the viewpoints modeling.

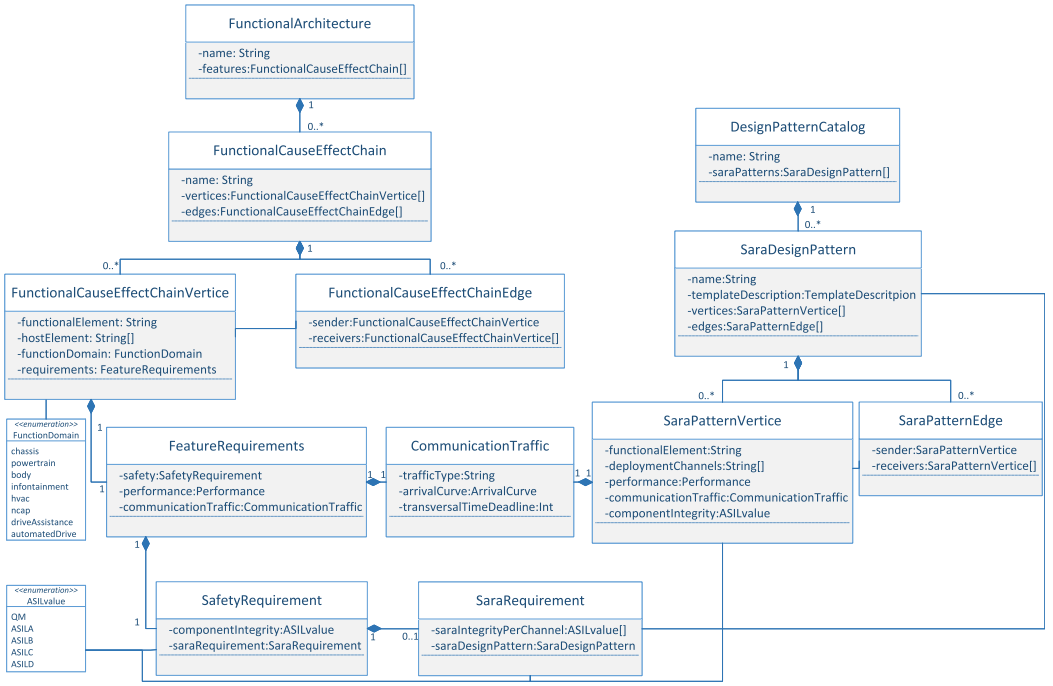


Fig. 6. Meta-model of the functional architecture with its classes, classes' attributes, and relations between classes.

of the vehicle features, using meta-models to assemble the feature cause-effect chains[5] and performance requirements on functional blocks.

4.1.1 Meta-Model. Figure 6 depicts the meta-model for the functional viewpoint, also called the *functional architecture*. The functional architecture consists of 0..* *FunctionalCauseEffectChain*, where each cause-effect chain represents a system feature. A feature cause-effect chain is modeled by 0..* *FunctionalCauseEffectChainVertice* and 0..* *FunctionalCauseEffectChainEdge*. A vertice represents a data processing element such as a sensing block. The edge represents the connections between vertices, having one vertice as sender and an array of vertices as receivers. Depending on the granularity of the model, a vertice can have an array of possible host elements (e.g., front camera, nomadic) and a related vehicle functional domain (e.g., powertrain, chassis). Some nodes will have the host element as *nomadic*. We use the term *nomadic* for functions that can be hosted in different integration platforms.[6] Those properties later constrain the system synthesis—that is, the deployment of the functions to the appropriate HW hosts. Every vertice, which is a functional block, has a *FeatureRequirements* attribute that represents the performance, communication, and safety requirements. The performance requirement defines the vertices computation need (e.g., processing resources, memory). The communication requirement describes the data traffic that the vertice will send. The *CommunicationTraffic* class defines the type of traffic (e.g., best effort,

[5]A cause-effect can be understood as the path from an input (e.g., sensor) across the required SW and HW elements until the desired output (e.g., actuators) (see [24]).
[6]In this work, the nomadic functions are assumed to be statically deployed at design time, as done in industry today. Two works [30, 34] explore the use of nomadic functions in a dynamic runtime environment, fully utilizing the possibilities of the SOA paradigm.

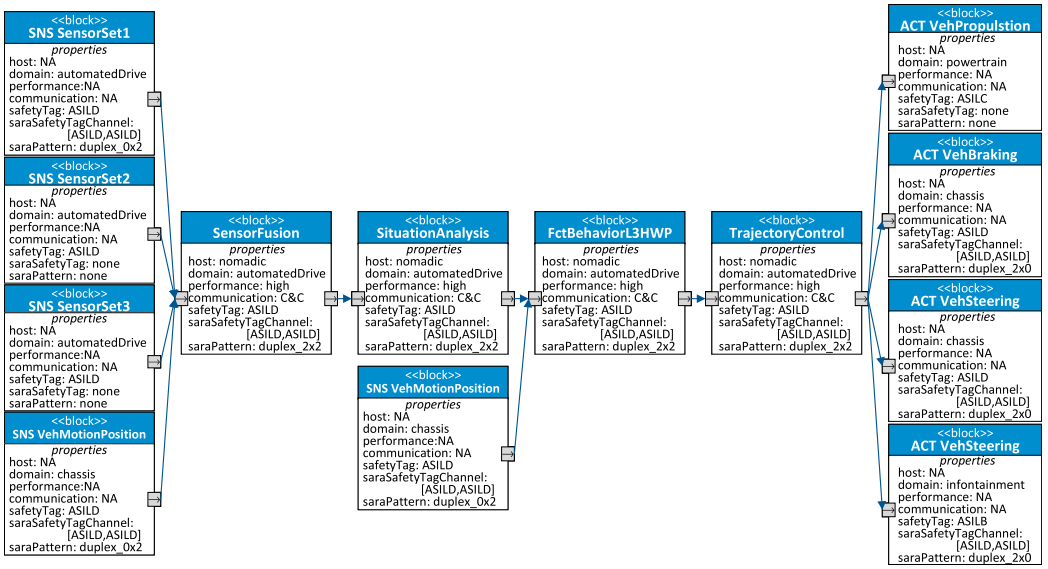


Fig. 7. Functional architecture example. Representation of an SAE Level-3 highway-pilot feature, describing the cause-effect chain using the functional viewpoint meta-model.

real time) with the related traffic pattern, called the *arrival curve*, and traversal time deadline constraint. The arrival curve defines all the traffic flow properties (e.g., packet type, period, payload size) that will be taken as input to evaluate the network performance. The *SafetyRequirement* class includes the vertice's ASIL and the SaRA requirements if necessary. The *SaraRequirement* class describes the design pattern to be applied with the *saraIntegrityPerChannel* attribute, giving an array with the ASIL requirement for each of the design pattern channels. The class *SaraDesignPattern* defines the template presented in Section 3.2, including the description of the pattern and the structure in vertices and edge elements. To support the system synthesis, the *SaraPatternVertice* class provides the deployment channels, performance, communication traffic specifications for the pattern's vertices, and the ASIL related to each pattern vertices according to the SaRA requirements. The class *DesignPatternCatalog* represents the library of patterns with 0..* *SaraDesignPattern* to be used by the system designers.

4.1.2 Example. Figure 7 depicts the functional architecture for an SAE Level-3 highway-pilot feature, as an example of a functional cause-effect described using the meta-model in Figure 6. The feature controls the car in a highway driving scenario, and a fallback system will perform a safety maneuver in case of a failure. The safety maneuver will drive the car to the rightmost lane of the highway in a given time, reaching the safe state when the car stops and deactivating the system until repair. The feature uses a set of vehicle functions with diverse ASIL requirements to perform the maneuvers. For example, the braking and steering systems have an ASIL D requirement, and the powertrain system has an ASIL C requirement. Some systems will have SaRA requirements, where, in this example, they have the same ASIL value for the first and second channel. Figure 7 uses a SysML Block Definition Diagram (BDD) to illustrate the cause-effect chain vertices as blocks and the edges as arrows. Each block possesses properties as per the functional architecture meta-model. When moving to the logical viewpoint, some blocks need to undergo a model transformation, increasing the granularity of the block by adding new elements and connections. In the example of

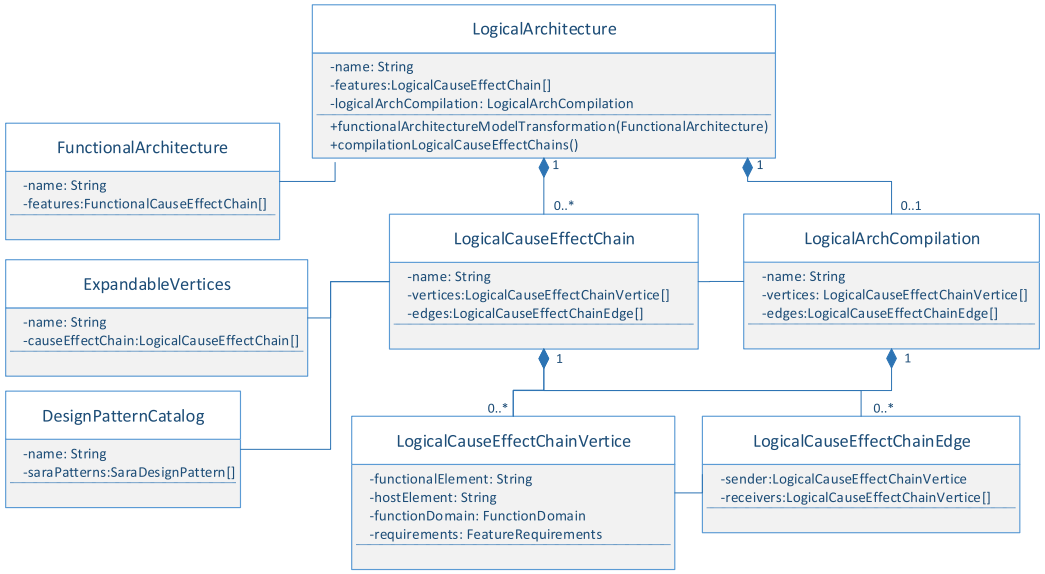


Fig. 8. Meta-model of the logical architecture with its classes, classes' attributes, and relations between classes.

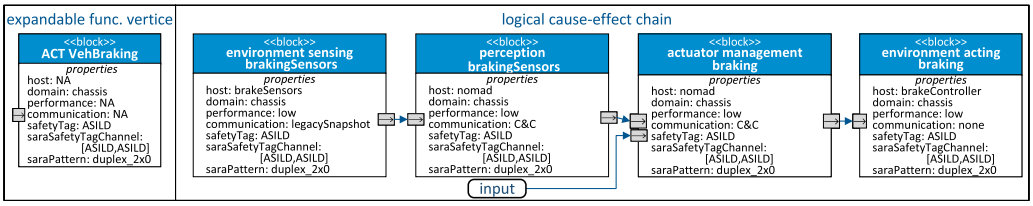
Figure 7, the blocks with the prefixes *SNS* and *ACT* will have a more fine-grained granularity, as specified by the logical architecture explained in the next section.

The other blocks (*SensorFusion, SituationAnalysis, FctBehaviorL3HWP*, and *TrajectoryControl*) contain all the information needed in the logical architecture. For the host, those blocks have the *nomadic* value, meaning that they can be deployed to any vehicle integration platform that matches the domain and the safety requirements. The communication property contains the network traffic. For the SaRA design pattern, the *duplex_2* $\times$ *2* pattern is applied, meaning it has a redundant pattern with two inputs and two outputs. The logical viewpoint will use this information during the model transformation to separate the channels and connect the elements accordingly. The following section describes the logical viewpoint, relating the presented functional architecture with the model transformation.

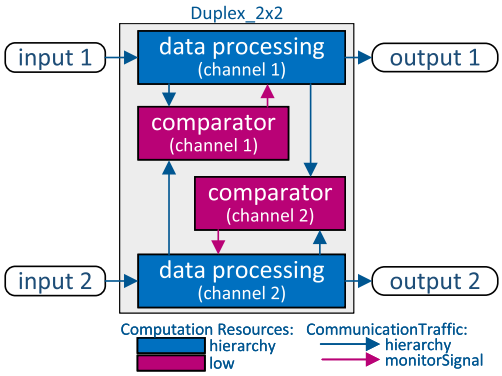
4.2 Logical Viewpoint

The logical viewpoint supports the solution design for the vehicle features, representing the system not solely in terms of functionalities like the functional viewpoint but rather in terms of architectural design. The logical viewpoint represents the implementation elements regarding performance, dependability, and resource demands. For this work, the logical viewpoint will depict the runtime SW architecture, which transforms the functional architecture into a set of computation tasks exchanging information according to the schedules described by the communication traffics. The design patterns are implemented in the logical viewpoint, expanding the functional architecture vertices to generate the fail-tolerant SW architecture.

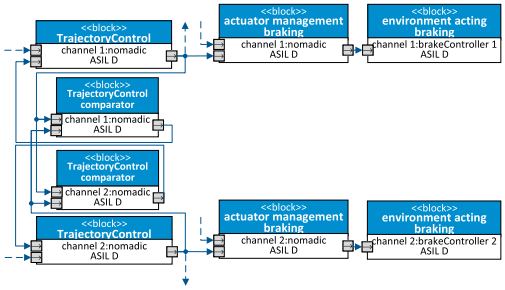
4.2.1 Meta-Model. Figure 8 depicts the meta-model for the logical viewpoint, also called *logical architecture*. The logical architecture consists of 0..* *LogicalCauseEffectChain*, which each represent the runtime SW architecture for a feature, represented as a cause-effect chain. The cause-effect chains are derived from the *logicalArchitecture.functionalArchitectureModelTransformation* method,



(a) Example of an expandable functional architecture vertice to a logical cause-effect chain.



(b) Example of a design pattern as a logical cause-effect chain.



(c) Extract from the SAE Level-3 HWP logical architecture illustrating the model transformations.

Fig. 9. Logical architecture example.

which will parse the functional architecture and transform the models according to the expandable vertices from *ExpandableVertices* and the patterns applied from *DesignPatternCatalog*. Similar to the functional architecture, the logical cause-effect chains consist of 0..* *LogicalCauseEffectChainVertice* and 0..* *LogicalCauseEffectChainEdge*. A vertice represents a SW task with a host, a functional domain, and a set of requirements using the same class *FeatureRequirements* from the functional architecture meta-model. The edge represents the data exchange between functions, having one sender and an array of receivers.

As each logical cause-effect chain represents a vehicle feature's runtime architecture solely, different features might rely on the same functions but with different properties (e.g., safety value, pattern applied) or have the same edges but with different receivers. To check for consistency and to generate the architecture to be deployed to the hosts, the *logicalArchitecture.compilationLogicalCauseEffectChains* method compiles all the logical vertices and edges from *LogicalArchitecture.features*, creating a *LogicalArchCompilation* object with functions and edges. The *LogicalArchCompilation* object has a unique object for the functions and edges according to the instance name, applying the highest ASIL and performance constraints required.

4.2.2 Example. As an example for the logical viewpoint, Figure 9 depicts a part of the model transformations applied to the SAE Level-3 HWP feature showing an excerpt of the logical architecture. Figure 9(a) shows the block *ACT VehBraking* as an example of an expandable functional architecture vertice, illustrating on the right side the logical vertices and edges. Notice that the block *actuator management braking* has a new *input* port to connect to other elements of the cause-effect chain. Moreover, the properties show the requirements for the performance, communication traffic, deployment host, and the SaRA pattern to be applied with the related ASIL for the first and second channels sequentially.

Figure 9(b) shows the example of a duplex 2 × 2 design pattern, with the inner and outer connections and the deployment channels. The figure also indicates the computation resources and communication traffic for the model transformation. The value *hierarchy* for a computation resource means that the block inherits the resource from the original expanded block.

Figure 9(c) shows an excerpt from the logical architecture of the SAE Level-3 HWP feature based on the duplex_2 × 2 pattern applied to the *TrajectoryControl* functional block. The figure also contains the connections to the expandable functional vertice *ACT VehBraking*. The dashed arrows indicate the edges not contained in this excerpt. One can see that the logical architecture separates the information flow into two redundant channels, with an ASIL D requirement for each channel. Furthermore, the figure shows the deployment host for the logical vertices, where the nomadic functions will be deployed according to the related channel and the *environment acting braking* functions will be deployed to redundant brake controllers. For this work, we assume that, in terms of fail-operational capability, the system implements hot-standby [10], with both channels working simultaneously and no additional delay from the recovery mechanisms.

The following section presents the technical viewpoint containing the meta-model for the computation hosts and communication networks. With the technical viewpoint describing the HW architecture and the logical architecture describing the features' runtime SW architecture, it becomes possible to perform the system synthesis, as will be demonstrated in Section 5.

4.3 Technical Viewpoint

The technical viewpoint supports the technical implementation, mapping the abstract elements from the logical viewpoint to the HW resources. In particular, it defines the computational resources executing the SW modules. The automotive industry often refers to the technical viewpoint as the actual vehicle's E/E architecture.

4.3.1 Meta-Model. Figure 10 depicts the meta-model for the technical viewpoint, also called the *technical architecture* hereafter. The technical architecture consists of 0..* *TechnicalArchitectureTopology*, where each topology represents one E/E architecture candidate solution. The E/E architecture follows the template presented in Section 3.1, having 0..* *TechnicalArchitectureNode*, 0..* *TechnicalArchitectureLink*, and 0..* *TechnicalArchitectureBus*. A technical architecture node represents a deployable element, hosting the logical vertices. The *TechnicalArchitectureNodeParameters* defines all properties of the node as follows:

Node type: Indicates if it is an integration platform computer, a zone gateway, or a specific embedded element such as an ECU, a sensor, or an actuator.

Zone location: Indicates the node placement in the vehicle (e.g., front, cabin, or rear compartment).

Gateway tunneling time: Indicates the delay induced by the forwarding of a frame from one network technology to another (e.g., a CAN frame tunneled into an Ethernet frame). This property is only applicable to nodes with a built-in gateway. In this work focusing on Ethernet and CAN networks, this delay is set on a per-node basis.

Switch time: Ethernet switching time. This property is only applicable to nodes with a built-in switch.

Internal connection speed: Indicates the link speed from the built-in switch to the node processor. This property is only applicable to nodes with a built-in switch.

Middleware overhead: Indicates the additional timing delay to process data services. This property is only applicable to nodes running nomadic functions.

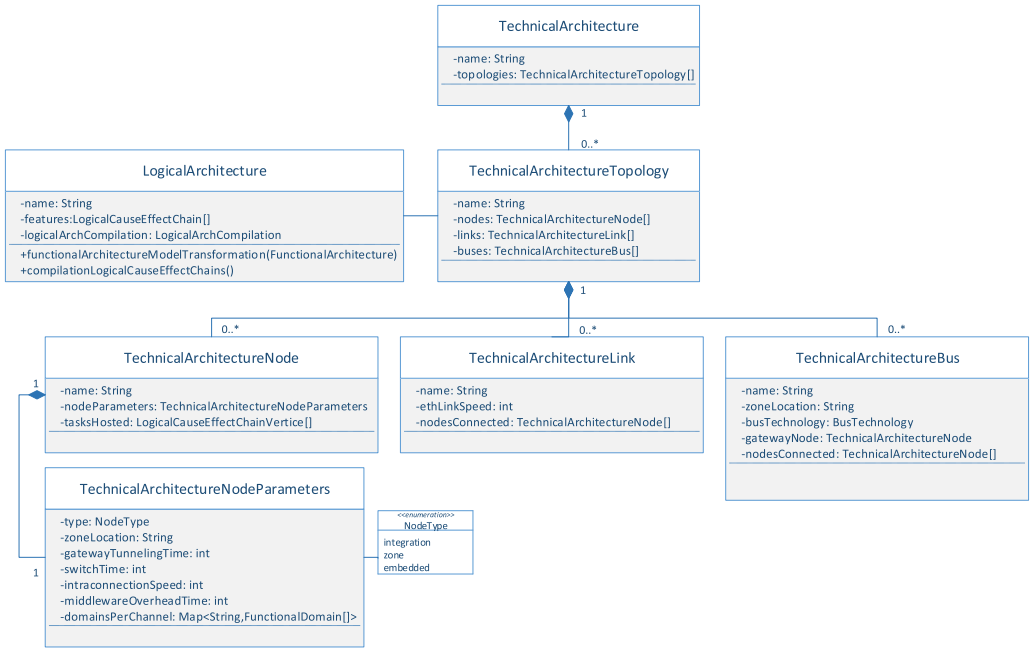


Fig. 10. Meta-model of the technical architecture with its classes, classes' attributes, and relations between classes.

Fall-back channels: Indicates the relationship between vehicle functional domains and deployment channels for nomadic functions. For this work, only integration platforms can host nomadic functions.

The *TechnicalArchitectureLink* and the *TechnicalArchitectureBus* describe the HW topology, expressing the communication network(s) the nodes are connected to. The link class represents an Ethernet connection with a given data speed, and the bus represents a CAN-FD connection. We assume that a bus does not span across different vehicle zones and is connected to a single gateway. This assumption typically holds in zone-based architecture.

4.3.2 Example. An example of the technical viewpoint using this meta-model is given in Figure 11. The example corresponds to the logical architecture's extract from Figure 9(c). The E/E architecture consists of two integration platforms with redundancy for the deployment channels dedicated to the two functional domains in the extract. The architecture also includes three zone gateways and two redundant brake controllers. The integration platforms have the intraconnection link speed defined as they host a switch, and the middleware delay accounts for the communication overhead between nomadic functions. The other nodes have this attribute defined to *null*, as they do not host nomadic functions. The integration platforms and the zone gateways have a built-in switch, and thus the *switchTime* is defined. The only elements with built-in gateways (CAN to Ethernet) are the zone gateways with the property *gatewayTunnelingTime* defined. The two redundant brake controllers are responsible for controlling the braking actuators. The last concern captured by the technical architecture is the communication architecture: networking topology and network protocols are described through the links between the nodes.

The following section presents the DSE approach, giving details on the system synthesis method, deployment combinations generation, and the MOA of the feasible solutions to compare E/E architecture candidates.

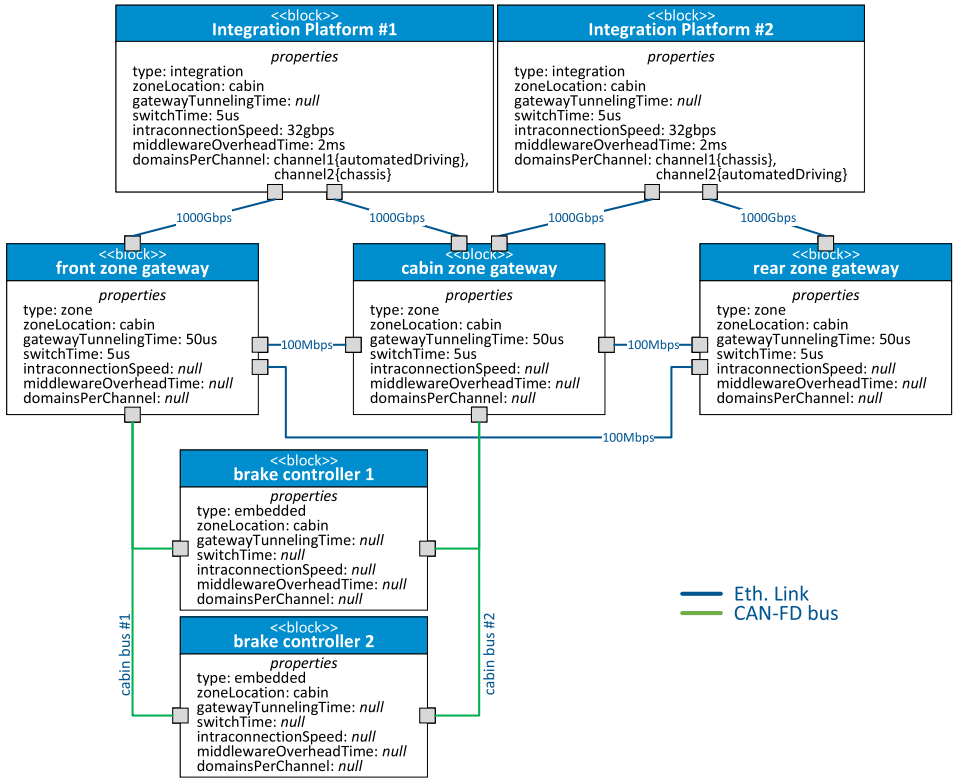


Fig. 11. Technical architecture example using the technical viewpoint meta-model. Representation of the integration platforms, zone gateways, controllers, and networks to host the logical architecture are extracted from Figure 9(c).

5 SYSTEM SYNTHESIS EXPLORATION AND MOA

DSE is a crucial concept in our approach as the logical architecture elements can have many deployment options (see Figure 8), and the technical architecture hosts (see Figure 10) can support multiple vehicle domains. That leads to a space of deployment candidates for the system synthesis, usually comprising both feasible and non-feasible configurations (i.e., configuration not meeting all of the system's requirements). A series of tests are performed to filter out unfeasible solutions. The last test is performance evaluation, which is the most time-consuming filter. Figure 12 illustrates the workflow of the DSE, depicting the generation of the viewpoints meta-models, the generation of deployment candidates at the system synthesis step, and the evaluation process leading to the MOA for the set of feasible solutions. The following sections present the complexity of the exploration and details in each of the workflow steps.

5.1 Complexity of the Problem

The system synthesis consists in populating the *TechnicalArchitectureNode.tasksHosted* attribute (from the technical architecture meta-model) with the *LogicalArchCompilation.vertices* (from the logical architecture meta-model). Assigning logical architecture vertices (i.e., the functions) to possible technical architecture hosts is a problem whose search space grows exponentially with the number of nomadic functions. Each nomadic function can be be allocated to any host that

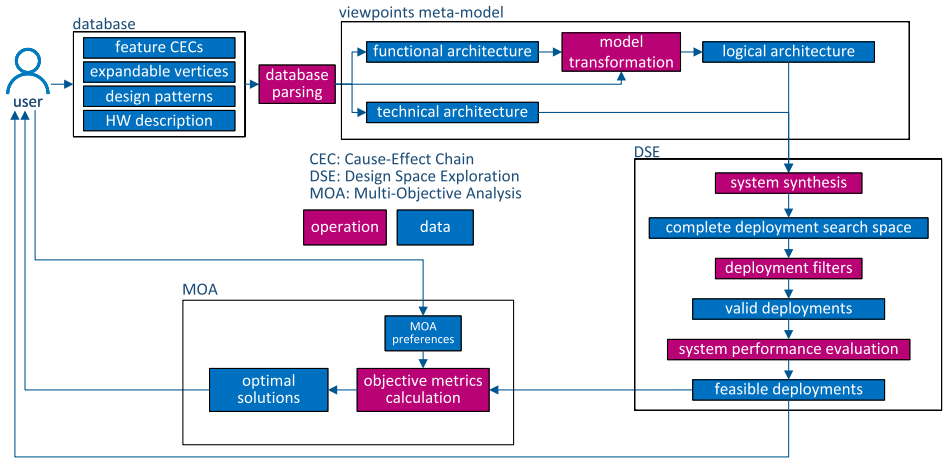


Fig. 12. System synthesis and DSE workflow. Representation of the creation of the viewpoints, generation of feasible deployments, and the MOA.

support the functional domain of the function (e.g., powertrain), knowing that the different channels of a given function must be on different hosts for redundancy purposes. The complexity of the allocation problem is given by the following equation:

$$\prod_{i=1}^{n} \frac{h_i!}{c_i! \cdot (h_i - c_i)!},$$

where n is the total number of nomadic functions, h_i represents the number of possible hosts for a given function i, and c_i is the number of channels of function i.

For instance, considering $n = 25$, two channels, and three possible hosts per nomadic function, the search space is 3^{25}. Assuming 0.5 seconds of computation time per candidate allocation (see Section 5.3), an exhaustive search on a single CPU would require more than 10^4 years of computation. Given that, in practice, the number of nomadic functions can easily range in the hundreds, which suggests that exhaustive search is not practical and that domain expertise is needed to reduce the search space as done in Section 6.2.

5.2 Viewpoints Generation

The viewpoints generation, as depicted in the upper part of the workflow steps in Figure 12, starts with the user populating a database. This database describes in a human-readable format the feature cause-effect chains, the expandable vertices, the design patterns, and the HW description for the E/E architecture. All this information must match the meta-models defined for the viewpoints explained in Section 4. In the workflow's next step, a Java program deserializes the database, parsing the information and checking for the meta-model's parameters consistency to generate the functional and technical architectures. As the database is created manually, it is prone to errors (e.g., typos in the name of a function or a parameter value). To avoid consistency errors, together with the database, a ground truth table is created to summarize all possible parameter values. The table lists the meta-model enumerations and the name of the functions to generate the cause-effect chains. The Java program checks this ground truth table with the database values, and, in case of discrepancy, an error message is provided to the user, indicating the inconsistencies. After the database parsing, and the check for inconsistencies, the Java program generates the logical

architecture through the model transformations explained in Section 4.2, applying the expandable vertices and the design patterns to the functional architecture. The Java source code[7] that parses the database with the checking has around 900 lines of code, and the source code that performs the model transformation has around 1,500 lines of code. With the logical and the technical architectures generated, the next step is to explore the deployment combinations through the system synthesis, as explained in the following section.

5.3 Design Space Exploration

The DSE approach, as depicted on the right side of the Figure 12, starts with the *system synthesis* step generating all possible combinations for the deployment of the logical architecture elements to the technical architecture hosts. To handle the complexity of the problem, we use domain-specific knowledge to constrain the possible technical architecture hosts for given logical architecture elements. For example, chassis and powertrain domain nomadic functions are hosted in specific integration platforms (Section 6.2 gives more details on the approach). Further, to reduce the number of deployment candidates, we use a set of filters to discard unfeasible designs. The block *complete deployment search space* represents the entire space of design candidates. However, many of those solutions will not match the fail-operational architecture constraints (i.e., concerning valid redundant hosts for redundant logical vertices and secondary communication paths for redundant logical edges). The *deployment filters* check for these properties and discard the solutions that do not match the requirements.

For the *valid deployments* only, a system performance evaluation is performed to discard candidates that do not match the requirements, such as in terms of worst-case traversal times, throughput, and max. link bandwidth utilization. The *system performance evaluation* step is the most computation intensive, where according to our observations and measurements in the work of Navet et al. [43] it can be in the order of 500 ms for a worst-case schedulability analysis using RTaW-Pegase [2], the tool used in this study. As shown in Section 5.1, this makes an exhaustive exploration of the design space impossible in a reasonable amount of time.[8] The solutions that pass the filters and the system performance evaluation are labeled as *feasible deployments*, and the MOA is performed on them.

After the system performance evaluation, the user can examine the set of feasible deployments and, if necessary, adapt the database content, the input to the process, to generate a new solution space.

5.4 Performance Criteria for the MOA

For the MOA step, as depicted in the lower part of Figure 12, we consider the following performance metrics and constraints to evaluate the quality of the design candidates:

Solution validity: A Boolean constraint that indicates *true* for valid deployments and *false* for invalid deployments. It is performed during the DSE filtering phase, where the deployment filter checks the correctness of the function allocation and the communication paths, including redundancy constraints. For the function allocation, a valid deployment matches the logical vertices *hostElement* and *functionalDomain* attributes with the technical node parameters. A valid communication path has the correct number of fall-back communication paths

[7] Not taking into consideration the external libraries used to access the database.

[8] Two works [37, 38] explore predicting the results of performance evaluation of TSN networks with graph neural networks, with a speedup greater than 1,000 over the exact approach used in this article, enabling thus much broader search spaces. Studying the viability of relying on prediction, which comes with the drawback that some predictions will unavoidably be wrong, is a potential follow-up work.

according to the redundant logical edges. A valid network performance ensures sufficient data bandwidth for the Ethernet links and no violation of the traversal time deadlines. The latter is checked with worst-case schedulability analysis.

Safety costs: An arithmetic expression that indicates the total sum of the technical architecture costs for various safety levels. The technical architecture nodes have an ASIL cost equal to the highest ASIL value of the functions they host. The arithmetic expression sums the ASIL costs of the technical nodes, where the objective is to find feasible solutions that minimize the costs. In this work, the following ASIL cost metric is used for the nodes:

QM	ASILA	ASILB	ASILC	ASILD
1	5	25	500	2500

Network topology costs: An arithmetic expression that indicates the total sum of the network topology costs. The technical architecture links have a cost related to the speed, and the arithmetic expression sums the link speed costs, where the objective is to find feasible solutions with the lowest cost. For this work, the following link speed cost metric is used[9]:

100 Mbps	1,000 Mbps	2,500 Mbps	5,000 Mbps	10,000 Mbps
1	2.5	3	3.5	8.5

Cabling costs: An arithmetic expression that indicates the total sum of the technical architecture cabling topology. The cost is related to the number of communication cables (Ethernet links and CAN-FD buses) crossing the vehicle zones, increasing the complexity of the wiring harness. Every zone crossing adds 1 to the total costs. The objective is to have the lowest cost.

Network performance: A set of performance metrics to evaluate the network performance using a real-time network design tool. Solutions that do not have enough bandwidth in the Ethernet links or meet the worst-case traversal times are discarded.

Bandwidth scalability: An arithmetic expression that indicates the network topology's scalability for additional traffic. Every technical architecture link has a value from 0 to 100%, representing the link bandwidth usage, in which high values indicate the existence of bottlenecks in the communication topology, whereas lower numbers mean that the network will be able to accommodate additional traffic. Assuming that any link with usage greater than 70% represents a bottleneck (i.e., a common threshold in the automotive industry), we use the equation $\frac{\sum f(n)}{\#\text{ links}}$ with $f(n) = \begin{cases} \frac{load(n)-70}{30} & \text{if } load(n) > 70\% \\ 0 & \text{otherwise} \end{cases}$ as a metric for bandwidth scalability, where n indicates a given Ethernet link. The lower the metric value, the better the overall scalability.

Slack average: An arithmetic expression that indicates how much margin there is for additional communication delays. With worst-case analysis, the amount of time left (i.e., the slack time) before the deadline is computed. The equation $\frac{\sum 1-\frac{slack(n)}{deadline(n)}}{\#\text{ frames}}$ evaluates the average of the slack times, where n is the index for frames. The objective is to have a low average value, which corresponds to a network with ample margins for additional traffic.

TSN protocols: In this work, the impact on the network performance of 802.1Qbu frame preemption [29] and 802.1Qav **Credit-Based Shaper (CBS)** [28] mechanisms are evaluated.

[9]10-Gbps links require additional HW at the physical layer level, which explains that their cost is nonlinear with the data rate.

Computation resources: An arithmetic expression that indicates the total sum of computations resources costs per technical architecture integration platform node. The integration platforms have a cost value equal to the sum of all *performance* attributes from the hosted logical vertices. The objective is to find solutions with the costs distributed evenly between the integration platforms.

For this work, we use the **Dhrystone Million Instructions per Second (DMIPS)** to measure the computation resources required. The following computation resources cost metric is used:

low ($\leq$ 1K DMIPS)	medium (> 1K DMIPS to $\geq$ 10 K DMIPS)	high (> 10 K DMIPS)
1	10	100

5.5 Objective Functions and Optimum Design for the MOA

In a multi-objective problem, the goal is typically to minimize one or several objective functions (see [16]). However, the performance criteria often conflict with each other (e.g., cost vs. performance), meaning that there is no single solution that simultaneously optimizes all the performance metrics. A classical approach to this problem is to look for *Pareto optimal* solutions—that is, solutions that exist for no other feasible solution such that they improve at least one of the objective functions without deteriorating the other objectives [25].

From the mathematical perspective, all Pareto optimal solutions are equally acceptable for the multi-objective problem. However, for practical use, usually, only one solution should be picked, and for that it is necessary to define a **Decision Maker (DM)**. The DM possesses domain knowledge and expresses preference relations between different solutions, such as prioritizing the costs over performance considerations. Miettinen [39] defines three distinct approaches concerning the point in time when the DM interacts with the exploration algorithm and provides the preferences for the solutions:

- *A priori*: The DM determines the preferences before the optimization process, and the exploration algorithm will return a solution that optimizes the preferences. The quality of a solution is determined for instance using the *weighted sum* or ϵ*-constraint* approach (see [25]).
- *A posteriori*: The DM defines the preferences partially, and the exploration algorithm will return a set of solutions that optimizes the partial preferences. One example of this approach is using evolutionary methods (see [16]). After the exploration, the DM will evaluate the trade-offs and complete the preferences to find a solution.
- *Interactive (progressive)*: The DM closely interacts with the exploration algorithm, redefining the preferences multiple times during the search execution. This approach is typically preferred in real-world problems, where the DM can learn progressively non-obvious trade-offs and tune the preferences to find an optimal solution.

For this work, we use the interactive approach. The user can iterate with the exploration workflow at the DSE step and the MOA step, as depicted in the lower part of Figure 12. To obtain the desired optimal E/E architecture solutions, the user, who sets the DM's parameters, can adapt the database models[10] or the MOA preferences. By changing the database, the user will influence the system synthesis and the filters, thus impacting the feasible deployments. By changing the MOA preferences, the optimal solutions will differ for the set of feasible deployments.

[10]Specifically, the design patterns and HW description may be updated during the DSE in light of the results obtained so far.

The optimization problem in this work can be formalized as follows:

$$\min \{f_1(x), f_2(x), ..., f_m(x)\}$$
$$s.t.\ x \in \chi,$$

where $f_i : \mathfrak{R}^m \rightarrow \mathfrak{R}$ are objective functions and $\chi \subseteq \mathfrak{R}^m$ defines the search space. To cast the multi-objective optimization problem into a single-objective optimization problem returning a single design according to the preferences, in the following we use the weighted sum method:

$$\min \sum_{i=1}^{m} w_i \cdot f_i(x)$$
$$s.t.\ x \in \chi,$$

where the weight coefficients w_i are non-negative values.

Ideally, the weights of each of the objective functions are set in the DM, which embeds the domain knowledge. Different schemes have been proposed in the literature to assign the weights. For instance, they can be set according to importance levels, like "high," "medium," or "low," each of which correspond to a numerical value [25]. Another way to allocate weights is to ask the user to rank some of the solutions and then derive the weights accordingly [8]. For an overview of preference modeling in multi-objective optimization, the reader can refer to other works [16, 27, 35].

Because different objective functions can differ in magnitude, their return values should be normalized to ensure a consistent relation between the weights and the objectives. There are different techniques for the normalization (see [25]). For this work, we normalize each of the objective functions in the 0 to 1 range, according to the *Nadir* and *Utopia* points defined in the DM. The Nadir point represents the maximum value for an objective function, and the Utopia point represents the minimum value. The normalization of each of the objective functions is given by

$$0 \leq \frac{f_i(x) - z_i^U}{z_i^N - z_i^U} \leq 1,$$

where z_i^N is the Nadir point and z_i^U is the Utopia point for an objective function f_i.

In the next section, we present the experimental setup including the fail-operational integrated E/E architectures selected for the MOA.

6 EXPERIMENTAL SETUP

To evaluate the framework presented in this work to create and quantify fail-operational integrated E/E architectures, we apply domain knowledge to reduce the total size of the DSE candidates, and we select as examples five technical architectures hosting a set of mixed-critical vehicle features with diverse SaRA requirements. Those technical architectures created in the course of the DSE have been manually selected as they represent distinct design choices relevant to the industry for next-generation automotive architectures. In the following sections, more details are provided on the features included, the usage of domain knowledge to constraint the deployment freedom of the nomadic functions, the technical architectures created, and the communication network configurations.

6.1 Vehicle Features

We model a set of vehicle features according to the meta-models for functional and logical architectures presented in this work. These models are one of the inputs of the problem (see Figure 12). Table 2 lists the features used and the function availability requirements. The table shows that the

Table 2. List of Features Included and the Functions with Availability Requirements

Features	Functions with Availability Requirements
Body/comfort (lights, seats, doors, HVAC)	–
Entertainment	–
Connectivity	–
SAE Level 0 and Level 1 (lateral and longitudinal manual control)	Steering Braking
SAE Level 2 and NCAP (AEB, AES, LKA, DMS, APA)	Braking Steering
SAE Level 3 (highway pilot and traffic jam pilot)	Steering Braking human machine interface sensorSet1 Vehicle motion position *Thinking*

AEB, Automated Emergency Braking; AES, Automated Emergency Steering; DMS, Driver Monitoring System; APA, Automated Parking Assistance; HVAC, Heating, Ventilation, and Air Conditioning.

features *SAE Level 0*, *Level 1*, *Level 2*, and *Level 3* have functionalities with availability requirements. All need fail-operational capabilities for the steering and the braking systems, assuming that the vehicle uses x-by-wire systems. The *SAE Level-3* features have in addition the human interface, a set of sensors, the vehicle position, and the *thinking* function. This function includes sensor fusion, situation analysis, functional behavior, and trajectory control functions for autonomous driving tasks. The human interface informs the user about the SAE Level-3 functionality status and receives commands (e.g., on/off). The sensorSet1 is a subset of all the sensors needed for the autonomous driving tasks, as proposed in the work of Caesar et al. [9]. The *vehicle motion position* is responsible for computing the vehicle position and inertial forces.

6.2 Reduction of Search Space Applying E/E Architecture Patterns

As explained in Section 5.1, it is not feasible to do an exhaustive evaluation of the entire deployment search space, and it is necessary to apply domain knowledge to reduce the number of candidate solutions for evaluation. In this work, using Bosch's internal expertise, we created a set of E/E architecture patterns that constrains the search space concerning the number of integration platforms, the functions domains (e.g., powertrain, chassis) per VC, and the communication network topologies. Figure 13 shows the E/E architecture patterns used in the exploration, following the integrated E/E architecture template from Figure 1.

Figure 13 separates the exploration dimensions in relation to the template layers and the networks that connect the elements between the layers (i.e., inter-layer connection) and inside the layers (i.e., intra-layer connection). To work around the search space explosion problem in the allocation of the nomadic functions, we constrain the deployment by creating five candidate designs for the number of VCs and the function domains they host. Those candidate solutions have been derived using Bosch's know-how—for instance, chassis and powertrain domains are here always on the same computers, because of the expected increase of cross-domain functionalities in next-generation cars. In Figure 13, we see that each VC hosts one or several main channel domains (i.e., the upper part of the box depicting a computer) and one or several secondary channels (i.e., the lower part). Further, we set rules for the network topology: either a ring or a tree topology for the

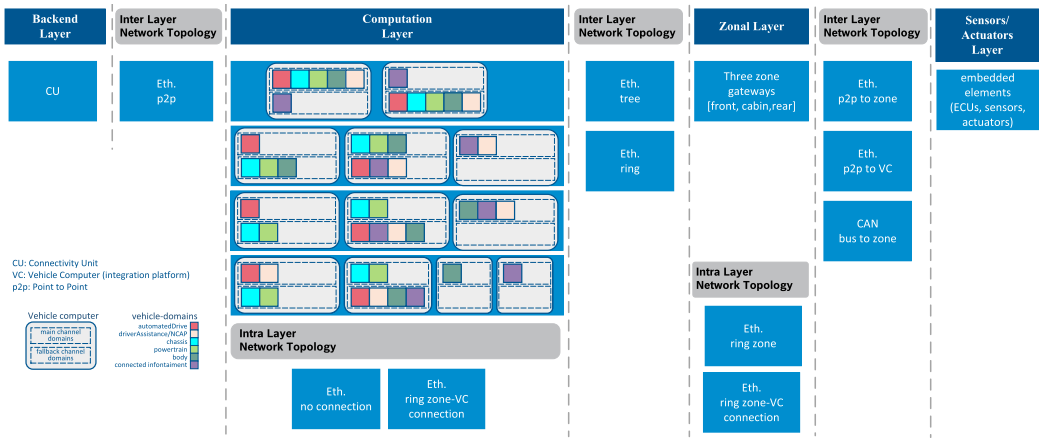


Fig. 13. E/E architecture patterns considered for the DSE. The configurations vary on the network topologies, the number of vehicle computers (i.e., 2 to 4), and the functions domains per vehicle computer. The inter layer network topologies represent the connections between the backend, computation, zonal, and embedded layers. The two intra layer network toplogies describe the connections within the computation layer and within the zonal layer. The patterns are based on a previous work by the authors [23].

connection between the zone gateways and the VCs, point-to-point links to zone, or VCs for embedded devices with Ethernet interfaces (e.g., cameras), whereas CAN buses are used for devices without an Ethernet interface.

Without those E/E architecture patterns, for the list of features from Table 2, and considering only the case of three VCs, the search space for the allocation of the nomadic functions would be in the order of 10^{14}. This number would be even higher if we explore all the possible network topologies. Using the E/E architecture patterns from Figure 13, this number comes down to around $17k$ combinations, making it possible to analyze the solutions in less than 2.5 hours, assuming 0.5 seconds per performance evaluation. From all the configurations, the ones that do not pass the deployment filters are discarded, as presented in Section 5.3, and only the feasible candidates will be considered for the MOA.

For the rest of this work, instead of showing all the feasible candidates, we will pick five E/E architecture solutions from the E/E architecture patterns from Figure 13 to further explore the MOA. Those five architectures were selected after a Bosch internal evaluation, as they represent possible automotive integrated E/E architectures for the next generation of vehicles with different HW resources and topologies.

6.3 Integrated E/E Architectures Selected

The five technical architectures selected represent good candidates for next-generation automotive integrated E/E architectures, based on distinct and realistic design choices in terms of network topology and number of integration platforms. All architectures pass the system deployment filters, having at least the minimum number of redundant hosts and communication paths to support the logical architecture corresponding to the previously listed features. For simplicity, in this work, the architectures' models only include the elements having Ethernet or CAN-FD connections (e.g., wireless, LIN, and FlexRay connections are not modeled here).

Figure 14 depicts the architectures selected. The architectures undergo two kinds of analyses: network performance for technical architectures #1, #2, and #3, and computational performance

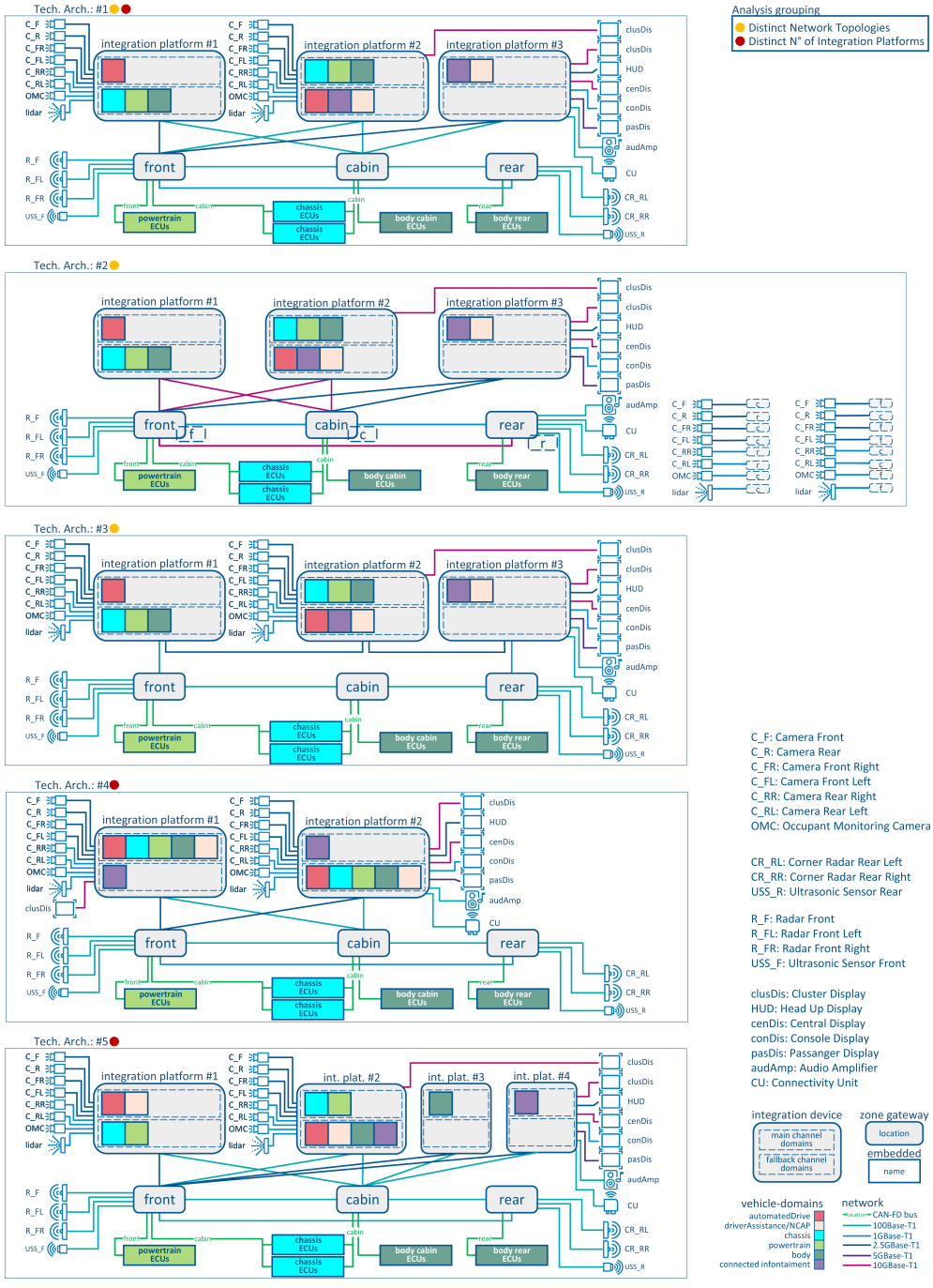


Fig. 14. Selected E/E architectures for evaluation. The technical architectures #1, #2, and #3 differ in the network topology, and the technical architectures #1, #4, and #5 differ in the number of integration platforms. In technical architecture #2, the small blocks with the letters *f*, *c*, and *r* represent a connection to the zone gateway from the cameras and lidars.

for technical architectures #1, #4, and #5. Section 7 will then present the MOA conducted on the basis of the analyses' results.

All architectures possess three zone gateways and the same embedded components (i.e., the elements of the sensors/actuators layer in Figure 1). Furthermore, they all have the ECUs organized in vehicle domains and connected to the zone gateways through CAN FD buses (powertrain, chassis, body cabin, and body rear ECUs block illustrated at the bottom of the architectures in Figure 14). According to the fail-operational requirements, some embedded elements (e.g., cameras, lidar, and chassis ECUs) are duplicated.

The architectures for the computational performance analysis group (technical architectures #1, #4, and #5) have the same network topology concept but different integration platforms. They rely on an Ethernet ring topology for the zones, and each integration platform has two connection points to the zone ring. The cameras, lidars, and displays are connected to the integration platforms. The radars and ultrasonic sensors are connected to the zone gateways. Technical architecture #1 has three integration platformsand technical architecture #4 has two integration platforms, which is the design with the minimal number of possible redundant integration platforms. In contrast, technical architecture #5 includes four integration platforms depicting a design with domain-oriented VCs, with one computer for infotainment, one for body, one for motion (chassis and powertrain), and one for autonomous driving and driver's assistance.

The architectures for the network performance analysis group (technical architectures #1, #2, and #3) have the same integration platforms, but they differ in the network topology. Technical architecture #2 has the same Ethernet ring topology as technical architecture #1, but the cameras, lidars, radars, ultrasonic sensors, audio amplifier, and connectivity unit are connected to the zone gateways. To avoid single points of failure when connecting redundant channels to the same zone gateway in technical architecture #2, the duplicated components (e.g., cameras and lidars) are connected to different zone gateways, guaranteeing redundant communication paths for these channels. Finally, technical architecture #3 has the same connections for the embedded components as technical architecture #1, but it differs by having an Ethernet ring passing through the zones and the integration platforms as well. The following section details the communication network design, such as link speeds, forwarding delays, and TSN protocols used.

6.4 Network Configuration

All the selected architectures use the data rate for the Ethernet links chosen to be the minimum possible speed to handle the communication requirements. We assume for all architectures that the middleware overhead delay in the integration platforms is equal to 2 ms, and the intraconnection speed (i.e., from the internal switch to the processor) is equal to 32 Gbps. All switches have a switching delay equal to 5 μs, and the gateways' tunneling delays (see Section 4.3.1) are set to 50 μs.

The communication traffic for the Ethernet frames is either periodic or periodic burst (i.e., packets making up a segmented message, like a camera frame, are queued at once). Due to the competitive nature of the network traffic, we use TSN QoS mechanisms to meet the communication requirements (see Section 4.3.1). The use of priorities (eight priority levels are available in TSN) with preemption [29] leads to feasible solutions in terms of timing requirements (throughput and deadlines) for all the selected architectures. The priority allocation and the timing verification (i.e., worst-case schedulability analysis) are performed using the network design tool RTaW-Pegase [2].

Notice that none of the architectures would meet the timing requirement with a single priority, known as the FIFO configuration. We explored the use of the CBS, defined in IEEE802.1Qav [28], but it did not lead to more efficient scheduling solutions. Indeed, CBS is mostly useful to bursty flows, such as camera streams, which, in our case study, had to be placed at the lowest priority

levels due to very time-constrained control messages. Thus, very few streams could benefit from the shaping performed on the bursty traffic.

The following section presents the MOA for the selected five architectures. In particular, we will explore two sets of analysis meant to evaluate (1) network performance and (2) computational performance. Further, we will determine the optimum design according to the DM preferences.

7 EXPERIMENTAL RESULTS

The experimental process starts with the creation of a database storing the features and integrated E/E architecture parameters described in Section 6 according to the viewpoint meta-models from Section 4. The database parsing and the creation of the functional, logical, and technical architectures, including the deployment, were performed with a Java program. For the network simulation, we used the RTaW-Pegase 3.9.8 [2] library running on Java JDK-13.[11] The MOA is programmed with Python 3.7 using the numpy, pandas, seaborn, and plotly libraries for the results' processing and the generation of the figures.

The evaluation of the five selected E/E architectures is organized in two experiments according to the analysis grouping:

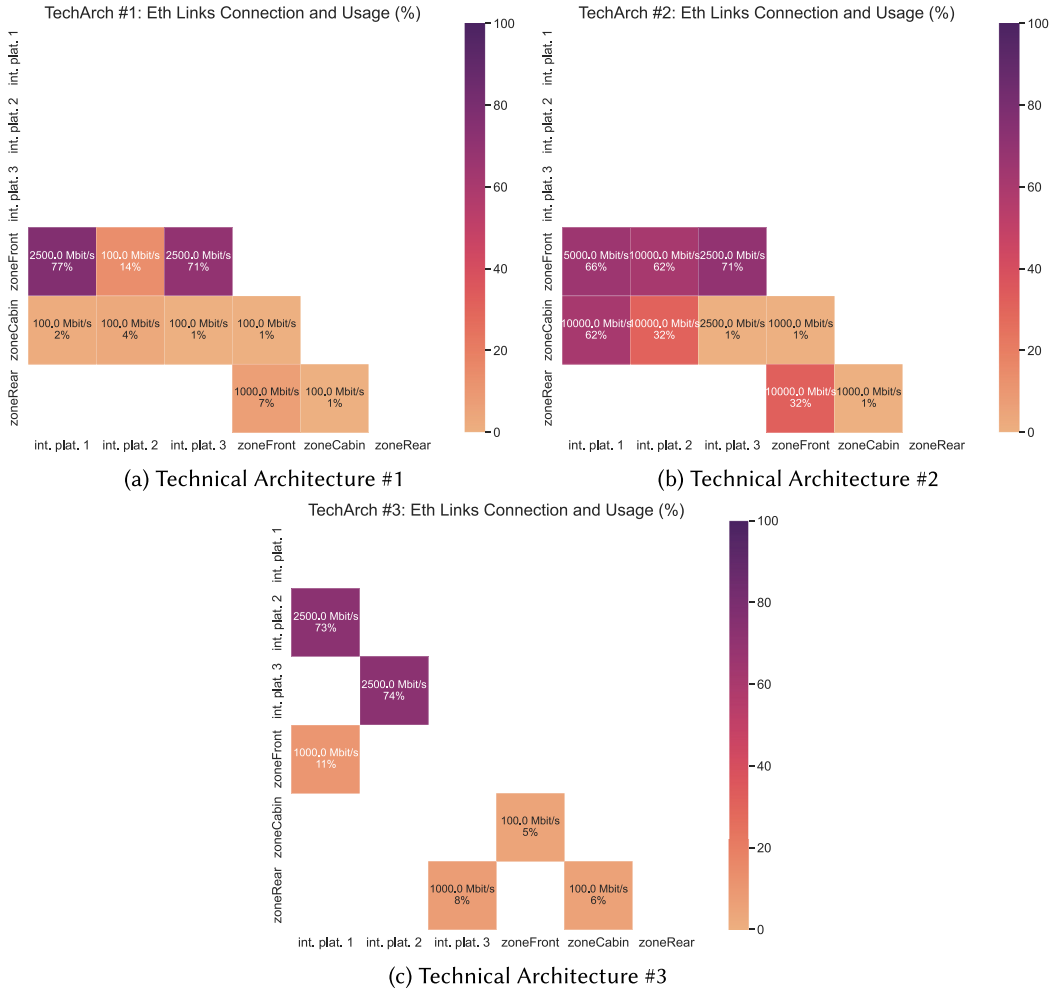
Network topologies experiment: First, we explore the impacts of different network topologies using the network performance analysis group (i.e., technical architectures #1, #2, and #3) from Figure 14. We compare the Ethernet link loads to analyze potential bottlenecks in the communication networks, and we evaluate the MOA with objective functions for *network topology cost*, *cabling cost*, *slack average*, and *bandwidth scalability*, as described in Section 5.4. We normalize the objective functions and apply the weighted method to obtain a single optimum solution. For simplicity, we assume that each objective function possesses a weight of 1, and the Nadir and Utopia points are equal to the minimum and maximum values, respectively.

Integration platforms experiment: Second, we explore the impacts of changing the number of integration platforms and the distribution of vehicle domain deployment channels using the computational performance analysis group (i.e., technical architectures #1, #4, and #5) from the exact figure. We compare the computation resources required for each integration platform according to the deployments. For the MOA, we evaluate the objective functions *network topology cost*, *cabling cost*, *slack average*, and *safety cost*, as described in Section 5.4. We normalize the objective functions and apply the weighted method to obtain a single optimum solution. For simplicity, we assume that each objective function has a weight of 1, and the Nadir and Utopia points are equal to the minimum and maximum values, respectively.

7.1 Evaluation of Integrated E/E Architectures: Network Topologies

Figure 15 depicts the Ethernet link speeds and percentage of the bandwidth usage for each link for the network performance analysis group (i.e., technical architectures #1, #2, and #3). Of the three designs, the network topology from #2 has the highest Ethernet link speeds because the cameras and lidars, which are very demanding in terms of bandwidth, use the zone ring to route the frames to the integration platforms. Figure 15(c) shows that solution #3 has two potential communication

[11]It is important to note that ISO 26262 (part 8, chapter 11) formulates specific requirements for SW tools used in the development of safety-related automotive E/E systems. The standard gives guidelines to determine the necessary tool confidence level and provide means to qualify a tool according to the tool confidence level. In this work, as we focus on the early stages of the automotive development process, it is not needed to guarantee the confidence of tools' outputs. However, it is expected that qualified tools, possibly qualified external verifiers, are applied at later stages of the development process, following ISO 26262 guidelines.



(a) Technical Architecture #1

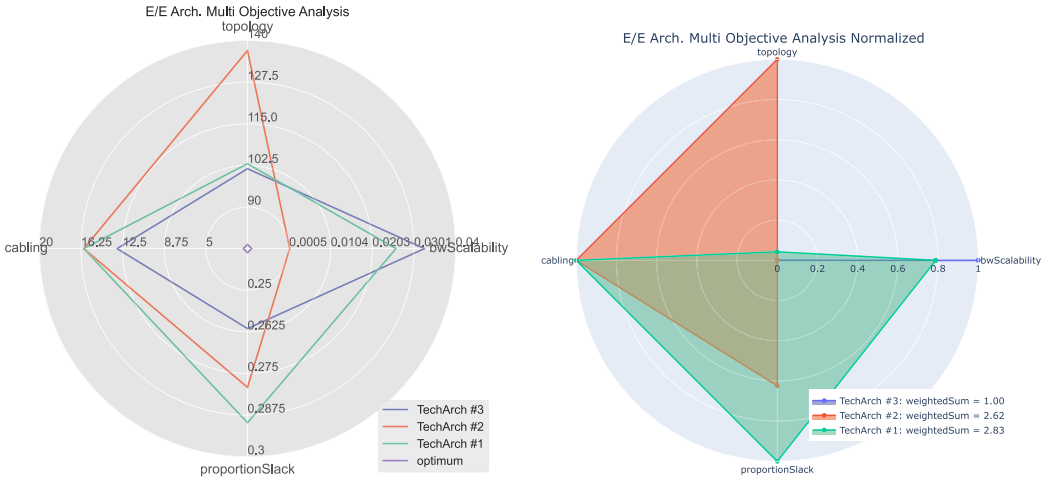
(b) Technical Architecture #2

(c) Technical Architecture #3

Fig. 15. Evaluation of the network performance analysis group (technical architectures #1, #2, and #3) in terms of the Ethernet links' speeds and utilization between the nodes. Each technical architecture is shown in a distinct sub-figure. The right side of each architecture indicates the color code for the link utilization, with darker colors representing higher percentages.

bottlenecks. The links between integration platforms #1 to #2 and #2 to #3 have a bandwidth usage percentage closer to 100%.

As an aid to decide on an optimum design regarding the communication network between the architectures, Figure 16 illustrates the MOA, depicting the objective function values in the form of a radar plot, with the optimum values in the center. Figure 16(a) shows the quality metrics in absolute values as described in Section 5.4. After the normalization and the application of the weighted sum method (see Figure 16(b)), technical architecture #3 comes out as the optimum design for having the lowest sum value. Solution #3 has the best metrics for network topology costs, cabling costs, and slack average. This is because it has a single ring for the communication network, hence reducing the number of cables. In addition, the high-demand bandwidth elements, such as the cameras, lidars, and screens, are connected directly to the integration platforms, allowing for a



(a) MOA: absolute values (b) MOA: normalized and weighted sum method

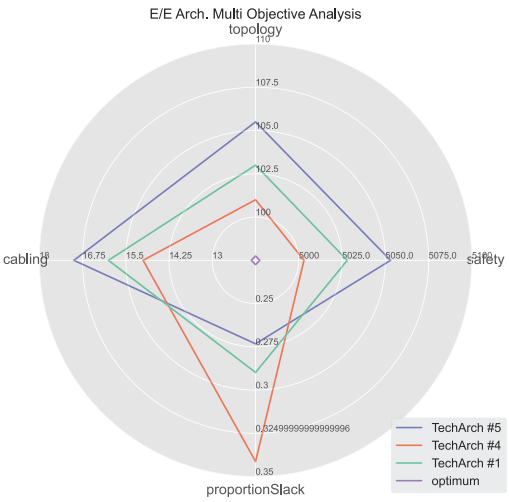
Fig. 16. Evaluation of the network performance analysis group (technical architectures #1, #2, and #3) in terms of the MOA for the network topologies. The center of each radar plot represents the optimum values. The total weighted sum results are illustrated in the legend of the normalized graph.

ring network with lower Ethernet link speeds. Looking at the cabling metric, one could expect that setup #2 would be the optimum having the embedded elements connected to the closest zone gateway. However, connecting two redundant elements to the same zone gateway will lead to a single point of failure, making the solution unfeasible. Because of that, it is necessary that the duplicates of the same component are not connected to the same zone gateway—for example, *Camera Front (C_F)* has one instance connected to the front zone gateway and the other instance connected to the cabin zone gateway. The only metric for which architecture #3 is outperformed by architectures #1 and #2 is the bandwidth scalability. It is possible to see in Figure 15(c) that the Ethernet links between integration platforms #1, #2, and #3 have a high total bandwidth utilization, leading to potential bottleneck points.

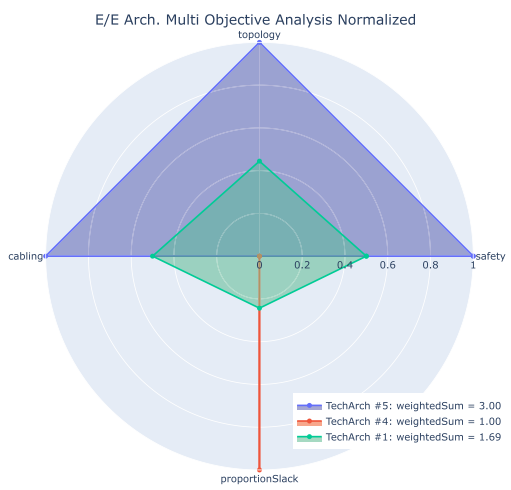
7.2 Evaluation of Integrated E/E Architectures: Number of Integration Platforms and Vehicle Domains Distribution

Figure 17 presents the evaluation's results of the computational performance analysis group (i.e., technical architectures #1, #4, and #5) that differs in terms of the number of integration platforms and the distribution of the vehicle domains (see Section 6.3). MOA is performed to support decision making: Figure 17(a) and (b) show the objective functions for the three candidate solutions in the form of a radar plot, with the optimum values in the center. Figure 17(a) shows the quality metrics in absolute values as described in Section 5.4. After the normalization and the application of the weighted sum method (see Figure 17(b)), technical architecture #4 comes out as the optimum design for having the lowest sum value. Indeed, solution #4 features the best metrics for safety, network topology, and cabling costs. That is due to the fact that it possesses only two ASIL D computers, the minimum as per the redundancy requirements, leading all other solutions to be less cost-effective.

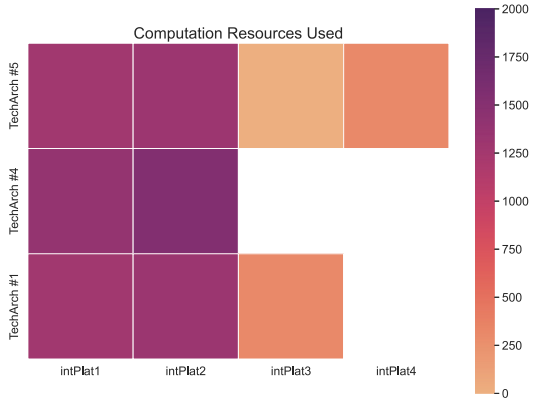
Network topology and cabling costs are also minimized as architecture #4 has fewer Ethernet connections to the zones gateways. However, this design is outperformed in the average slack metric because the reduced number of Ethernet links increases the load on the links. This means that the evolutivity of the E/E architecture will be less than with other candidate solutions.



(a) MOA: absolute values



(b) MOA: normalized and weighted sum method



(c) Computation resources needed

Fig. 17. Evaluation of the computational performance analysis group (technical architectures #1, #4, and #5) in terms of the number of integration platforms and vehicle domain distribution. The center of each radar plot ((a) and (b)) represents the optimum values. The total weighted sum results are given in the legend of the normalized graph. The right side of (c) illustrates the amount of computation resources used, with darker colors representing higher resources amount.

The computational resources needed by the integration platforms, as described in Section 5.4, is presented in Figure 17(c). The figure shows that integration platforms 1 and 2 have the highest computational requirements. That is because the automated driving functions, which are very demanding in terms of computation resources, are hosted in those integration platforms. In the same figure, we observe that the additional integration platforms 3 and 4 only use a small quantity of CPU resources, making it the best solution to have only two integration platforms as in architecture #4. This explains why the MOA, with the specific quality metrics chosen in this study, returns architecture #4 as the best design concerning the number of integration platforms and the distribution of the vehicle domains onto the integration platforms.

However, it should be noted that decisions by the OEMs about the HW architecture consider mass production constraints. This means that the choices made for a given architecture design

cannot be completely independent of the OEM's product portfolio. Our approach could be extended in that direction in a follow-up study.

8 RELATED WORKS

Profound changes are ongoing in the automotive industry, driven by megatrends such as automated driving and connectivity. New design approaches are emerging to handle the arising technologies, and with that, there is a variety of publications exploring the issues and solutions. Natale and Vincentelly [13], back in 2010, forsaw that the need to cope with the increased functional complexity, along with cost, flexibility, and extensibility constraints, would challenge the traditional vehicles E/E architecture designs. They proposed that a necessary change in the design paradigm would be moving from *Federated Architecture* to *Integrated Architectures*, similar to the widely adopted IMA in the avionics sector [57]. Navale et al. [42] presented an overview of the evolutions and, as they call it, *revolutions* required for automotive E/E architectures, sharing their vision of the future technologies. The publication explores how the megatrends will affect the car's architecture and highlight several bottlenecks, such as computation power and communication channels. To overcome the potential bottlenecks, they list technologies that could fulfill the next-generation requirements, like *Automotive Ethernet*, *Advanced Gateways*, and *Integration Platforms*.

Focusing on integration platforms in the context of the advent of SOA and the separation of concerns between HW and SW, Traub et al. [53] showed how solutions from the IT world could enhance the automotive E/E architecture, increasing flexibility and scalability. Closely related is the work of Kugele et al. [33], which explored the research challenges arising from bringing such solutions from the IT world to the vehicle architecture. They created a list of questions to be solved by the automotive community organized in four impact focus areas: Safety, Communication, SOA, and Intelligence.

Integrating IT solutions into the next generation of automotive E/E architectures is a non-trivial task. One of the challenges is to design such systems while fulfilling the many automotive QoS requirements and specifically dependability [3]. Dependability is a property comprising several distinct attributes: Reliability, Availability, Maintainability, Safety, and Security (RAMSS for short), which often are correlated issues. Indeed, a design that does not meet one requirement may impact the others. As stated by Avizienis et al. [3], safety is a critical attribute guaranteeing the"absence of catastrophic consequences on the user(s) and the environment."Today, there are several safety standards and methodologies to support the development of systems with various levels of dependability requirements. Some of the essential standards for automotive safety are ISO 26262 [17], focusing on the functional safety of automotive E/E systems, and ISO/PAS 21448 [18], focusing on the safety of the intended functionality. Recently, new standards were published to address the safety of autonomous vehicles, namely ISO TR 4808 [19] and UL 4600 [54], both providing guidelines for the design, validation, and verification of SAE $\geq$ Level-3 cars.

As remarked in ISO TR 4808[19], safety is an intrinsic attribute of a system, which must be addressed during the design phases. The use of proven-in-use architecture patterns is certainly an efficient way to enhance safety, building on the seminal work of Gamma et al. [21] in the 1990s, which presented a catalog of reusable object-oriented SW design patterns. Later, Douglass [14] explored the patterns' concept for real-time systems using an MBSE approach with Unified Modeling Language. Another noteworthy contribution in this landscape is the doctoral dissertation of Armoush [1], which provided a catalog of SW and HW design patterns to support the design of safety-critical embedded systems, considering the pattern topology and non-functional metrics (e.g., random failure rate) for the evaluation of the solutions. Matos et al. [55] investigated the design of safety IMA systems in the avionics sector with design patterns formalized in **System**

Modeling Language (SysML), and quantify the solutions' availability and integrity failure rate. Later, Khalil [32] proposed a pattern library for the reuse of safety mechanisms in the automotive domain and its use as a plugin to a generic research CASE (Computer-Aided Software Engineering) tool.

When applying architectural patterns to design safe automotive E/E systems, the engineers typically face the challenge of finding optimal solutions in a multi-objective problem (e.g., costs against performance). There is a large diversity of work on the multi-objective problem and architecture exploration. For example, the works of Pinto et al. [45] and Zverlov and Voss [60] explored the design of system-on-chips, evaluating the timing, safety, cost, and energy consumption properties according to the communication protocols, system typology, and mapping of tasks to the cores. Looking for more generic cyber-physical system architectures, the works of Glaßet al. [22] and Bajaj et al. [4] explored the system synthesis and communication routing to minimize the system's costs while guaranteeing the desired reliability and performance.

Focusing on automotive architectures, Kanajan et al. [31] started to evaluate the trade-offs for network performance and HW topology between centralized and federated architectures. Looking for safety with ISO 26262 in focus, Rupanov et al. [47] presented an MBSE approach that allows combining safety mechanisms to design the systems architecture with the lowest failure rate. Schätz et al. [51] contributed to this line of work with a method to automate the allocation of SW functions to HW components finding deployments that meet the safety requirements. Closely related, still with a focus on ISO 26262, the work of Frigerio et al. [20] used the concepts of ASIL decomposition to benchmark domain versus zone-based automotive architectures capable of performing safety-critical autonomous applications running in non-redundant HW architectures. The work evaluates total cost, computation and communications loads, and total communication cable length. Similarly, Eder et al. [15] presented a method to automatically generate safe automotive E/E architectures based on a formal language to describe the HW and SW elements. The exploration uses satisfiability modulo theories to find Pareto optimal solutions for costs, power consumption, and networks usage.

The two latter works [15, 20] used fixed topologies for the evaluation, not exploring different network designs (e.g., ring and star) and different E/E architecture patterns (e.g., domains per integration platforms). Furthermore, both works used static analysis of the communication loads, not evaluating the impact of the network topology on the performance (e.g., latency and jitter). Finally, both works relied on ISO 26262 to guide the synthesis of safe automotive architectures. However, the topic of safety-available requirements as stated by ISO 2626:2018-Part 10 [17] has not been addressed yet.

9 CONCLUSION AND FUTURE WORK

In this work, we presented a framework to support the synthesis and the evaluation of integrated automotive E/E architectures with SaRA requirements in the early phases of the vehicle development process. First, we explained the concepts for SaRA requirements based on ISO 26262, and we created a template to describe design patterns for systems with fail-operational capabilities. Second, we created meta-models for the functional, logical, and technical architectures to describe automotive integrated E/E architectures, including the design patterns template and the system synthesis enabling the DSE with MOA. Because of the exponential growth of the DSE search space, we suggested a set of E/E architecture patterns based on domain knowledge to constrain the synthesis combinations. To illustrate the methodology, we selected five automotive architectures based on the proposed E/E architecture patterns varying in the number of VCs and network topology hosting a set of vehicle features with mixed SaRA requirements, including highly automated

driving functions. According to the metrics defined for evaluation, we performed MOA and identified solutions that are optimal with respect to different objectives.

The framework proposed to model integrated automotive architectures with SaRA and QoS requirements enables DSE with MOA using MBSE techniques and performance evaluation. This work may benefit OEMs, Tier-1 suppliers, and researchers interested in the design of E/E architecture using model-based approaches.

The article's results match an overall E/E architectural trend foreseen by our colleagues at Robert Bosch GmbH, expecting a reduction of the number of VCs down to 2 with a ring topology connecting them to the zone controllers.

As future work, we plan to enhance the methodology to include SOA, analyzing the impacts of signals and services translation as presented in the work of Oliveira et al. [23]. In addition, we would like to include scalability metrics in the MOA, evaluating which E/E architectures can better scale in supporting additional vehicle features. Finally, to prune the search space early in the DSE, we will study the possibility to couple the approach in this article with constraint programming or integer linear programming as in the work of Nuzzo et al. [44].

ACKNOWLEDGMENTS

We want to thank the colleagues from Robert Bosch GmbH for providing suggestions that helped improve the presented methodology, particularly Carsten Gebauer and Dr. Marcel Mausser, members of the center of competence for vehicle safety, for clarifications on the SaRA topics according to ISO 26262:2018. Additionally, we would like to thank Razvan Mihalach and Damon Martini, experts in in-vehicle networks architecture and simulation, for the advice on the design of integrated automotive E/E architectures and the choice of the metrics for the multi-objective analysis.

REFERENCES

[1] Ashraf Armoush. 2010. *Design Patterns for Safety-critical Embedded Systems*. Ph. D. Dissertation. RWTH Aachen University, Aachen, Germany. Advisor(s) Stefan Kowalewski. http://publications.rwth-aachen.de/record/51773/files/3273.pdf.

[2] RealTime at Work. [n.d.]. 2019. RTaW-pegase: Modeling, simulation and automated configuration of communication networks. https://www.realtimeatwork.com/software/rtaw-pegase.

[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. 2004. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing* 1, 1 (2004), 11–33. https://doi.org/10.1109/TDSC.2004.2

[4] Nikunj Bajaj, Pierluigi Nuzzo, Michael Masin, and Alberto Sangiovanni-Vincentelli. 2015. Optimized selection of reliable and cost-effective cyber-physical system architectures. In *Proceedings of the 2015 Design, Automation Test in Europe Conference Exhibition (DATE)*. 561–566. https://doi.org/10.7873/DATE.2015.0913

[5] C. Becker, J. Brewer, D. Arthur, and F Attioui. 2018. *Functional Safety Assessment of a Generic Steer-by-wire Steering System with Active Steering and Four-wheel Steering Features*. National Highway Traffic Safety Administration (NHTSA).

[6] C. Becker, L. Yount, S. Rosen-Levy, and J. Brewer. 2018. *Functional Safety Assessment of an Automated Lane Centering System*. National Highway Traffic Safety Administration (NHTSA).

[7] Hugh Blair-Smith. 2009. Space shuttle fault tolerance: Analog and digital teamwork. In *Proceedings of the 2009 IEEE/AIAA 28th Digital Avionics Systems Conference*. 6.B.1–1–6.B.1–11. https://doi.org/10.1109/DASC.2009.5347450

[8] Jürgen Branke, Salvatore Greco, Roman Słowiński, and Piotr Zielniewicz. 2015. Learning value functions in interactive evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 19, 1 (2015), 88–102. https://doi.org/10.1109/TEVC.2014.2303783

[9] Holger Caesar, Varun Kumar Reddy Bankiti, Alex Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. nuScenes: A multimodal dataset for autonomous driving. 11618–11628. https://doi.org/10.1109/CVPR42600.2020.01164

[10] Marco Chiesa, Andrzej Kamisiński, Jacek Rak, Gábor Rétvári, and Stefan Schmid. 2021. A survey of fast-recovery mechanisms in packet-switched networks. *IEEE Communications Surveys Tutorials* 23, 2 (2021), 1253–1301. https://doi.org/10.1109/COMST.2021.3063980

[11] Oliver Creighton, Nicolas Navet, Patrick Keller, and Jörn Migge. 2020. Towards computer-aided, iterative TSN-and ethernet based EE architecture design. *2020 IEEE Standards Association (IEEE-SA) Ethernet & IP @ Automotive Technology Day* (sep 2020). http://hdl.handle.net/10993/44490.

[12] Joseph D'Ambrosio and Rami Debouk. 2013. ASIL decomposition: The good, the bad and the ugly. https://doi.org/10.4271/2013-01-0195

[13] Marco Di Natale and Alberto Luigi Sangiovanni-Vincentelli. 2010. Moving from federated to integrated architectures in automotive: The role of standards, methods and tools. *Proc. IEEE* 98, 4 (2010), 603–620. https://doi.org/10.1109/JPROC.2009.2039550

[14] Bruce Powel Douglass. 1999. *Doing Hard Time: Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns*. Addison-Wesley Longman Publishing Co., Inc., USA.

[15] Johannes Eder, Sebastian Voss, Andreas Bayha, Alexandru Ipatiov, and Maged Khalil. 2020. Hardware architecture exploration: Automatic exploration of distributed automotive hardware architectures. In *Softw Syst Model*, Vol. 19. 911–934. https://doi.org/10.1007/s10270-020-00786-6

[16] Michael T. Emmerich and André H. Deutz. 2018. A tutorial on multiobjective optimization: Fundamentals and evolutionary methods. *Natural Computing: An International Journal* 17, 3 (sep 2018), 585–609. https://doi.org/10.1007/s11047-018-9685-y

[17] International Organization for Standardization (ISO). 2018. ISO 2626:2018 - road vehicles - functional safety. (2018).

[18] International Organization for Standardization (ISO). 2019. ISO/PAS 21448:2019 - Road vehicles - Safety of the intended functionality. (2019).

[19] International Organization for Standardization (ISO). 2020. ISO/TR 4804:2020 Road vehicles - safety and cybersecurity for automated driving systems - design, verification and validation. (2020).

[20] Alessandro Frigerio, Bart Vermeulen, and Kees G. W. Goossens. 2021. Automotive architecture topologies: Analysis for safety-critical autonomous vehicle applications. *IEEE Access* 9 (2021), 62837–62846. https://doi.org/10.1109/ACCESS.2021.3074813

[21] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Adisson-Wesley, Boston, MA, USA.

[22] Michael Glaß, Martin Lukasiewycz, Rolf Wanka, Christian Haubelt, and Jürgen Teich. 2008. Multi-objective routing and topology optimization in networked embedded systems. 74–81. https://doi.org/10.1109/ICSAMOS.2008.4664849

[23] Ricardo Gonzalez de Oliveira, Christian Kerstan, and Achim Henkel. 2021. Keynote: Service oriented architecture chances and challenges. *Automotive Ethernet Congress 2021* (fev 2021). http://hdl.handle.net/10993/46249.

[24] Ricardo Gonzalez de Oliveira, Indrasen Raghupatruni, Arne Hamann, and Achim Henkel. 2021. *Virtual Verification of Cause-Effect Chains in Automotive Cyber-Physical Systems*. 279–290. https://doi.org/10.1007/978-3-658-33521-2_21

[25] Oleg Grodzevich and Oleksandr Romanko. 2006. Normalization and other topics in multi-objective optimization. *Proceedings of the Fields-MITACS Industrial Problems Workshop*.

[26] Flühr H. 2012. *Flugzeugsysteme. In: Avionik und Flugsicherungstechnik*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-33576-1_11

[27] Jussi Hakanen, Tinkle Chugh, Karthik Sindhya, Yaochu Jin, and Kaisa Miettinen. 2016. Connections of reference vectors and different types of preference information in interactive multiobjective evolutionary algorithms. In *Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. 1–8. https://doi.org/10.1109/SSCI.2016.7850220

[28] IEEE. 2009. IEEE standard for local and metropolitan area networks - virtual bridged local area networks amendment 12 forwarding and queuing enhancements for time-sensitive streams (Std 802.1Qav-2009 ed.). (2009).

[29] IEEE. 2016. IEEE standard for local and metropolitan area networks - bridges and bridged networks - amendment 26: Frame preemption (IEEE Std 802.1Qbu-2016 ed.). (2016).

[30] Alexandru Kampmann, Bassam Alrifaee, Markus Kohout, Andreas Wüstenberg, Timo Woopen, Marcus Nolte, Lutz Eckstein, and Stefan Kowalewski. 2019. A dynamic service-oriented software architecture for highly automated vehicles. In *Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2101–2108. https://doi.org/10.1109/ITSC.2019.8916841

[31] S. Kanajan, C. Pinello, Haibo Zeng, and A. Sangiovanni-Vincentelli. 2006. Exploring trade-off's between centralized versus decentralized automotive architectures using a virtual integration environment. In *Proceedings of the Design Automation Test in Europe Conference*, Vol. 1. 1–6. https://doi.org/10.1109/DATE.2006.243895

[32] Maged Khalil. 2019. Improving solution reuse in automotive embedded applications using a pattern library based approach. In *Proceedings of the 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 653–659. https://doi.org/10.1109/MODELS-C.2019.00100

[33] Stefan Kugele, Vadim Cebotari, Mario Gleirscher, Morteza Farzaneh, Christoph Segler, Sina Shafaei, Hans-Jörg Vögel, Fridolin Bauer, Alois Knoll, Diego Marmsoler, and Hans-Ulrich Michel. 2017. Research challenges for a future-proof E/E architecture - a project statement. In *INFORMATIK 2017*, Maximilian Eibl and Martin Gaedke (Eds.). Gesellschaft für Informatik, Bonn, 1463–1474. https://doi.org/10.18420/in2017_146

[34] Stefan Kugele, David Hettler, and Jan Peter. 2018. Data-centric communication and containerization for future automotive software architectures. In *Proceedings of the 2018 IEEE International Conference on Software Architecture (ICSA)*. 65–6509. https://doi.org/10.1109/ICSA.2018.00016

[35] Longmei Li, Iryna Yevseyeva, Vitor Basto-Fernandes, Heike Trautmann, Ning Jing, and Michael Emmerich. 2017. Building and using an ontology of preference-based multiobjective evolutionary algorithms. In *Proceedings of the 9th International Conference on Evolutionary Multi-Criterion Optimization - Volume 10173* (Münster, Germany) *(EMO 2017)*. Springer-Verlag, Berlin, Heidelberg, 406–421. https://doi.org/10.1007/978-3-319-54157-0_28

[36] Andreas Lock, Nigel Tracey, and Detlef Zerfowski. 2020. Entering new worlds. New E/E architectures with vehicle computers offer new opportunities (ETAS). *RealTimes* 2019/2020 (2020), 6–9. https://www.etas.com/en/company/realtimes_2019_2020-entering-new-worlds-new-e-e-architectures-with-vehicle-computers.php.

[37] Tieu Long Mai and Nicolas Navet. 2021. Improvements to deep-learning-based feasibility prediction of switched ethernet network configurations. In *Proceedings of the 29th International Conference on Real-Time Networks and Systems* (NANTES, France) *(RTNS'2021)*. Association for Computing Machinery, New York, NY, USA, 89–99. https://doi.org/10.1145/3453417.3453429

[38] Tieu Long Mai and Nicolas Navet. 2021. Deep learning to predict the feasibility of priority-based ethernet network configurations. *ACM Trans. Cyber-Phys. Syst.* 5, 4, Article 45 (sep 2021), 26 pages. https://doi.org/10.1145/3468890

[39] K Miettinen. 2012. *Nonlinear Multiobjective Optimization*. Vol. 12. Springer, Berlin.

[40] Timo Möller, Asutosh Padh, Dickon Pinner, and Andrea Tschiesner. 2019. The future of mobility is at our doorstep - compendium 2019/2020 (McKinsey center for future mobility). (Dec. 2019). https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-future-of-mobility-is-at-our-doorstep.

[41] Ahmed Nasrallah, Akhilesh S. Thyagaturu, Ziyad Alharbi, Cuixiang Wang, Xing Shao, Martin Reisslein, and Hesham ElBakoury. 2019. Ultra-low latency (ULL) networks: The IEEE TSN and IETF detnet standards and related 5G ULL research. *IEEE Communications Surveys Tutorials* 21, 1 (2019), 88–145. https://doi.org/10.1109/COMST.2018.2869350.

[42] V. Navale, K. Williams, A. Lagospiris, M. Schaffert, et al. 2015. (R)evolution of E/E architectures. *SAE Int. J. Passeng. Cars - Electron. Electr. Syst. 8(2)* (2015). https://doi.org/10.4271/2015-01-0196

[43] N. Navet, T.L. Mai, and J. Migge. 2019. Using machine learning to speed up the design space exploration of ethernet TSN networks. http://hdl.handle.net/10993/38604.

[44] Pierluigi Nuzzo, Nikunj Bajaj, Michael Masin, Dmitrii Kirov, Roberto Passerone, and Alberto L. Sangiovanni-Vincentelli. 2020. Optimized selection of reliable and cost-effective safety-critical system architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (2020), 2109–2123. https://doi.org/10.1109/TCAD.2019.2963255

[45] Alessandro Pinto, Alvise Bonivento, Alberto Sangiovanni-Vincentelli, Roberto Passerone, and Marco Sgroi. 2006. System level design paradigms: Platform-based design and communication synthesis. *ACM Trans. Design Autom. Electr. Syst.* 11 (01 2006), 537–563. https://doi.org/10.1145/996566.1142982

[46] Klaus Pohl, Harald Hönninger, Reinhold Achatz, and Manfred Broy. 2012. *Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology*. Springer, Heidelberg. https://doi.org/10.1007/978-3-642-34614-9

[47] V. Rupanov, C. Buckl, L. Fiege, M. Armbruster, A. Knoll, and G. Spiegelberg. 2014. Employing early model-based safety evaluation to iteratively derive E/E architecture design. *Science of Computer Programming* 90 (2014), 161–179. https://doi.org/10.1016/j.scico.2013.10.005 Special Issue on Component-Based Software Engineering and Software Architecture.

[48] SAE. 2021. SAE-J3016_202104. taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. (2021). https://www.sae.org/standards/content/j3016_202104.

[49] Adam Schnellbach. 2016. *Fail-operational automotive systems*. Ph. D. Dissertation. Graz, TU, Graz,, Austria. Advisor(s) Mario Hirz.

[50] Bernhard Schätz, Vincent Aravantinos, Sebastian Voss, Sabine Mavin, and Florian Hözl. 2015. AutoFOCUS 3: Tooling concepts for seamless, model-based development of embedded systems.

[51] Bernhard Schätz, Sebastian Voss, and Sergey Zverlov. 2015. Automating design-space exploration: Optimal deployment of automotive SW-components in an ISO26262 context. In *Proceedings of the 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/2744769.2747912

[52] Stephan Sommer, Alexander Camek, Klaus Becker, Christian Buckl, Andreas Zirkler, Ludger Fiege, Michael Armbruster, Gernot Spiegelberg, and Alois Knoll. 2013. RACE: A centralized platform computer based architecture for automotive applications. In *Proceedings of the 2013 IEEE International Electric Vehicle Conference (IEVC)*. 1–6. https://doi.org/10.1109/IEVC.2013.6681152

[53] Matthias Traub, Alexander Maier, and Kai L. Barbehön. 2017. Future automotive architecture and the impact of IT trends. *IEEE Software* 34, 3 (2017), 27–32. https://doi.org/10.1109/MS.2017.69

[54] Underwriters Laboratories (UL). 2020. UL4600 - standard for evaluation of autonomous products. (2020).

[55] Humberto Luiz Valdivia de Matos, Adilson Marques da Cunha, and Luiz Alberto Vieira Dias. 2014. Using design patterns for safety assessment of integrated modular avionics. In *Proceedings of the 2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC)*. 4D1–1–4D1–13. https://doi.org/10.1109/DASC.2014.6979473

[56] Josetxo Villanueva, Jörn Migge, and Nicolas Navet. 2021. QoS-predictable SOA on TSN: Insights from a case-study. *Automotive Ethernet Congress 2021* (feb 2021). http://hdl.handle.net/10993/46285.

[57] Christopher B. Watkins and Randy Walter. 2007. Transitioning from federated avionics architectures to integrated modular avionics. In *Proceedings of the 2007 IEEE/AIAA 26th Digital Avionics Systems Conference*. 2.A.1–1–2.A.1–10. https://doi.org/10.1109/DASC.2007.4391842

[58] Y.C. Yeh. 1996. Triple-triple redundant 777 primary flight computer. In *Proceedings of the 1996 IEEE Aerospace Applications Conference. Proceedings*, Vol. 1. 293–307 vol.1. https://doi.org/10.1109/AERO.1996.495891

[59] Sergey Zverlov, Maged Khalil, and Mayank Chaudhary. 2016. Pareto-efficient deployment synthesis for safety-critical applications in seamless model-based development. In *Proceedings of the 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*.

[60] Sergey Zverlov and Sebastian Voss. 2014. Synthesis of pareto efficient technical architectures for multi-core systems. In *Proceedings of the IEEE 38th Annual International Computers, Software, and Applications Conference Workshops, COMPSACW 2014*. https://doi.org/10.1109/COMPSACW.2014.63

Received 20 June 2022; revised 6 November 2022; accepted 4 January 2023