TECHNISCHE UNIVERSITÄT WIEN

TU VIENNA

Fail-Operational Strategies for Highly-Integrated Automotive ECUs

DIPLOMA THESIS

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Privatdoz. Dipl.-Ing. Dr. techn. Wilfried Steiner

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology
Institute of Computer Engineering

by
Rupert Schorn
Matriculation number 01325700

Vienna, January 2021

Institute of Computer Engineering (E191)
A-1040 Wien, Treitlstraße 3, Internet: https://ti.tuwien.ac.at

Acknowledgements

I would like to thank Privatdoz. Dipl.-Ing. Dr.techn. Wilfried Steiner for the supervision of this thesis. Furthermore, I want to thank Dr. Anna Ryabokon for her support, especially for providing all the necessary infrastructure the practical part of this thesis is based on.

This work was funded by the ECSEL JU project PRYSTINE which receives grant from the European H2020 research and innovation programme through the Electronic Component Systems for European Leadership (ECSEL) Joint Undertaking under grant agreement No 783190. And National Authorities such as the Austrian Ministry for Transport, Innovation and Technology under the funding programme „IKT der Zukunft“(Austrian national funding for ECSEL, grant agreement no 865307) and from the German Federal Ministry of Education and Research (BMBF funding, grant agreement no 16ESE0329).

Abstract

The ongoing transition towards fully automated systems for on-road vehicles requires highly available and reliable Electrical and/or Electronic (E/E) systems. When an Automated Driving System (ADS) is supposed to monitor the driving environment, Fail-Operational (FO) systems providing high availability and reliability need to be developed. If a single failure occurs, specific systems must remain operational for a certain amount of time to either hand over control to a human driver or automatically manoeuvre the vehicle into a safe state. Various approaches on architecture level exist to develop systems capable of FO behaviour. Depending on the application, some approaches are more qualified compared to others. However, it is always a trade-off between the degree of redundancy and diversity added on hardware or software level and economic requirements.

This thesis is based on state-of-the-art approaches to build FO ADSs and on a detailed analysis. For this purpose, e.g. failure modes, Automotive Safety Integrity Levels (ASILs) or Fault Containment Regions (FCRs) are considered. Several design concepts were elaborated in this thesis. Based on the evaluation of the proposed concepts, one of them was selected for the implementation because it addresses the main challenges defined by automotive industry. The implemented FO concept was successfully integrated in a lab demonstrator using existing Electronic Control Units (ECUs), containing proper safety controllers, number crunching processors and communication links. The FO behaviour is demonstrated by using fault injection. Finally, the overall performance of the demonstrator is evaluated by measuring the fail-over time of the system which shows acceptable performance for real-world application scenarios.

Kurzzusammenfassung

Der derzeit stattfindende technologische Übergang zu voll autonomen Systemen für Straßenfahrzeuge erfordert E/E-Systeme mit entsprechend hohen Verfügbarkeits- und Zuverlässigkeitskennzahlen. Die Überwachung der Umgebung durch autonome Systeme erfordert die Entwicklung fehlertoleranter Systeme, die den notwendigen Grad an Verfügbarkeit und Zuverlässigkeit liefern. Beim Auftreten eines einzelnen Fehlers müssen gewisse Systeme zumindest für eine gewisse Zeitspanne weiterhin funktionsfähig bleiben, um entweder die Kontrolle an den Fahrer zu übergeben oder das Fahrzeug in einen sicheren Zustand zu manövrieren. Für die Entwicklung von fehlertoleranten Systemen existieren verschiedene Architekturen, wobei deren Verwendbarkeit anwendungsspezifisch ist und immer ein Kompromiss zwischen dem Grad der Redundanz bzw. Diversität und den wirtschaftlichen Vorgaben gefunden werden muss.

In dieser Arbeit liegt der Fokus auf einer State-of-the-Art-Recherche und einer detaillierten Untersuchung von fehlertoleranten autonomen Systemen. Dabei werden unter anderem Konzepte wie Fehler-Modes, ASILs oder FCRs berücksichtigt. Von den untersuchten Architekturen wird ein ausgewählter Ansatz mithilfe von existierenden ECUs, bestehend aus Safety-Prozessoren, Number-Crunching-Prozessoren und geeigneten Kommunikationsverbindungen in Form eines Prototypen implementiert. Mithilfe von Fehlerinjektionen wird das Fehlertoleranzverhalten demonstriert und dessen Leistungsfähigkeit durch Messung der Fail-Over-Zeit evaluiert. Die Auswertung der Messergebnisse zeigt akzeptable Werte für den Einsatz in realen Anwendungsszenarien.

Contents

List of Figures

List of Tables

1 Introduction

One of the present major challenges affecting multiple disciplines in the automotive domain is related to the deployment of systems targeting driving automation. In terms of Electrical and/or Electronic (E/E) systems this comes along with the current revolution in the automotive industry regarding the ongoing transition from fail-safe to fail-operational systems. The associated challenges are faced in evolving new technologies, but also in establishing new development methods and processes besides considering social aspects (e.g. human behaviour) [1].

1.1 Motivation

Enabling automated driving demands E/E systems with stringent dependability requirements. Besides fault prevention, fault removal and fault forecasting, fault tolerance is an inevitable measure to eventually satisfy those requirements [2]. In the aerospace industry fault-tolerant systems are already successfully deployed by utilizing suitable Fail-Operational (FO) architectures. Mainly due to economic reasons, i.e. optimizing the costs, those existing FO architectures can't simply be transferred into the automotive domain [3]. However, the required Mean Time to Failure (MTTF) of $10^9 h$ must be satisfied, or even outperformed because of higher exposure times in the automotive domain compared to the aerospace domain [4].

Fundamental concepts of all FO architectures are applying redundancy and diversity on Hardware (HW) (mechanical and electrical components) and Software (SW) (e.g. information processing) level. Contrary to the aerospace domain, only single faults, potentially leading to hazardous events, must be tolerated [5]. This is because vehicles can be manoeuvred into a safe state usually easier and faster compared to an aircraft.

Considering all these aspects, FO architectures to be applied on automotive Electronic Control Units (ECUs) suitable to host automated driving applications must be defined. In literature various FO architectures are proposed (see Chapter 2), but still technical challenges to satisfy safety levels required by applicable standards must be overcome.

1.2 Standards

Over the last years a couple of standards related to driving automation were published, targeting various aspects in the automated driving domain. However, the standardization process is still ongoing and the current (released) standards do not satisfy the required

maturity to deploy autonomous driving systems on a profound basis. Some of relevant standards are listed in Table 1.1.

Standard	Description
SAE J3016_201806	Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles [6]
ISO/PAS 21448:2019	Road vehicles - Safety of the intended functionality [7]
ISO 26262:2018	Road vehicles - Functional safety [8]
ISO/SAE DIS 21434	Road vehicles - Cybersecurity engineering [9]
ISO/CD TR 4804	Road vehicles - Safety and cybersecurity for automated driving systems - Design, verification and validation methods [10]

Table 1.1: Standards related to automated driving

The research and development performed in this thesis mostly are based on the SAE J3016 taxonomy [6] and on the ISO 26262 standard [8]. Relevant parts of those standards are described in the following sections and in the description of the applied methodology for the architecture investigations (see Section 3.1).

1.2.1 SAE J3016

A major reference for all stakeholders involved in the automated driving domain is the *J3016* taxonomy [6]. It is published and maintained by SAE International [11] and provides a common understanding, terminology and classification related to driving automation systems. The most important terms, definitions and classifications related to this thesis are summed up in the following subsections.

Terminology

The following definitions are based on J3016 document released by SAE [6]:

A ***Dynamic Driving Task (DDT)*** refers to the vehicle operation in on-road traffic. Examples for DDT are lateral or longitudinal vehicle motion control.

A ***driving automation system***, composed of hardware and software capable of performing at least parts of a DDT, is used to reference systems of driving automation levels 1-5.

Compared to a driving automation system, an ***Automated Driving System (ADS)*** targets automation levels 3-5 and therefore performs entire DDTs.

Operational conditions (e.g. environmental, geographical, time-of-day restrictions, traffic characteristics, ...) that a driving automation system is capable of by design is defined

as an ***Operational Design Domain (ODD)***.

After a DDT fallback was performed, the risk of having a crash is reduced by bringing the vehicle into a condition called ***Minimal Risk Condition (MRC)***. This condition varies according to the driving automation level, the system failure or the ODD.

If a DDT performance relevant system failure occurs or the ODD is exited, the user or the ADS must either continue the DDT or achieve a MRC, depending on the driving automation level of the feature. This response is referred to as ***DDT fallback***.

The environment monitoring (detection, recognition, classification, etc.) and appropriate response calculation (if needed) related to the DDT is referred to as ***Object and Event Detection and Response (OEDR)***.

Classification of driving automation levels

In J3016 [6] SAE defines 6 (0-5) descriptive and informative levels of driving automation. They can be considered as minimum requirements a driving automation system must be capable of to be assigned to a distinct level. The categorization is performed according to the following parameters:

- driving automation system performs longitudinal and/or lateral motion control
- driving automation system performs OEDR
- driving automation system performs DDT fallback
- driving automation system is restricted by an ODD

According to these parameters, Table 1.2 summarizes the defined levels of driving automation according to [6].

The driving automation levels are mutually exclusive on feature level, i.e. a feature can only be correlated to one distinct level. But a driving automation system is able to contain multiple features assigned to different driving automation levels. Each feature in a driving automation system is the implementation of a usage specification, which is the combination of a driving automation level and an ODD [6]. The level of driving automation must be considered from the very beginning when developing any driving automation system. There is no test that might be able to verify a specific level of driving automation for a given feature, so these levels are assigned and not measured [6].

DDT fallback taxonomy

The responsibility for performing the DDT fallback depends on the level of driving automation (see Table 1.2). Switching to a DDT fallback must be executed in case a (DDT performance-relevant) system failure or an out-of-ODD condition (i.e. a defined ODD for the engaged feature is left) occurs. In that situation the driving automation system is not

Level	Name	Narrative definition	Sustained lateral and longitudinal vehicle motion control	OEDR	DDT fallback	ODD
Driver performs part or all of the DDT						
0	No Driving Automation	The performance by the driver of the entire DDT, even when enhanced by active safety systems.	Driver	Driver	Driver	n/a
1	Driver Assistance	The sustained and ODDspecific execution by a driving automation system of either the lateral or the longitudinal vehicle motion control subtask of the DDT (but not both simultaneously) with the expectation that the driver performs the remainder of the DDT.	Driver and System	Driver	Driver	Limited
2	Partial Driving Automation	The sustained and ODDspecific execution by a driving automation system of both the lateral and longitudinal vehicle motion control subtasks of the DDT with the expectation that the driver completes the OEDR subtask and supervises the driving automation system.	System	Driver	Driver	Limited
ADS ("System") performs the entire DDT (while engaged)						
3	Conditional Driving Automation	The sustained and ODD-specific performance by an ADS of the entire DDT with the expectation that the DDT fallback-ready user is receptive to ADS-issued requests to intervene, as well as to DDT performance-relevant system failures in other vehicle systems, and will respond appropriately.	System	System	Fallback-ready user (becomes the driver during fallback)	Limited
4	High Driving Automation	The sustained and ODD-specific performance by an ADS of the entire DDT and DDT fallback without any expectation that a user will respond to a request to intervene.	System	System	System	Limited
5	Full Driving Automation	The sustained and unconditional (i.e., not ODD-specific) performance by an ADS of the entire DDT and DDT fallback without any expectation that a user will respond to a request to intervene.	System	System	System	Unlimited

Table 1.2: Driving automation levels [6]

able to perform the entire DDT anymore and depending on the driving automation level further actions are initiated [6]:

Level 1-2: The driving automation feature stops its operation and the human driver continues the DDT [6].

Level 3: The ADS continues performing the DDT "for at least several seconds" [6, p. 7]. The DDT fallback-ready user intervenes and either continues performing the DDT or achieves a MRC if necessary. This decision is up to the user. If the user fails to take over control when prompted, a failure mitigation strategy can be initiated by the ADS. A failure mitigation strategy could be e.g. to stop a vehicle in the present traffic lane and turn on hazard lamps [6]. A use case sequence for Level 3 fallback including failure mitigation strategy is shown in Figure 1.1.

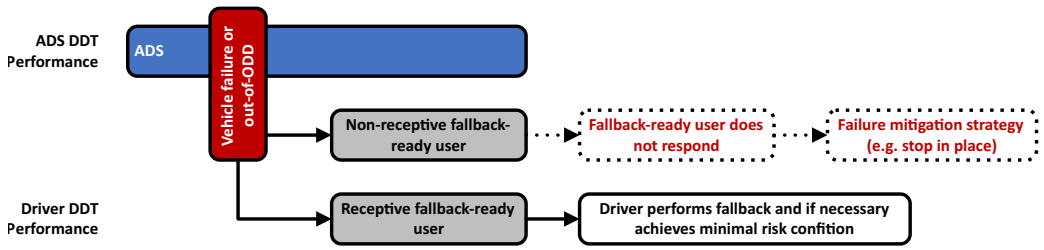

Figure 1.1: Level 3 feature fallback use case sequence [6]

Level 4-5: DDT fallback is performed by the ADS and a MRC must be achieved. Design limitations (e.g. low speed) are allowed for the fallback and are referred to as limp-home mode. If the vehicle is drivable, the user alternatively might choose to continue the DDT manually. In addition to the automatic DDT fallback, a failure mitigation strategy can be defined to react to rare, catastrophic events. A possible failure mitigation strategy in this case could be a stop-in-place manoeuvre [6]. Figure 1.2 shows a use case sequence for a DDT fallback including a failure mitigation strategy for a level 4 feature.

The J3016 taxonomy does not define specific time intervals (e.g. related to DDT fallback) that must be satisfied to comply with a distinct level of driving automation. General phrases like "timely manner", "timely request" or "sufficient time for a typical person to respond appropriately to the driving situation at hand" are used instead [6].

1.2.2 ISO26262 Road vehicles - Functional safety

The *ISO 26262 Road vehicles - Functional safety* standard [8] is an adaptation of the generic functional safety standard *IEC 61508 Functional safety of electrical/electron-*

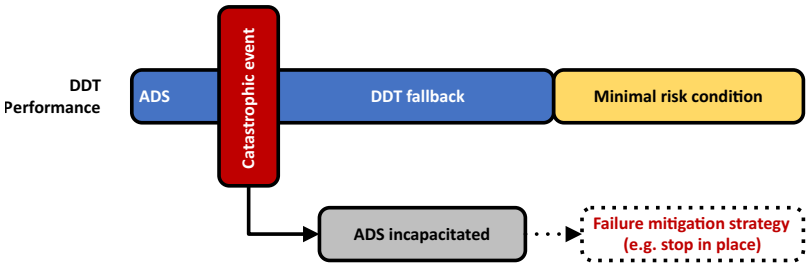

Figure 1.2: Level 4 feature fallback use case sequence [6]

ic/programmable electronic safety-related systems [12]. It covers multiple aspects related to functional safety of E/E systems in the automotive domain throughout the complete safety lifecycle (e.g. management, concept, development, production, operation). Some (simplified) processes and artefacts of the *ISO 26262* standard are applied in Chapter 3. However, e.g. safety analysis methods related to the ongoing transition from fail-safe to fail-operational systems are not covered sufficiently [1]. Furthermore, the terminology of the standard is vague and misleading with respect to automated driving [13]. The ambiguity of developing ADSs caused by the limitations of *ISO 26262* might be eliminated by the upcoming standard *ISO/CD TR 4804* [10].

1.3 Scope of the thesis

The main goal of this thesis is a comparison of selected FO architectures potentially suitable to be deployed on automotive ECUs for hosting Automated Driving Features (ADFs), based on a state-of-the-art research. The selected architectures are investigated in detail, applied on a hypothetically defined ADS and finally evaluated and compared. One of the selected approaches is implemented in a prototypical ADS to demonstrate the DDT fallback (also referred to as fail-over mechanism). The implemented ADS prototype must ensure that the actuator consistently applies the output either from the primary or the fallback node. If this is not already ensured implicitly by the selected FO architecture, proper voting and signalling mechanisms must be implemented on the ADS.

It is explicitly not a goal of this thesis to develop a series proven FO system.

1.4 Structure

This thesis is structured as follows: At first, a research study about state-of-the-art, i.e. existing FO architectures, fail-over mechanisms and autonomous driving platforms was performed and is described in Chapter 2. Based on these research results, three architectures were investigated in detail in Chapter 3, following a defined methodology.

The 1-out-of-2 with Diagnostics (1oo2D) FO architectures was selected, implemented in a prototypical ADS and evaluated as described in Chapter 4. Finally, in Chapter 5 the results are compared and limitations as well as improvements regarding the FO concepts are summed up. Appendix A contains a Hazard Analysis and Risk Assessment (HARA) on the hypothetical ADS the investigations are based on.

2 State of the art

The current state-of-the-art of different aspects related to FO strategies is covered in this chapter. Existing system, HW and SW architectures are described in Section 2.1, followed by switch over handling from a primary to a fallback unit (DDT fallback or fail-over mechanism) in Section 2.2. Finally, Section 2.3 contains existing autonomous driving HW and SW platforms to be used for prototyping or series projects.

2.1 Fail-operational architectures

FO architectures aim at improving a system's reliability by adding redundancy [14]. Several concepts already exist, all of them apply redundancy and diversity in a slightly different way. Commonly utilized FO architectures in the automotive domain are two-channel approaches. Most of the presented concepts focus on HW redundancy, however, some of them reduce hardware redundancy by adding redundancy on software level [15].

2.1.1 Triple Modular Redundancy (TMR)

An early Fault-Tolerance (FT) approach, originally applied in the space and military domains, is TMR, proposed e.g. in [14]. Three identical modules (e.g. computers or less complex systems) provide their outputs to a voter that forwards a majority opinion based selected result to a consuming component (see Figure 2.1). The system does not fail as long as none or only one module fails. In [14], the reliability of a TMR system is described in detail.

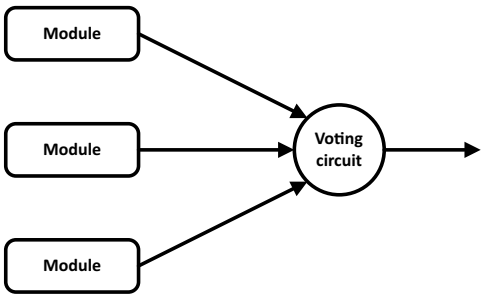

Figure 2.1: Original proposed TMR architecture [14]

The TMR approach is further applied in the aerospace domain, as it is utilized in e.g. the Boeing 777 flight computer (computing system, aeroplane electrical power, hydraulic

power and communication path) [16]. In this proposed architecture, additional diversity measures are applied to the system (n-version dissimilarity issue).

As besides fly-by-wire systems in the aerospace domain also drive-by-wire systems in the automotive domain evolved, the TMR approach is used in safety critical automotive applications, too. E.g. in [17], TMR is used for fault tolerant multiprocessor architectures, where three identical Central Processing Units (CPUs) are forwarding their results to a majority voter. Another approach according to [17] is to apply the TMR pattern on a heterogeneous architecture, consisting of three loosely coupled cores. Another denotation for the TMR approach in the automotive domain is 2-out-of-3 (2oo3) [18].

Conflicting requirements of the TMR architecture are sufficient independency of the Fault Containment Regions (FCRs) and output data consistency, since a certain amount of coordination among the FT entities is needed that adds dependencies in return [3].

2.1.2 Simplex architecture

The Simplex architecture originally was developed to enable online upgrades of real-time SW applications in the control engineering domain [19]. This is achieved by combining a reliable controller, one or more high performance controllers and a switch (see Figure 2.2), denoted as Analytic Redundant Unit (ARU). The output of the high performance component(s) is used, unless a SW upgrade is initiated. In this case the (degraded) calculations are performed by the reliable controller, until the high performance controllers are online again. For safety critical applications the reliable controller and the switch must be deployed on a separate HW to guarantee non-interference between the reliable and high performance components. All components can be deployed on the same HW for non-safety critical applications instead [19].

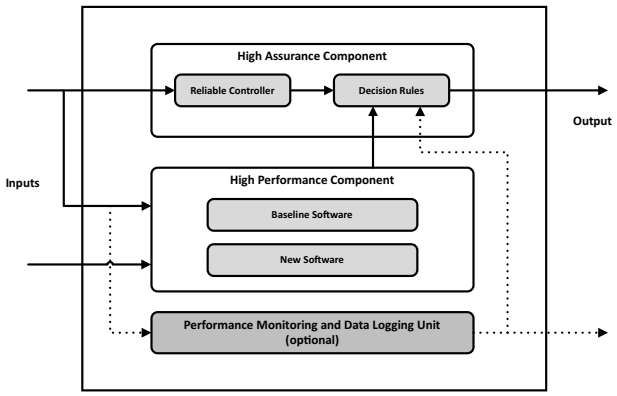

Figure 2.2: ARU of a simplex architecture [19]

In [20] a reference model for the Simplex architecture was proposed, considering the plant to be controlled, external context, machine, domain model and safety requirements.

In a case study with a fleet of remotely controlled cars, this Simplex reference model was applied.

To tolerate faults on application and system level (Real Time Operating System (RTOS), microprocessor), a system level Simplex approach using HW/SW co-design was published in [21]. In addition to the fault-tolerance on application level, the system level fault tolerance is achieved by deploying the safety controller and the decision module onto a dedicated processing unit and hence, isolate them from the high performance components. The defined fault model in [21] tolerates logical and resource sharing faults. Faults in the HW where the safety controller and the decision module are hosted on are assumed to be rare events. If these faults are not considered as rare events, e.g. a TMR approach can be used to mask faults in the safety components [21].

In [22] and [23], fault recovery is achieved via performing a restart of the full system, where the safety controller of the Simplex architecture is used for fault detection and triggering the restart. This approach requires that unavailability of the system caused by the restart does not result in a deadline miss of a critical task.

2.1.3 1oo2D

The 1oo2D architecture consists of two channels, however, only one of them sends the calculated results to the consuming entity (e.g. actuator) while the other acts as fallback. The two channels of the 1oo2D architecture are referred to as e.g. *primary mission controller* and *secondary mission controller* [24]. Both channels are implemented as fail-silent computing platforms [25], hence, the computing platforms must be equipped with diagnostic capabilities [26]. The primary mission controller is in charge of executing the intended functionality initially. In case a failure occurs on the primary channel, a mode transition to fail-silent is performed for this unit and the secondary mission controller starts taking over the operation [5].

The performance of the fault-detection mechanisms is essential for the dependability attributes of the overall system. Potential diagnostic methods are e.g. consistency checking, comparison of results with redundant modules or information redundancy. The secondary mission controller can either be configured in a hot-standby or cold-standby approach. In a hot-standby mode (unit is powered up even if primary mission controller is working correctly) a major drawback is wear-out, whereas for a cold-standby mode (unit is powered up after fault on primary unit is detected), startup time has to be taken into account [5]. A 1oo2D cold-standby architecture is shown in Figure 2.3.

To implement a FT system, usually it is not sufficient to just add redundant ECUs. The complete system has to be taken into account and analysed, hence, redundancy would be required for sensors, actuators and other contributing components. In [24] a system architecture based on the 1oo2D pattern is proposed. A hot standby approach is not considered to be feasible because it does not meet cost-effectiveness requirements in terms of a high energy consumption. Instead, the secondary mission controller is in standby

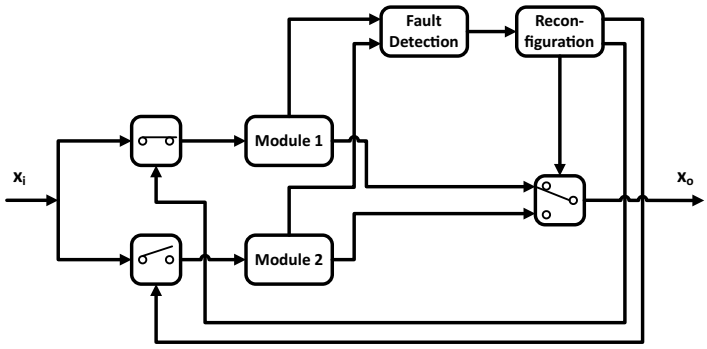

Figure 2.3: 1oo2D in a cold standby configuration [5]

mode, but a state and input data backup is provided to the redundant controller in order to fasten switch-over to the fallback in case a failure occurs on the primary mission controller. Once the secondary mission controller is activated the remaining operation time is reduced because of the missing fault-tolerance capability. Therefore it is used for fallback operations, e.g. reaching a safe state [24].

Another 1oo2D approach is proposed in [26]. Safety features like a lockstep mechanism and a hardware watchdog are suggested here to provide fail-safe behaviour for each channel. To significantly reduce the risk of common cause failures, diverse hardware platforms are used. An automotive case study has been carried out where diversity among the channels was achieved by using an Infineon Aurix safety controller for one channel and two ARM Microcontroller Units (MCUs) with software based lockstepping for the other channel [26].

Further examples where a 1oo2D architecture is supposed as a valid approach for enabling ADFs are mentioned in [25] and [27].

2.1.4 2-out-of-2 with Diagnostics (2oo2D)

The 2oo2D utilizes two channels, too, but compared to the 1oo2D approaches, the voting activity is shifted to the subsequent entity (e.g. actuator), hence both channels are sending their computed results. In [18] a 2oo2D approach is proposed which either can be implemented as fully redundant or as limp-home system. This pattern can be applied on MCU level as well as on core level [18].

Similar to the 1oo2D pattern, both channels are implemented as fail-silent units, hence the actuator is either controlled correctly or not at all. From long-term safety perspective it is important to know whether all units of the system still operate correctly or if a subsystem is faulty and therefore maintenance activities need to be triggered [28]. A

similar 2oo2D approach is also proposed in [29].

2.1.5 Hybrid architecture

In the aerospace domain, a high degree of redundancy and diversity is commonly accepted and used (see Section 2.1.1). This is not always done because of further increasing the availability of the systems, but also to extend the maintenance intervals, since a faulty unit doesn't necessarily have to be replaced immediately [30]. As an alternative to e.g. utilizing a TMR approach, architectural concepts using only two nodes may be the more suitable solution in the automotive domain, mainly due to economic reasons [30]. Full symmetric redundancy (as used in 1oo2D and 2oo2D) often is not necessary, since the emergency operation, activated once a fault occurred on the primary node, is only used for a short time period (to reach a safe state) [31]. A *hybrid* architectural approach, composed of a self monitored primary system and a limp-home system (see Figure 2.4) can be used as an alternative for deploying ADFs [28] [31].

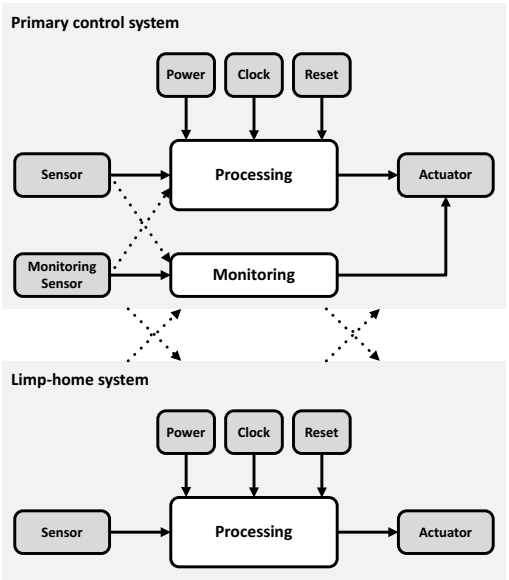

Figure 2.4: Hybrid redundancy approach [31]

The voting between the primary control and the limp-home system's output data can either be done by a separate voting component or on the actuator [31]. The hybrid architectural approach can be implemented on two independent ECUs or on the same board [18].

To deal with the dilemma of high performance, high Automotive Safety Integrity Levels (ASILs) and high cost efficiency, an application level software implemented fault detection is proposed in [31].

In [32] a Automated Driving System Computing Platform (ADS-CP) targeting Society of Automobile Engineers (SAE) levels 3+ is proposed based on a hybrid architecture (or optional a 2oo2D approach if full redundancy is used). The primary system is composed of two number crunching processors, one or two safety controllers (Infineon AURIX [33]) and a deterministic Ethernet switch [32]. The safety controller(s) are responsible for monitoring the system. A backup stand-by system provides redundant and diverse processing capabilities. It can either be designed as reduced (only one Performance Host (PH)) or fully redundant [32].

Another hybrid architecture, potentially suitable for targeting SAE level 4 applications, is proposed in [30]. On the main node a high diagnostic coverage is deployed to ensure fail silent behaviour in case a fault occurred. The result of two PHs are verified by a Safety Host (SH) and forwarded to the actuator in case the verification was successful. The fallback node is equipped with less safety features compared to the main node and always sends its calculated results to the actuator. The voting activity in this approach is shifted from the ADS-CP to the actuator. As long as the main node sends data, it is used by the actuator. Otherwise the data sent by the fallback node is selected by the actuator and vehicle operation continues in a degraded mode [30].

2.1.6 Fault-Tolerant Software architectures

Besides FT mechanisms implemented on system or HW level, various measures purely implemented on SW level exist. In [34] three different FT SW architectures are investigated and compared in terms of parameters like development effort, Real-Time (RT) behaviour or performance.

The first approach, analysed in [34], denoted as *Generic single-version architecture*, combines fault detection, fault containment and fault recovery mechanisms. A diagnostic Software Component (SWC) is responsible for detecting faults of the focused SWC. In case a fault was detected, the diagnostic SWC notifies an interface SWC, responsible for disabling the output of the monitored (faulty) SWC B. Further, a checkpoint memory is used to restore the state of the faulty SWC to a healthy state and therefore potentially enables SWC B to recover from the fault. Freedom from interference must be ensured between the focused SWC and the other SWCs, e.g. by applying memory partitioning mechanisms (see Figure 2.5).

The *n-version programming* approach utilizes n implementations of a SWC, based on the same specification, but using diverse designs and implementations. The outputs of the SWCs are masked by a voting SWC, selecting the outputs following a majority paradigm (see Figure 2.6) [34]. On system and HW level, this approach corresponds to the TMR approach, described in Section 2.1.1 (for $n = 3$).

Finally, the n-self-checking programming approach implements n diverse instances of a SWC, each instance separately monitored by a diagnostic SWC (see Figure 2.7). The diagnostic SWCs notify the individual health states to a voting SWC, which selects one healthy output and forwards it to the receiver SWC [34]. For $n = 2$, this approach is

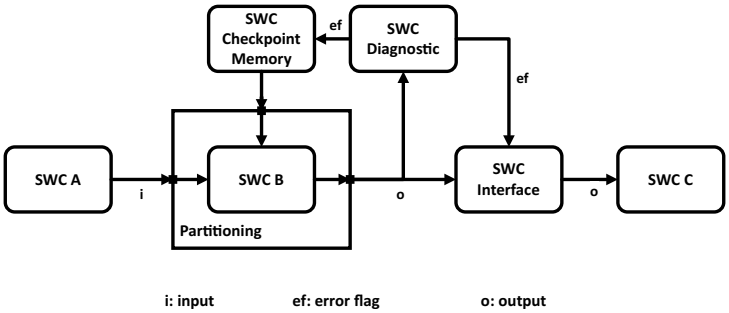

Figure 2.5: Generic single-version architecture [34]

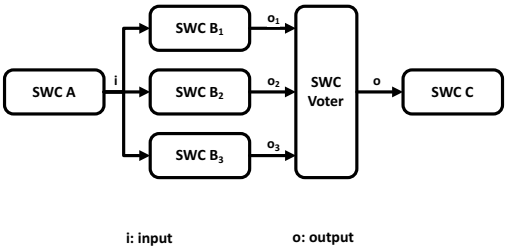

Figure 2.6: N-version programming [34]

similar to the 1oo2D approach on system and hardware level (see Section 2.1.3).

All presented FO SW architectures finally are compared with respect to parameters like development effort, RT behaviour or performance [34].

Another approach potentially contributing to enable FT capabilities on SW level is *coded processing*. In [15] the Safely Embedded Software (SES) approach is proposed to transform data and instruction codes into a coded value domain. Therefore an $AN + B$ code is utilized to add redundancy to the original code values to enable the detection of bit error(s), depending on the selected transformation parameters (e.g. hamming distance), as displayed in an exemplarily transformation in Figure 2.8.

The verification of the original and coded values can be performed e.g. at the end of each task cycle. Main disadvantages of the *coded processing* approach is a higher memory and processing power consumption [15].

2.2 Fail-over mechanisms

The switch-over from a primary unit to a fallback unit in case a failure occurred (DDT fallback) is denoted as fail-over mechanism in this thesis. This includes a failure detection

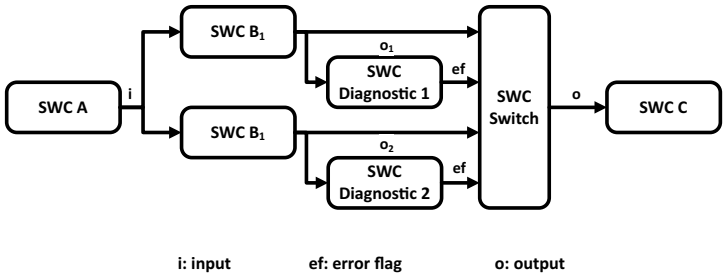

Figure 2.7: N-self-checking programming [34]

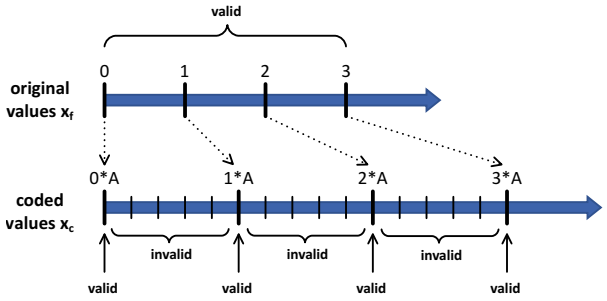

Figure 2.8: Exemplary transformation $x_c = A * x_f$ [15]

mechanism and mode switch activities (deactivating the faulty unit and activating a fallback unit). In literature various failure detection mechanisms are proposed for diverse application areas, which are covered here.

Currently most implementations of failure detectors are using an all-to-all communicational approach, where each node sends a *"I am alive"* message (heartbeat) to all other nodes [35]. Depending on the platform (e.g. number of nodes), this might not always be an efficient approach, hence a variety of failure detectors has been investigated and can be categorized based on a state of the art research according to Figure 2.9.

Depending on which node initiates the diagnostic status exchange, a push or a pull model is denoted [36]. For the pull approach (pinging), the observer node sends a dedicated *"Are you alive?"* message to the monitored node, which responds with a *"I am alive"* message in case no error occurs [37]. When the push model (heartbeat) is applied, the monitored nodes periodically send *"I am alive"* messages to the observers, without being pinged in advance [37]. Since the push approach requires a half number of messages, it is more efficient compared to the pull model [35].

The state categorization classifies failure detectors into stateful and stateless mecha-

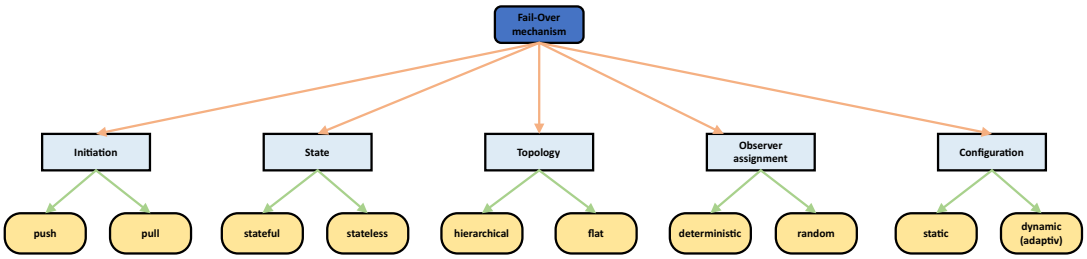

Figure 2.9: Exemplary categorization of fail-over mechanisms

nisms. The stateless approach uses individual messages to exchange diagnostic information, whereas the stateful detection mechanism monitors application specific messages to track the health state of participating nodes [38].

If a large number of nodes or processes need to be monitored, a hierarchical arrangement of the observed entities reduces the communication overhead by combining information [36]. Dividing the entities into groups, where a group member only monitors all other members in the same group and one dedicated group leader communicates with all other group leaders, is an example of a two-level hierarchical organization to reduce the amount of traffic [35]. Hence, the topology categorization distinguishes between a flat and a hierarchical approach.

The assignment of monitored entities to an observer can be done deterministic or random. Gossip style protocols are an example of a random selection of nodes to be monitored [36].

Another classification of the fail-over mechanisms is related to the persistence of the entity assignment to observers. A static configuration does not change the assignment during runtime, whereas a dynamic (adaptive) approach is able to dynamically react on changing environmental conditions (e.g. network) [36].

An early approach of using a simple heartbeat mechanism to implement fault tolerant real-time systems is proposed in [39]. Two processors periodically and synchronously exchange heartbeats to indicate their health states. To determine the heartbeat period, a zero-missed-deadline and one-missed-deadline approach are proposed. Among other approaches in [40] the simple heartbeat mechanism is extended by applying a ping service in case a heartbeat was missed.

A failure detector for high performance computing applications is proposed in [35]. In this heartbeat based failure detector a virtual observation ring is established to reduce communication overhead. This topology needs to be reconnected once a node is suspected to be faulty.

Architectures for intra-host and inter-host failure detection are proposed in [36]. The intra-host architecture is composed of a failure detection module (realized as a daemon process) and a watchdog. Additionally, the modules to be monitored are using a specific library, providing an interface to the failure detection module. For the inter-host failure detection the failure detection modules directly exchange their state information, using a publish-subscribe communication mechanism.

The heartbeat based failure detection mechanism proposed in [41] is implemented on the application software layer. A warm standby mode or cold standby mode can be applied on the secondary unit in this approach. The heartbeats are exchanged via User Datagram Protocol (UDP) datagrams over Ethernet and additionally through a Universal Asynchronous Receiver-Transmitter (UART) channel. The approach targets wireless sensor network applications and the proposed implementation achieves an average failover time of $379ms$.

A stateful failure detection approach is the safety co-pilot, proposed in [42]. It performs application specific correctness checks to coordinate the operation of the primary and redundant units. Several verification modules were proposed and one of them (i.e. vehicle collision verification) was presented in detail.

2.3 Autonomous driving platforms

Due to economic reasons (reduce time to market, costs, etc.) various platforms suitable for executing ADFs evolved over the last years. These HW and SW platforms partly not just assist the development phases, but also target to be utilized in automotive series products.

More than 100 global members (e.g. Original Equipment Manufacturers (OEMs), Tier 1 suppliers) contribute to the Apollo open autonomous driving platform [43]. Additionally HW development platforms (e.g. computing units, Global Positioning System (GPS), Inertial Measurement Unit (IMU), camera, Radio Detection and Ranging (RADAR), Light Detection and Ranging (LIDAR)) a SW platform (e.g. RTOS, runtime framework, control, planning, perception) and cloud services (High-Definition (HD) map, Vehicle-to-Everything (V2X)) are provided, too. Current Apollo version 5.5 supports ADFs towards fully autonomous urban road driving, including 360° perception and upgraded prediction, planning and control modules [43].

The Nvidia Drive autonomous vehicle development platforms contain HW, SW and architectural contributions [44]. The Drive AGX developer kit (HW and SW) is used as in-vehicle Artificial Intelligence (AI) computer in the car reference architecture (Drive Hyperion). Further, a simulation platform (Drive Constellation) and a deep neural network training platform (Nvidia DGX) are part of the Drive platforms [44]. The architecture of the Nvidia Drive OS platform SW, utilized on the Drive AGX developer kit, is displayed

in Figure 2.10.

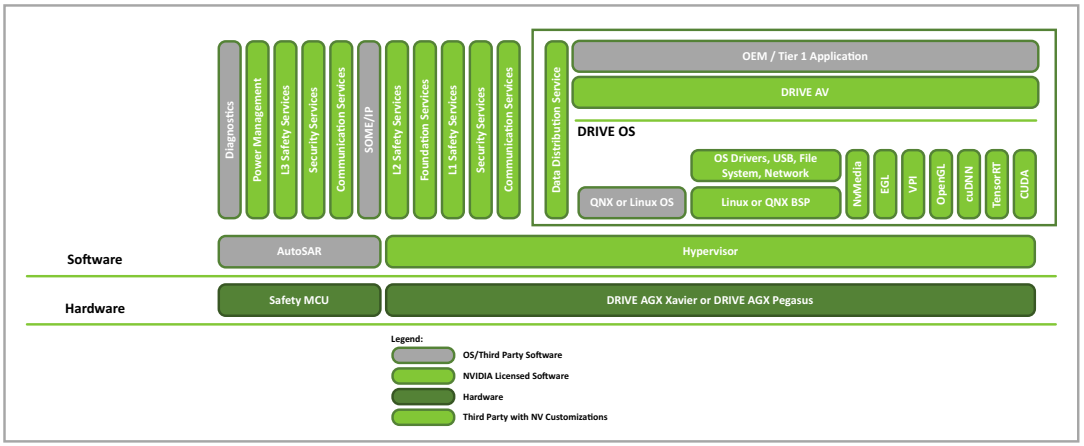

Figure 2.10: Architecture of Nvidia Drive OS [44]

The AUTomotive Open System ARchitecture (AUTOSAR) development partnership (more than 280 partners) provides a standardized software architecture for automotive ECUs [45]. A modular concept of the architecture abstracts the HW to increase SW reusability. The classic platform targets applications of safety integrity levels up to ASIL D, hosted on low computing power hosts, with high real-time requirements (micro seconds), while the adaptive platform is hosted on high computing power hosts with mid real-time requirements (milli seconds) and safety integrity levels of at least ASIL B [45]. Figure 2.11 shows the architecture of the Classic AUTOSAR stack.

Research results on the architecture of dependable distributed RT systems are summarized in [46]. These orthogonal concepts are based on a highly reliable and available global time base. The proposed architecture, denoted as Time-Triggered Architecture (TTA), targets any type of dependable distributed real-time system, independent of the application area or computational power of the nodes [46].

TTTech's safety software platform MotionWise utilizes concepts proposed in [46] among others [47]. It abstracts multiple hosts and provides e.g. RT, scheduling, communication and automotive (e.g. persistency) services via an AUTOSAR like Application Programming Interface (API) to the application layer [47].

An automated driving platform consisting of an ECU and MotionWise as software platform is TTTech's RazorMotion [48]. Multiple hosts (2x PH, 1x SH) provide different computing power levels and ASILs. They are connected via a deterministic Ethernet backbone and provide camera/display interfaces and automotive vehicle bus connectivity like Controller Area Network (CAN) [48].

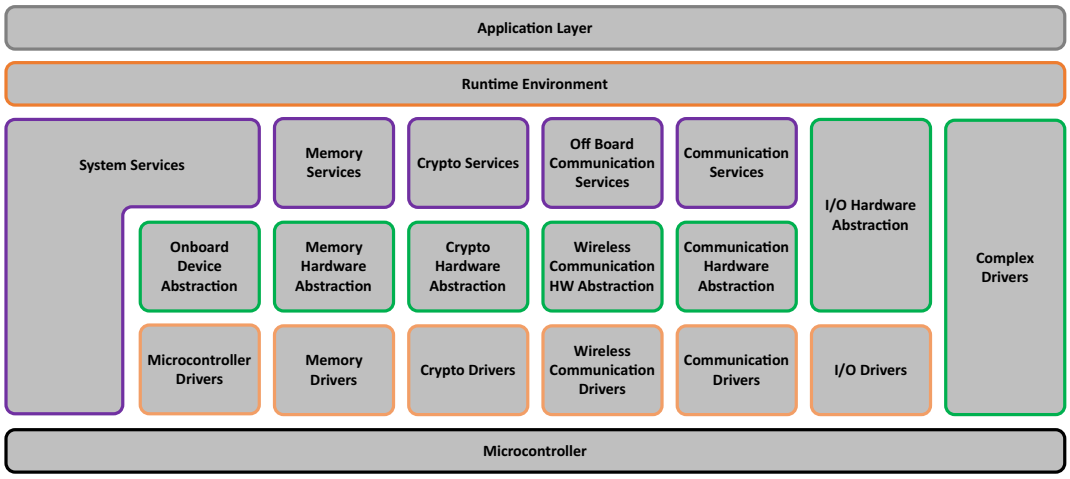

Figure 2.11: Architecture of the Classic AUTOSAR stack [45]

A cloud based storage and computing service solution is provided by Amazon AWS [49]. It further provides deep learning frameworks to support the development of autonomous vehicles [49].

AutonomouStuff automotive platform support the development of ADFs by providing vehicles equipped with suitable HW and SW components to enable e.g. development and customization of drive-by-wire applications [50].

3 FO architectures in the Automated Driving (AD) domain

This chapter contains investigations of the following three FO architectures:

- Simplex architecture
- 1oo2D architecture
- Hybrid architecture

These three FO architectures have been selected based on the discussion in Section 2.1. A major requirement for FO architectures is to avoid deadline misses of safety critical application tasks and therefore prevent dramatic hazards. This is conceptually illustrated in Figure 3.1, where a Quality of Service (QoS) parameter is defined as a function of missed deadlines. A missed deadline depends on the period of the application task and on the time interval between the occurrence of an error and operation takeover by the fallback, denoted as Fail-Over Time Interval (FOTI). If application task period and FOTI are equal, a QoS value of 1 is defined in this model. This is the assumed minimum value to be targeted for safety critical systems in the presented simplified approach, since a zero-missed deadline approach is considered to be essential. [1]

3.1 Methodology

Developing complex safety-critical systems while achieving required levels of dependability is not a trivial task. To harmonize the development processes and therefore enforce certain minimum quality levels, various standards, guidelines and methods have been developed over the last decades (see Section 1.2). In the automotive domain the ISO 26262 standard has become a major reference for dealing with functional safety of road vehicles [8]. However, a guideline of defining a fault-hypothesis for a safety-critical system is missing in the ISO 26262 standard. A fault-hypothesis is one powerful tool in designing safety-critical systems, and the guideline presented in [4] explains involved steps and defines its position in a complete design process. In the following sections both approaches are described briefly, benefits and drawbacks are analysed and a combined approach for further investigations in this chapter is defined.

[1] A zero-missed deadline approach is applied if any missed deadline of safety critical tasks potentially leads to a hazardous event [39].

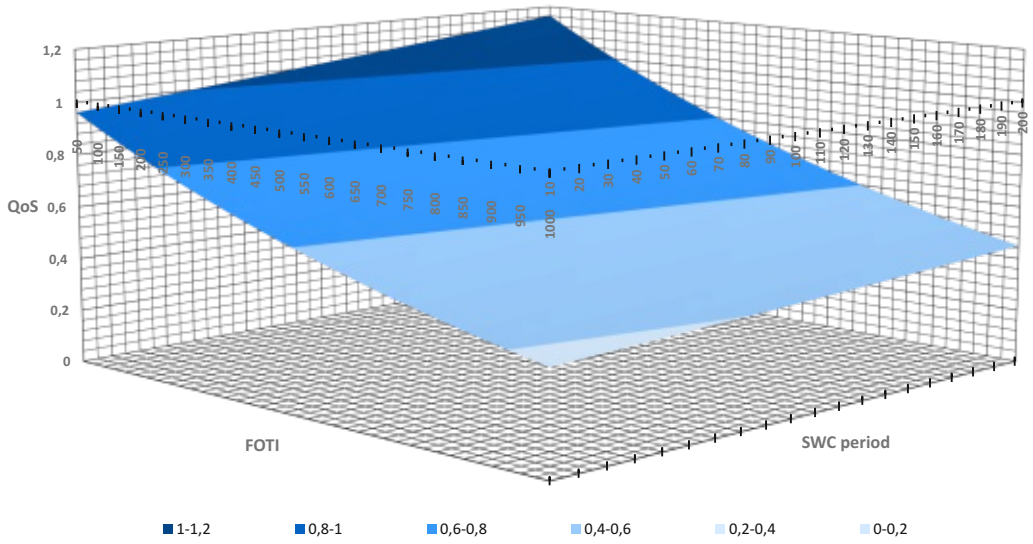

Figure 3.1: QoS as a function of FOTI and task period

3.1.1 Design according to ISO 26262

The ISO 26262 standard references functional safety of road vehicles' E/E systems and provides frameworks, references and tools to support activities during the whole automotive safety lifecycle [8].

A simplified extract of the workflow defined in ISO 26262 is shown in Figure 3.2. Starting point of the concept phase is the definition of the item as system (or a combination of systems), providing a function at vehicle level. Main inputs for this step are e.g. a system description (at vehicle level), requirements (legal, quality, performance, availability) or interfaces to the environment or other items [8].

Based on the results of the item definition, ISO 26262 suggests to perform a HARA. Its main goal is to identify possible hazardous events in case the behaviour of the item differs from the intended functionality. Safety goals, including associated ASILs are derived from the identified hazards, which can be seen as top level safety requirements [8].

Both mentioned work products (item definition and HARA) are input for the last step in the ISO 26262 concept phase: the definition of a Functional Safety Concept (FSC). A FSC specifies strategies on item level related to e.g. fault detection, fault tolerant mechanisms or functionality degradation to comply with the safety goals [8]. An additional input to the FSC is a System Architectural Design (SAD), which is not covered by this standard. FSC and SAD induce the next phases in ISO 26262 (Product development at system/hardware/software level), starting with the definition of a Technical Safety Concept (TSC) [8]. Since the focus of this chapter is on FO architectures at system

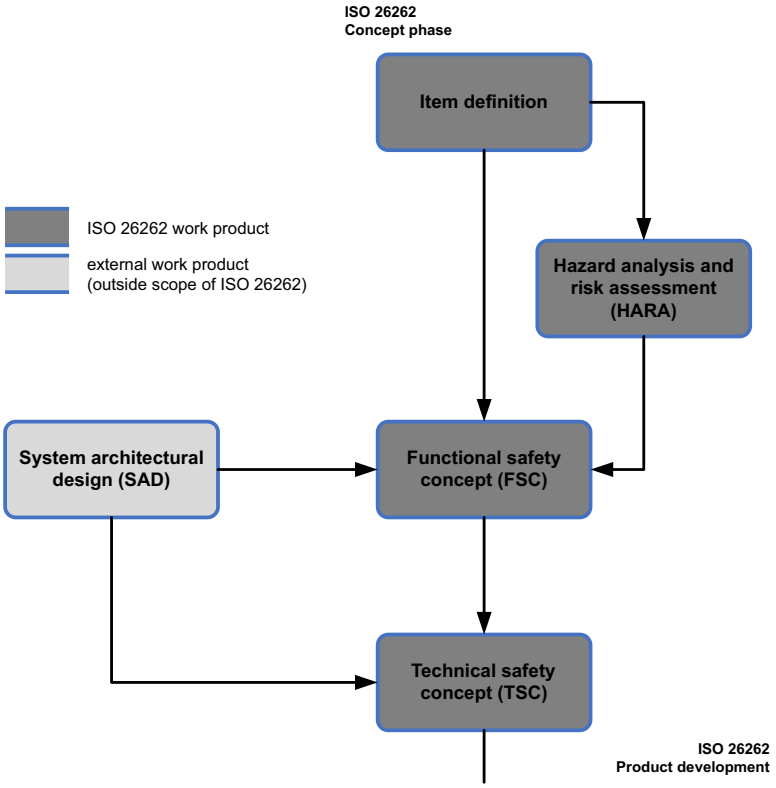

Figure 3.2: Simplified extract of design approach according to ISO 26262 [8]

level, those phases are not investigated on hardware and software level. In practice, the simplified uni-directional approach of Figure 3.2 is not feasible, therefore iterations and bi-directional transitions between the design steps are accepted.

In literature, several limitations of the ISO 26262 standard, especially regarding the design of FO systems, are documented. One limitation according to [51] is the focus at one item at a time. This is a problem in case functions of different items are tightly coupled to each other. Further limitations have been found regarding safety concepts or safety analysis methods especially for FO systems [1]. In [52], some limitations related to e.g. missing failure rate models or architectures to achieve certain ASILs for FO systems in the ISO 26262 standard are addressed.

3.1.2 Design based on a fault hypothesis

A design approach tailored for safety-critical real-time computer systems is presented in [4]. It has been published years before the ISO 26262 standard was released and does not provide that degree of details regarding description and tools/methods to be used. But still it consists some useful concepts that are missing in the ISO 26262 standard and

should be considered for the design of FO systems.

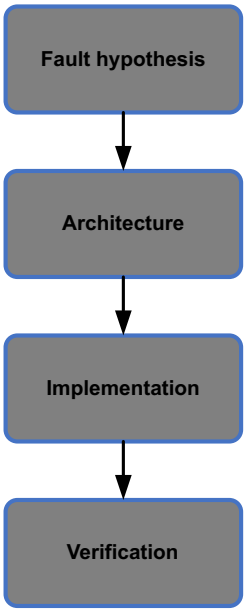

Figure 3.3:
Design approach based on a fault hypothesis [4]

Figure 3.3 provides an overview of the suggested phases to be performed for designing fault-tolerant systems. The fault hypothesis must be defined at the beginning. It includes [4]:

- separation of fault space into covered and uncovered faults (rare events)
- definition of FCRs
- specification of failure modes (assumptions)
- failure rate (assumptions)
- definition of detection and recovery timing

The results of the fault hypothesis are the main contributions to the architectural design in this approach. The architecture must be suitable to tolerate the defined covered faults, and also potentially react on uncovered faults, e.g. by implementing a Never-Give-Up (NGU) strategy [4]. Implementation and verification of the design are not treated in this thesis, since the main focus is on architecture level.

3.1.3 Combined fault-tolerant design approach

In order to further investigate three selected FO architectures, a methodology that integrates the *ISO 26262* (Section 3.1.1) and the fault-hypothesis approach (Section 3.1.2) is used. The item definition and HARA can be performed without any knowledge of the chosen architectural pattern, because in this phase the functionality of the system is defined, but not any details regarding architectural design or implementation [8]. The definition of the fault hypothesis already requires input regarding architectural design, because e.g. the definition of FCRs must fit to the component decomposition of the item. So the architectural patterns (Simplex, 1oo2D and Hybrid) are input for the fault hypothesis and the SAD. The non-generic phases (fault hypothesis, SAD, FSC) are highly depending on each other, so bidirectional transitions between them are unavoidable.

The common design tasks (item definition, HARA) are documented once in the following sections. Fault hypothesis, SAD and FSC are done for each chosen architecture separately.

Fault Hypothesis: The first step of defining a fault hypothesis is a decomposition of the given system into FCRs. It must be ensured, that the subsystems that form a FCR fail independently, since already very small correlations of the FCR's failure probabilities might have a significant impact on the overall dependability [4]. For the fault hypothesis

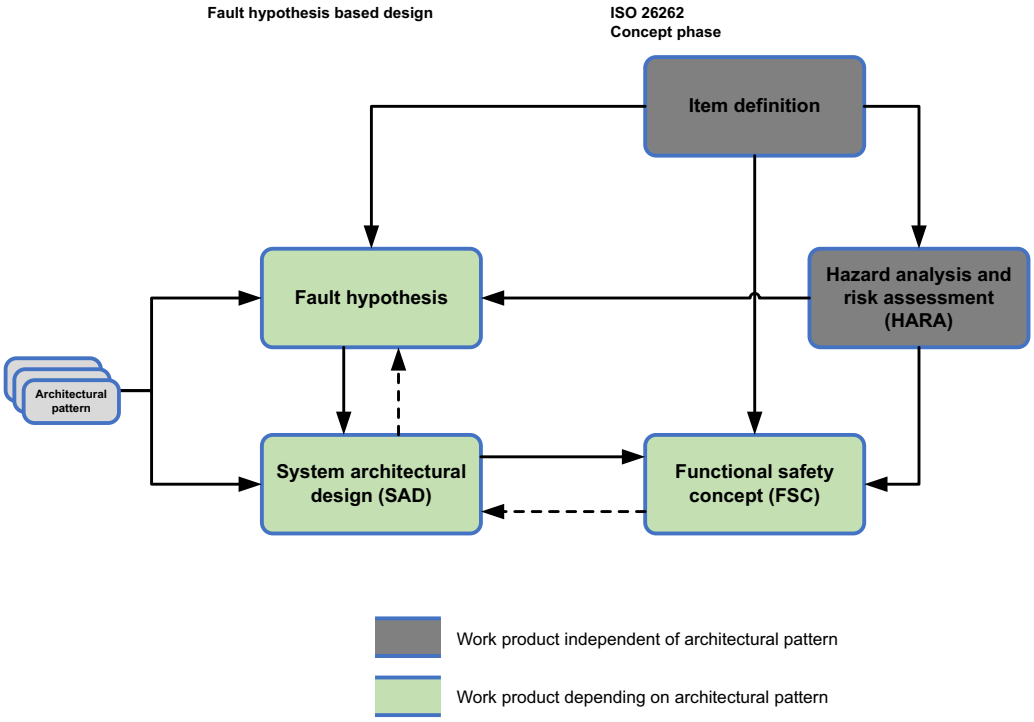

Figure 3.4: Combined fault-tolerant design approach

of the ADS-CP hardware and software faults are treated separately, and therefore the FCRs are defined differently for hardware and software investigations, as performed in [53].

SAD: The goal of the SAD is to provide a high level design on system level, considering resulting artefacts of previously performed steps as well as the FSC. The inputs used in this thesis are summarized in Figure 3.5.

Used artefacts from the item definition as inputs for the SAD are the system requirements and high level component decomposition (see Section 3.2). In addition to the system requirements, several assumptions related to the available (HW) components are made:

- Maximum ASIL of high computational power processors (referred to as PH): ASIL B
- Arbitrary number of diverse (and therefore common-mode fault-free) PHs are available
- Maximum ASIL of safety microcontrollers (referred to as SH): ASIL D

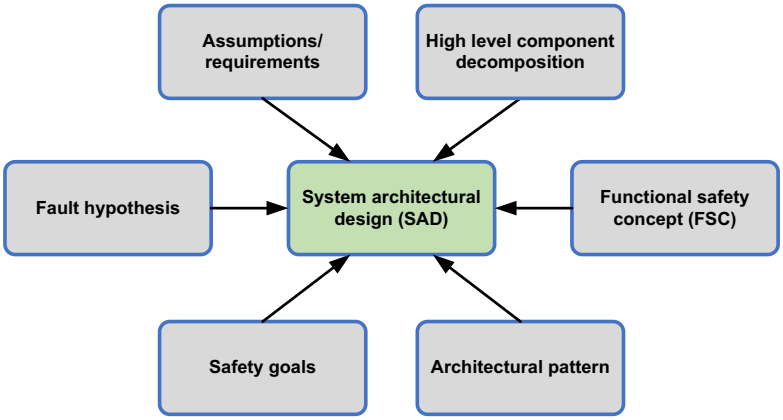

Figure 3.5: SAD input artefacts

- Arbitrary number of diverse (and therefore common-mode fault-free) SHs are available
- Maximum ASIL of peripheral components (power management, communication, ...): ASIL D

The safety goals (see Section 3.3) are used to determine the required ASILs for the HW and SW components, while the fault hypothesis mainly provides input regarding the required degree of independence between components.

The overall structure of the architectural design, hence the method how redundant units are combined, is determined by the used architectural pattern. Finally, the inputs provided by the FSC determine safety measures relevant for the system design to fulfil the safety goals. All the inputs are considered in the following investigations, separated into HW and SW architecture.

FSC: One of the FSC's main objectives is to define safety measures in order to fulfil the safety goals, which are top level requirements of the FSC. Further, the degraded functionality mode of the ADS-CP is specified[8]. The FSC is tightly coupled with the SAD and therefore both artefacts are created iteratively in the chosen approach (see Section 3.1.3). Main input artefacts (represented in Figure 3.6) for defining the FSC are requirements and assumptions from the item definition (see Section 3.2), the safety goals as a result of the HARA (see Section 3.3), the SAD and the fault hypothesis.

The defined safety goals in Table 3.3 already consider high level safety measures. These safety measures are refined in the FSC, assigned to specific components on vehicle and/or system level and traced to specific safety goals and covered HW/SW faults.

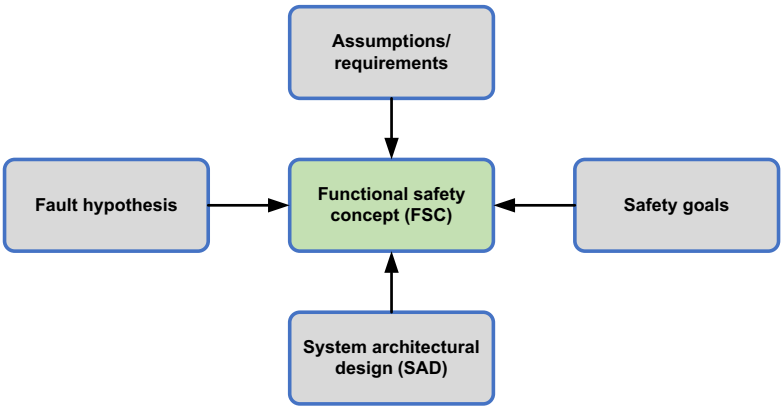

Figure 3.6: FSC input artefacts

3.2 Item definition

The design process of the chosen combined approach is initiated with the item definition, performed according to [8]. A hypothetical system is defined initially as a basis for all further investigations. It is as generic as possible, but as detailed as necessary to provide a profound basis for applying and discussing different architectural patterns. This system description is used to identify the item, extract its functionality and define interfaces to other items or the environment.

The described system targets applications towards autonomous driving compliant to SAE levels 3+. A central high performance computing platform, referred to as ADS-CP, receives sensor data (RADAR, LIDAR, camera) and performs calculations (sensor fusion, object detection, trajectory planning) on the received data to find a safe trajectory for the vehicle. The calculated trajectory is sent to the Vehicle Control Unit (VCU) in form of steering, acceleration and braking data. All components of the system (sensors and ECUs) are supplied by a power supply component. The supposed system is shown in Figure 3.7.

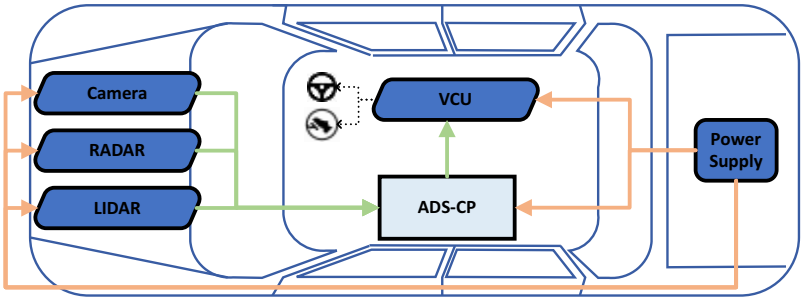

Figure 3.7: System overview

Based on this high level system description, the item, where the FO concepts are evaluated on, is identified as the ADS-CP. Several assumptions and requirements on system and item level are established in Table 3.1 and Table 3.2. Requirements related to environmental conditions or legal, quality and performance requirements [8] are outside the scope of this thesis.

#	Description
AS1	Parameters of the power supply are not considered in the ongoing design process. If required by the FO architectural pattern, redundant and diverse power supplies can be assumed.
AS2	Parameters of the sensors (RADAR, LIDAR, camera) are not considered in the ongoing design process. If required by the FO architectural pattern, redundant and diverse communication paths can be assumed.
AS3	Parameters of the VCU are not considered in the ongoing design process. If required by the FO architectural pattern, redundant and diverse communication paths can be assumed.

Table 3.1: System assumptions

The item's boundary within the car is defined according to Figure 3.8. Considered interfaces are:

- connection to power supply
- communication interface to sensor set (RADAR, LIDAR, camera)
- communication interface to VCU

The main purpose of the item is to enable the execution of ADFs with high reliability and an availability according to the targeted SAE level. To accomplish that, a certain

#	Description
REQ1	The system shall host SWCs, capable of executing safety critical functions with high reliability and availability levels, requested by the targeted SAE driving automation level. The necessary integrity level should be determined through a HARA.
REQ2	The SWCs shall implement algorithms capable of processing sensor set data (RADAR, LIDAR, camera) used for executing ADFs. Particularly, sensor fusion, object detection, trajectory planning and calculation of steering, acceleration and braking data shall be performed.
REQ3	The system shall remain operational in case a single failure occurs. The covered and uncovered faults shall be specified in the definition of a fault hypothesis.
REQ4	The system shall receive sensor data via suitable communication interfaces. Therefore, all prevalent network technologies used in the automotive domain may be used.
REQ5	The system shall send steering, acceleration and braking data to the VCU via suitable communication interfaces. Therefore, all prevalent network technologies used in the automotive domain may be used.
REQ6	The FO concept may also allow degraded functionality when executing fallback functions, as long as the specifications of the targeted SAE driving automation levels are not violated.
REQ7	The system is assumed to execute ADF of SAE driving automation levels 3+. The maximum achievable driving automation level should be determined individually for different architectural patterns.
REQ8	No restriction regarding power consumption is defined.
REQ9	No requirement regarding bandwidth of the used communication channels is defined.
REQ10	No fail-over time limit is specified. The resulting fail-over time must fit for the targeted SAE driving automation level.
REQ11	The ASIL level of a potential fallback system may be lower than the ASIL level of the primary system. If so, the resulting restrictions must be defined.

Table 3.2: System requirements

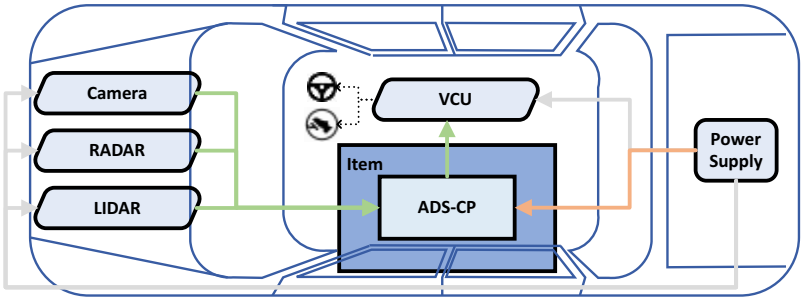

Figure 3.8: Item definition and boundaries to the rest of the system

degree of FO capability is required, which will be investigated in more detail in Section 3.4, Section 3.5 and Section 3.6. A high level design of the item decomposes the ADS-CP into a controller component, which is responsible for executing specific ADF algorithms (sensor fusion, object detection, trajectory planning, steering/drive train data calculation), a power management and an external communication component. This item decomposition into components is shown in Figure 3.9 and does not yet consider any necessary redundancy or diversity.

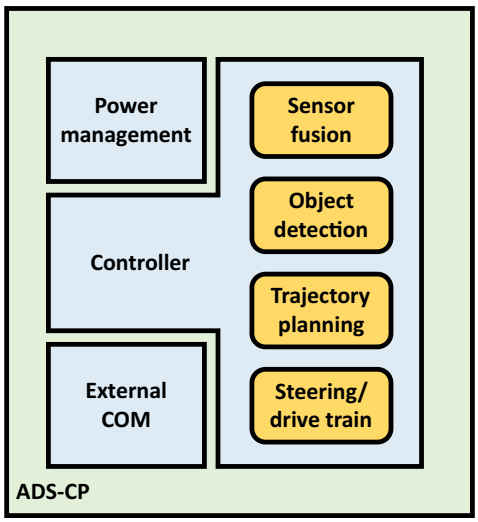

Figure 3.9: High level component decomposition of the ADS-CP

3.3 HARA

The suggested approach by ISO 26262 to identify safety goals and therefore top level safety requirements is to perform a HARA. It is a method to determining hazardous events resulting from any malfunctioning behaviour of the item, assigning ASILs and deriving safety goals [8]. A HARA is commonly used in the concept phase for any safety critical system, not just in the automotive industry, hence, several variants exist regarding relevant information and assumptions to be considered for performing a HARA.

The proposed HARA variant in the ISO 26262 standard is based on the item definition. Suggested supporting methods to systematically identify hazards are Failure Mode and Effects Analysis (FMEA) and Hazard and Operability Analysis (HAZOP). The identified hazards are used to determine relevant hazardous events, including their consequences and associated operational situations and modes. The hazardous events are further classified by assigning levels of severity, probability of exposure and controllability, which might then result in a specific ASIL. For hazardous events rated with an ASIL, functional objectives in form of safety goals are defined, which are input for the FSC [8].

With respect to AD, some definitions of terms in the ISO 26262 standard are vague. An "operational situation" according to ISO 26262 is a "scenario that can occur during a vehicle's life" [8]. Since the terms "situation" and "scenario" are widely spread in the AD domain, it must be distinguished between ISO 26262 definition and generic meaning in the AD context [13]. In [13], the term "operational scenario" is therefore used instead of "operational situation".

Another shortage of the suggested HARA procedure in ISO 26262 is the absence of considering the driver's role. In the description of the operational situations and modes it should be defined whether the driver is in the loop or not [51]. Another mentionable remark related to the HARA is the fact that for some ADFs the driver would not be able to control the hazardous event in any way, because he is not designated to be in the control loop for highly or fully automated functions. Hence, the controllability of these hazardous events would always be rated as *C3* (difficult to control or uncontrollable) [51]. A very low controllability of hazardous events for highly automated functions is also mentioned in [13].

The chosen approach to perform a HARA for the defined ADS-CP is a simplified combination of suggested approaches in [8], [13] and [51]. Since the ASILs for the high level decomposed components in Section 3.2 need to be determined in order to apply the chosen architectural patterns, a simplified, minimal HARA approach is sufficient here. The following attributes are considered for each component:

- malfunction of the component
- operational (simplified) scenario
- impact of the malfunction

- probability of exposure, severity and controllability classification according to [8]
- resulting ASIL for the hazardous event according to [8]
- derivation of a associated safety goal

It is assumed that the driver is out of the control loop for all identified hazardous events, hence, the controllability levels are specified as *C3* (difficult to control or uncontrollable). Since the focus of this thesis is less on the HARA, but on the comparison of FO architectures, no assisting systematic method for hazard determination has been applied (e.g. FMEA or HAZOP). The HARA was performed using brainstorming to identify malfunctions of the components. The minimal HARA of the ADS-CP is listed in Appendix A. A more comprehensive HARA for an automated unmanned vehicle was performed in [13].

#	Safety Goal	ASIL level	Safety measure
SG1	A single invalid, corrupt or lost frame of 1 sensor type must not lead to any unsafe situation	ASIL D	compensation measure in trajectory planning (interpolation); End-to-End (E2E) protection of sensor data
SG2	Permanent invalid, corrupt or lost frames of 1 sensor type must not lead to any unsafe situation	ASIL D	compensation measure in sensor fusion (interpolation); E2E protection of sensor data
SG3	Invalid or corrupt frames must be detected	ASIL D	E2E protection of sensor data
SG4	Permanent loss of frames of at least 2 sensor types must trigger the emergency operation, i.e. an emergency brake using the last calculated steering angle	ASIL D	sensor data reception via diverse channels; implementation of emergency operation on SWC level
SG5	The power supply of the ADF-CP must not be interrupted	ASIL D	redundant/diverse power supply
SG6	The power supply must conform to the specified boundaries	ASIL D	voltage monitoring by power management; switch over capability to redundant/diverse power supply
SG7	Correct steering data must be sent continuously	ASIL D	SWC (and hosting HW) development according to ASIL D; steering data transmission via diverse channels; E2E protection of steering data
SG8	Correct drive train data must be sent continuously	ASIL D	SWC (and hosting HW) development according to ASIL D; drive train data transmission via diverse channels; E2E protection of drive train data
SG9	Sensor fusion algorithm mustn't be in a incorrect, not working state for more than 3 frames per second	ASIL D	SWC (and hosting HW) development according to ASIL D
SG10	Sensor fusion data mustn't contain incorrect, corrupt or missing data for more than 3 frames per second	ASIL D	SWC (and hosting HW) development according to ASIL D
SG11	Object recognition algorithm mustn't be in a incorrect, not working state for more than 3 frames per second	ASIL D	SWC (and hosting HW) development according to ASIL D
SG12	Object recognition algorithm mustn't produce more than 6 frames per second containing false positives	ASIL B	SWC (and hosting HW) development according to ASIL B
SG13	Object recognition algorithm mustn't produce more than 3 frames per second containing false negatives	ASIL D	SWC (and hosting HW) development according to ASIL D
SG14	Object recognition data mustn't contain poor quality data (no/invalid/wrong data) for more than 3 frames per second	ASIL C	SWC (and hosting HW) development according to ASIL C
SG15	Trajectory planning algorithm mustn't be in an incorrect, not working state for more than 1 frame per second	ASIL D	SWC (and hosting HW) development according to ASIL D
SG16	Trajectory planning algorithm must provide a trajectory with minimum threat to human lives (pedestrians, driver, other drivers) in case no safe trajectory could be found	ASIL D	SWC (and hosting HW) development according to ASIL D
SG17	Trajectory planning algorithm mustn't provide no or an unsafe trajectory for more than 1 frame per second	ASIL D	SWC (and hosting HW) development according to ASIL D

Table 3.3: Safety Goals

The determined safety goals are listed in Table 3.3. For each safety goal, high level safety measures independent of the architectural pattern are derived. More detailed safety measures are treated individually in the FSCs and SADs for all selected FO architectures (see Section 3.4, Section 3.5 and Section 3.6). As shown in Table 3.3 these high-level safety measures are mostly a requirement for a certain ASIL. For each selected FO architecture it is elaborated how the respective ASIL is established.

3.4 Simplex architecture

The basic concept of the Simplex architecture, based on the state of the art research performed in Section 2.1.2, consists of high-performance processing components and reliable processing components. The high-performance component is referred to as *mission controller*, the reliable processing part is denoted as *base controller*. A *decision module* verifies the result of the *mission controller* and forwards it to the communication channel if no error was detected, otherwise the *base controller*'s result is sent (see Figure 3.10).

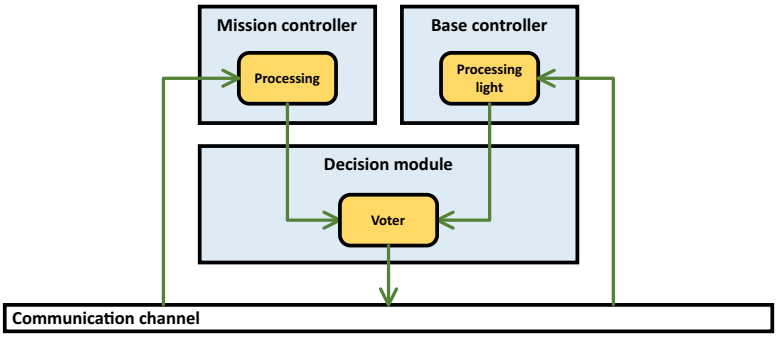

Figure 3.10: Simplex architectural pattern

The Simplex architecture may not be a preferred choice as FO architecture for an ADS as the complexity of even the base controller may not allow to realize a sufficiently high reliability. However, the Simplex architecture was selected, because it is a basic architecture that allows to introduce more advanced architectures more easily and it serves well for comparison with the other selected architectures.

3.4.1 Fault hypothesis

Following the selected methodology to be applied for classifying the FO architectures (see Section 3.1.3), the fault hypothesis is split up into HW and SW sections.

Hardware faults

The HW related FCRs of the ADS-CP following the Simplex architectural approach (see Figure 3.10) are defined according to the high level component decomposition performed in the item definition phase (see Section 3.2).

The FO capability of the ADS-CP is achieved by defining independent FCRs, forming a Fault-Tolerant Unit (FTU). A single fault must be confined within the affected FCR and must not lead to any failure of the overall system. Since according to the architectural pattern a fault in the *base controller* or the *decision module* would cause the whole system to fail, the FCR consisting of safety controller (acting as base controller and decision module) and *Supply 2* is considered to be verified completely and therefore considered as free of any types of faults [21].[2]

A fault in power supply component *Supply 1*, *power management*, *high-performance controller* or *external Communication (COM)* would cause a failure in the *mission controller*, so all aforementioned components form a separate FCR (see Figure 3.11). If the *mission controller* fails, the *decision module* inside the *safety controller* part selects the *base controller*'s output, hence the fault in the *mission controller* is masked out.

In order to tolerate a fault in the ADS-CP's (external) communication channel, a redundant channel is added. Both communication channels fail independently, assuming that they are sufficiently diverse, and therefore can be considered as separate FCRs (see Figure 3.11).

For specifying the fault hypothesis of the Simplex approach, the following assumptions are considered:

- The FCR containing the *safety controller* and power supply *Supply 2* is assumed to be free of any type of faults.
- It is assumed that single faults occur. More than one fault at the same time are considered to be rare events (uncovered faults). This is commonly accepted and used e.g. in [4] and [53].
- The temporal classification of the covered faults (see Table 3.4) are considered to be either transient or permanent.
- All other faults (not covered in Table 3.4) are assumed to be rare events. A NGU strategy is defined to react on these uncovered faults (see Section 3.7).
- The FCR containing a communication channel is assumed to not create correct frames spontaneously. The communication channels won't add any arbitrary delay when sending/receiving data [54]. Furthermore it is assumed that the communication channels are designed in a way such that a faulty mission controller is contained.

[2] Since in practice a full verification is not feasible, faults in the safety controller part can be masked out by e.g. using a TMR approach [21].

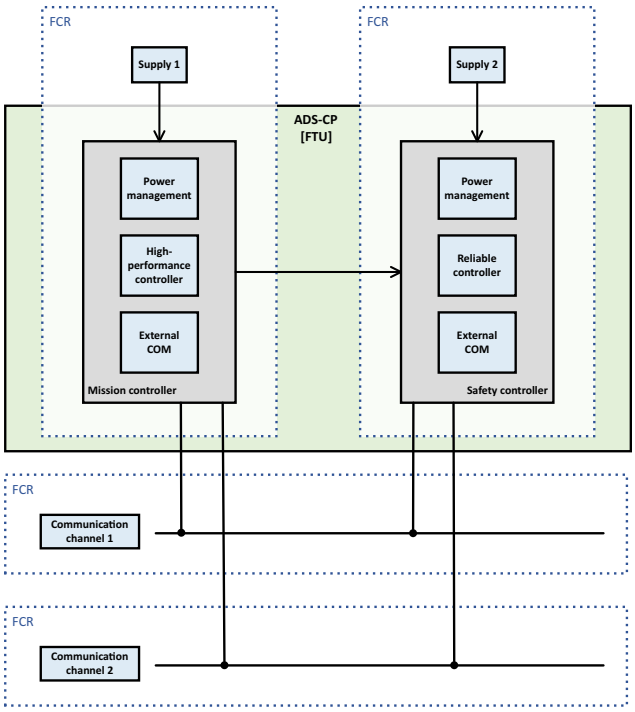

Figure 3.11: FCR decomposition of the Simplex approach (hardware components)

Based on these assumptions, the covered hardware faults are defined according to Table 3.4.

Software faults

The specification of the fault hypothesis related to software faults is based on the ADS-CP's high level component decomposition (see Figure 3.9) and the Simplex architectural pattern (see Figure 3.10). It is assumed, that the application layer relies on an underlying platform SW and an Operating System (OS). For the software related FCR definition, the following layers are considered:

- System SW (OS and platform SW)
- Application software (Sensor fusion, Object detection, Trajectory planning, Steering/drive train, Decision module SWCs)

It is assumed, that the (same) platform SW is deployed on the mission controller and on the safety controller. Defining the platform SW as separate FCRs on mission controller and safety controller would require sufficient degree of independence between both platform SW instances, hence a single FCR is defined (see Figure 3.12). Using this approach requires the assumption for the platform SW to be free of any type of faults to avoid a single point

#	Component	Fault	Impact
SCHWF1	Supply 1	Voltage level of power supply below the specified range	Mission controller not supplied anymore, no output can be produced
SCHWF2	Supply 1	Voltage level of power supply above the specified range	Mission controller must shut down to prevent potential damage, no output produced anymore
SCHWF3	Power management (Mission controller)	Mission controller's power management fails to provide the required voltage levels for an arbitrary component	Mission controller should perform a transition to a safe state; Incorrect results would be masked out by the decision module
SCHWF4	Controller (Mission controller)	Controller provides no result on all interfaces (fail-silent behaviour) on the mission controller	Mission controller should perform a transition to a safe state; Incorrect results would be masked out by the decision module
SCHWF5	External COM (mission controller)	External communication component on the mission controller fails to forward received data from the bus to the controller	Mission controller should perform a transition to a safe state; Incorrect results would be masked out by the decision module
SCHWF6	Bus 1, Bus 2	Communication channel not ready to transmit data due to fault on the physical level	Transition to a degraded mode must be performed by the ADS-CP

Table 3.4: Simplex fault hypothesis: covered hardware faults

of failure [53]. Since for the high performance controller and for the safety controller anyway different and independent OSs are used (see Section 3.4.2), separate FCRs are defined.

The application level components on the safety controller (Sensor fusion, Object detection, Trajectory planning, Steering/drive train, Decision module SWCs) are assumed to be free of faults (as all components on the safety controller) to avoid single point of failures. So on the safety controller's SW level all SWCs form a single FCR where this assumption is applied on. The SWCs on the mission controller form separate FCRs, since a fault in any SWC is isolated and not affecting other components.

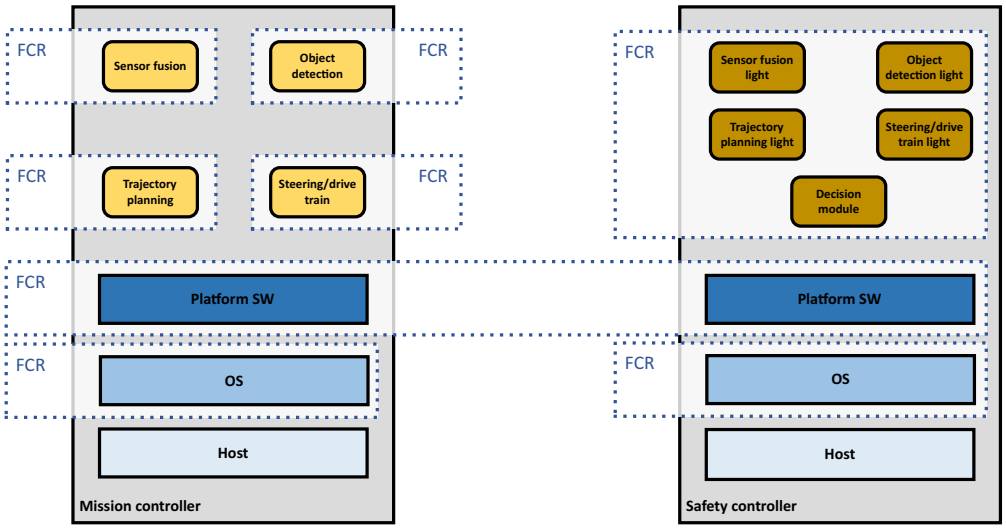

Figure 3.12: FCR decomposition of the Simplex approach (software components)

Fundamental assumptions regarding software faults of the Simplex approach are:

- The occurrence of a single fault is assumed. More than one fault at the same time are considered to be a rare event (uncovered fault). This is commonly accepted and used e.g. in [4] and [53].
- The temporal classification of the covered software faults (see Table 3.7) are permanent faults by definition.
- All other faults (not covered in Table 3.7) are assumed to be rare events. A NGU strategy is defined to react on these uncovered faults (see Section 3.7).
- The FCR containing the platform SW is assumed to be fault-free (as assumed e.g. in [53]).
- The system SW handles the communication between SWCs and between mission controller and safety controller. Since the absence of faults in the platform SW and in the safety controller's OS is assumed, failure modes like arbitrary timing/value message failures in the communication system don't have to be considered.

The covered SW faults of the fault hypothesis are defined according to Table 3.5.

3.4.2 SAD

The SAD is defined according to the selected methodology in Section 3.1.3. Similar to the definition of the fault hypothesis, a separation into HW and SW related investigations is done.

#	Component	Fault	Impact
SCSWF1	All SWCs (application level) on the mission controller	SWC produces no output data	Mission controller should perform a transition to a safe state; Incorrect results would be masked out by the decision module
SCSWF2	All SWCs (application level) on the mission controller	SWC produces incorrect output data (detectable by consumer)	Mission controller should perform a transition to a safe state; Incorrect results would be masked out by the decision module
SCSWF3	All SWCs (application level) on the mission controller	SWC fails to fetch input data	Mission controller should perform a transition to a safe state; Incorrect results would be masked out by the decision module

Table 3.5: Simplex fault hypothesis: covered software faults

HW architecture

The mission controller and safety controller (composed of base controller and decision module) are integrated on a single ECU (see Figure 3.13). This is tolerated here because no common mode faults related to spatial proximity are considered in the fault hypothesis. The ongoing investigation on the HW architecture is separated into mission controller (i) and safety controller (ii).

i) *Mission controller:* The core components of the mission controller are two diverse PHs. Since it is assumed that the maximum ASIL of a PH is ASIL B, the requirement to host SWCs of ASIL D depends on an ASIL decomposition from one ASIL D component to two ASIL B components, denoted as ASIL B(D). This requires a sufficient degree of independence among the components used for decomposition [8].

   The communication between the PHs is realized via an Ethernet backbone, consisting of Ethernet switch A (ASIL D). The switch is also connected to the safety controller's Ethernet backbone, hence a cascaded switch configuration is used. The mission controller receives sensor data via two diverse communication interfaces (CAN, Ethernet). The received Ethernet frames are forwarded from a transceiver component (Ethernet transceiver A) to both PHs via Ethernet switch A, while the received CAN messages are directly sent to the PHs from CAN transceiver A. The mission controller must not send any data to the external communication channels, all calculated results are sent to the safety controller instead.

   All components of the mission controller are supplied via a separate power management unit, realized e.g. by deploying a Power Management Integrated Circuit (PMIC). It is responsible for providing all required voltage levels to the

components and for the state management of the mission controller. A safe state of the mission controller is reached by disabling all power supplies, which is triggered by the safety controller in the defined SAD.

ii) *Safety controller:* The safety controller, hosting the base controller and the decision module in accordance with the Simplex architectural pattern, is assumed to be free of any type of faults.[3] It is composed of a SH, communication infrastructure (Ethernet switch, Ethernet transceiver, CAN transceiver) and a power management unit.

The power management unit again can be implemented using a PMIC, providing the requested voltage levels to all components of the safety controller part. Since the safety controller does not fail, no mode management is necessary here.

The external communication of the safety controller is realized via appropriate transceivers for both used communication channels (Ethernet, CAN) and an Ethernet backbone (implemented via Ethernet switch B). The SH receives sensor data via both communication channels and verifies the mission controller's calculated results. In case the verification was successful, the mission controller's data is forwarded to the external communication channels. Otherwise the lightweight calculations performed by the SH are sent. Besides sensor data processing and verification tasks, the SH is also responsible for the mission controller's mode management by controlling the Power management A component.

SW architecture

The SW architecture for the Simplex approach describes the SWC design (i) and the dataflow between all SWCs, the system SW and the vehicle (ii).

i) *SWC design:* The SW architecture is based on a layered approach, where the lowest layer on top of the silicon are the OSs (see Figure 3.14). On the PHs a Portable Operating System Interface (POSIX) compliant OS is deployed (e.g. QNX [55]), while on the safety host an OS based on the AUTOSAR [45] standard is used (e.g. MICROSAR [56]). On top of the OS layer a platform SW, implementing a time triggered architecture, handles communication (between SWCs and vehicle) and task scheduling.

The SW related functional blocks in the high level component decomposition (see Section 3.2) are implemented as separate SWCs on top of the platform SW (sensor fusion, object detection, trajectory planning, steering/drive train). To satisfy the reliability requirement of ASIL D, a ASIL decomposition for the SWCs hosted on the (ASIL B) PH is performed according to [8], hence a sufficient degree of independency among the SWCs must be ensured. On the SH, lightweight versions of the sensor fusion, object detection, trajectory planning and steering/drive train SWCs in terms of reduced feature set are deployed. Additionally, a decision module SWC is hosted,

[3] In practice the reliability of a system utilizing a Simplex architecture is highly determined by the reliability of the base controller and the decision module (and their respective FCRs).

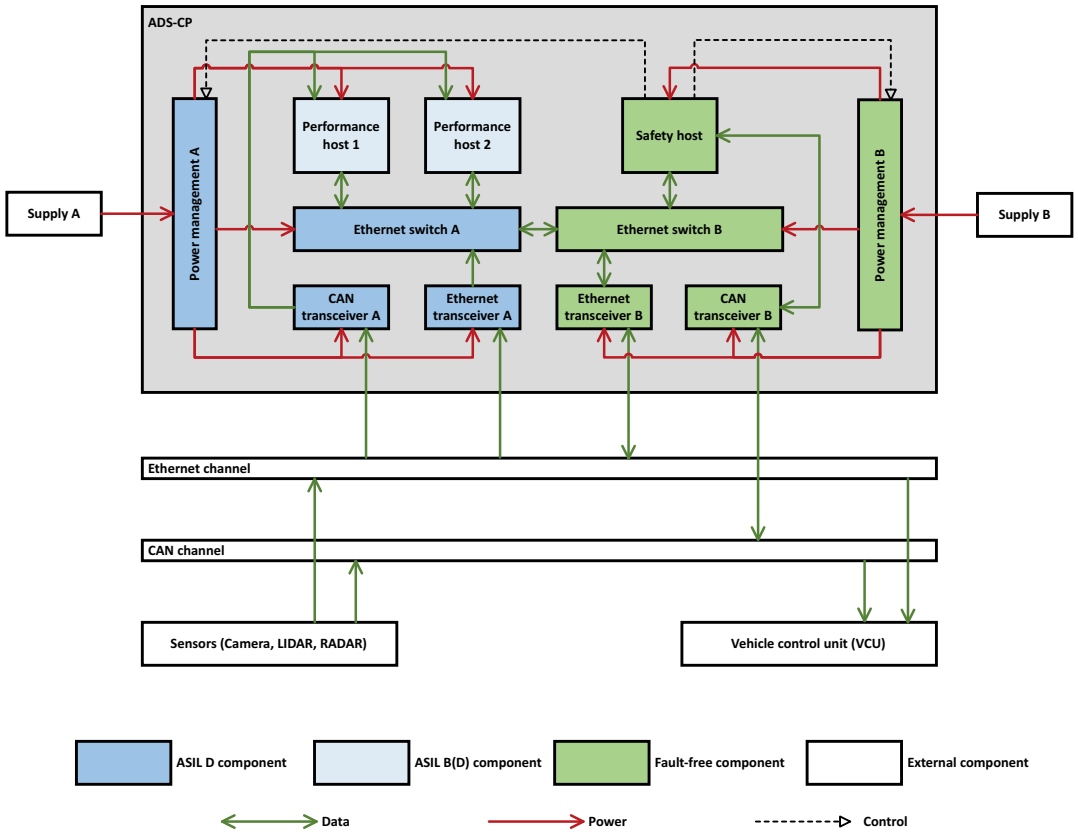

Figure 3.13: Hardware architecture of the Simplex approach

responsible for verifying the PHs' calculations and selecting the data to be sent to the VCU accordingly.

ii) *Dataflow definition:* Sensor data (camera, LIDAR, RADAR) is sent via redundant and diverse communication channels (Ethernet, CAN) to the sensor fusion SWCs on all hosts. The results of the SWCs hosted on the PHs are verified by the decision module SWC on the SH.[4] If the verification was successful, the mission controller's data is sent to the VCU via the redundant and diverse external communication channels, otherwise the base controller's results are sent (see Figure 3.15).

3.4.3 FSC

As explained in Section 3.1.3, the high level safety measures (see Table 3.3) are refined in the FSC and traced to the safety goals and covered faults defined in the fault hypothesis.

[4]Possible concepts to deal with replica indeterminism are explained in [42]. In this investigations on the Simplex approach, these problems are neglected.

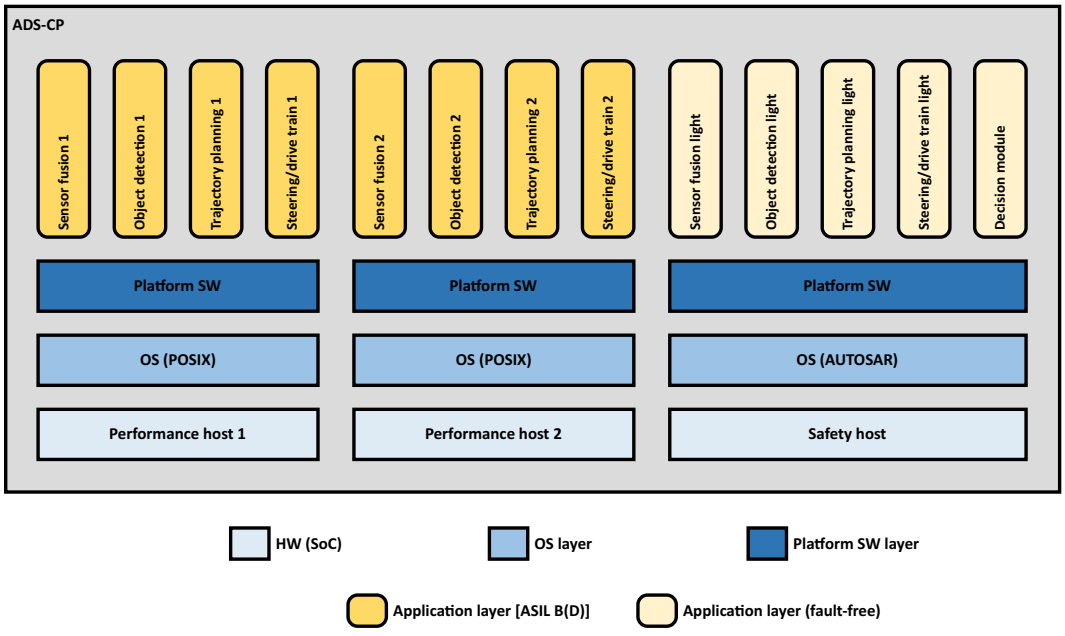

Figure 3.14: Software architecture of the Simplex approach (SWCs per host)

SWC development according to ISO26262

As determined in the HARA, the SWCs must be developed according to ASIL D for the mission controller. The SWCs hosted on the PHs are decomposed to independent ASIL B(D) components (see Section 3.4.2).

Since the safety controller is assumed to be free of faults, no ASIL requirements are assigned to the decision module, sensor fusion light, object detection light, trajectory planning light and steering/drive train light SWCs.

Covered safety goals: **SG7-SG17**

HW development according to ISO26262

In order to be able to host ASIL D SWCs, the HW must be developed according to ASIL D for the mission controller (Power management A, CAN transceiver A, Ethernet transceiver A, Ethernet switch A).

The safety controller is assumed to be free of faults, so no ASIL requirements are assigned to its HW components.

Covered safety goals: **SG7-SG17**

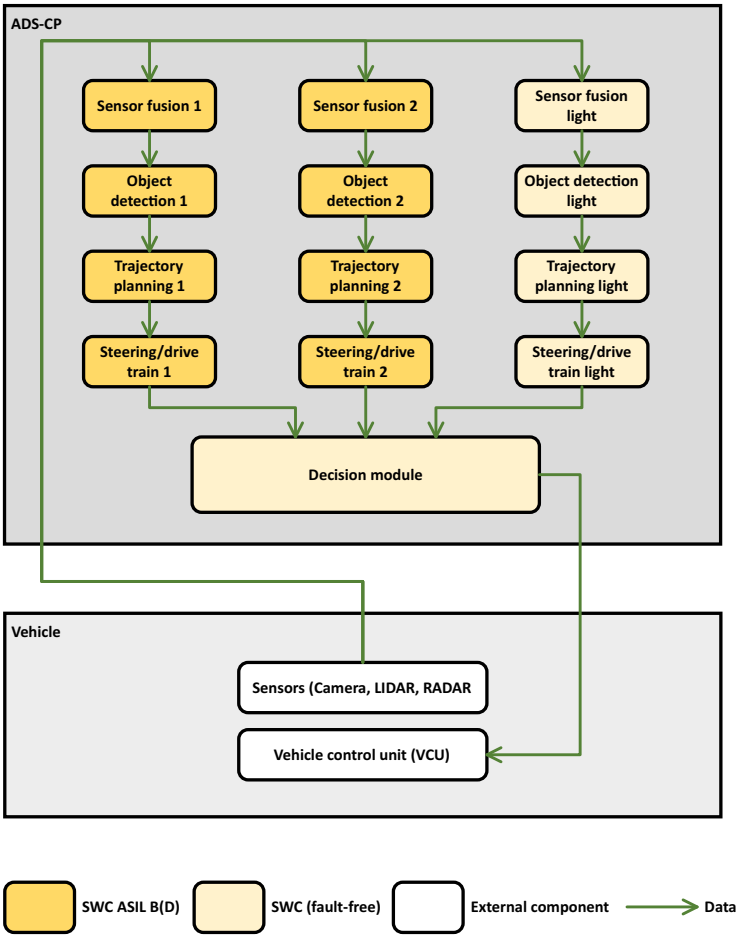

Figure 3.15: Software architecture of the Simplex approach (dataflow)

Functional safety measures implemented in SWCs

To compensate loss of sensor data appropriate measures (e.g. interpolation) must be implemented in the ADF SWCs. If a permanent loss of sensor data (at least 2 types) occurs, an emergency operation must be implemented in the trajectory planning SWC (controlled breaking manoeuvre until standstill).

Covered safety goals: **SG1**, **SG2**, **SG4**

E2E protection of data sent over communication channels

All data sent over any communication channel must be checked to ensure its integrity. Adding an E2E protection (Cyclic Redundancy Check (CRC), sequence counter) to all data being sent (CAN messages, Ethernet frames) ensures proper error detection.

Covered safety goals: **SG1-SG3**, **SG7-SG8**

Redundancy and diversity

On vehicle level the ADS-CP's power supplies, the sensor sets and the communication channels connected to the ADS-CP must be implemented in a redundant and diverse way (CAN, Ethernet). Each sensor set uses a dedicated communication channel for data transmission and mission controller and safety controller are supplied by dedicated power supplies to minimize the risk of common cause faults.

The ADS-CP must provide independent interfaces to the redundant and diverse communication channels (CAN, Ethernet) on vehicle level. The base controller is added as redundant unit to the mission controller (Simplex architecture) and ensures (degraded) operation in case a fault occurs on the mission controller. The mission controller's result is verified by the decision module and forwarded to the VCU in case the verification was successful. The safety controller (hosting base controller and decision module) is also responsible for monitoring the voltage levels and disabling the mission controller if the specified voltage levels are violated.

The mission controller's interfaces to the external communication channels (CAN, Ethernet) only must be used for receiving data, but not for sending. Sending data to the vehicle is only handled by the decision module, hosted on the safety controller.

Covered safety goals: **SG4-SG8**

Trace to fault hypothesis: **SCHWF1-SCHWF5**, **SCSWF1-SCSWF3**

Degradation strategy

Once a fault occurred on the mission controller, the driver must be alerted that the vehicle is operating in a degraded mode in terms of features and operation time [8]. If the error of the mission controller disappeared after a restart (transient fault), the ADS-CP is able to operate in normal mode again and an appropriate notification to the driver must be initiated again.

Trace to fault hypothesis: **SCHWF6**

3.5 1oo2D architecture

Some fail-operational approaches based on the 1oo2D architecture were already outlined in Section 2.1.3. The fundamental architectural pattern all mentioned 1oo2D approaches are based on is shown in Figure 3.16.

The concept of the 1oo2D approach is based on the combination of two fail-silent units, referred to as *primary mission controller* and *secondary mission controller*. Providing

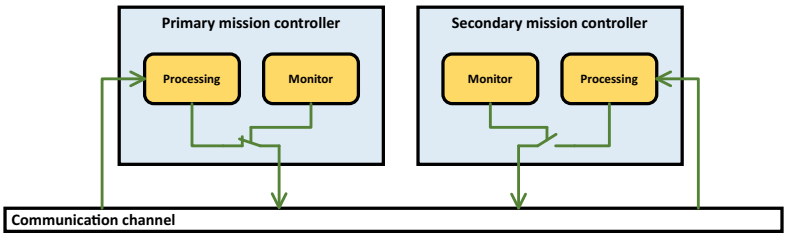

Figure 3.16: 1oo2D architectural pattern

fail-silent capability requires sufficient diagnostic capabilities on both units to monitor its behaviour and ensure that no data is sent to the communication bus anymore in case a fault occurred. The monitoring components on both units exchange diagnostic data in order to provide a consistent view on the current state of both units. Valid states for this chosen 1oo2D approach are either one unit sends valid data or no unit sends data at all. The mode switch of the units is handled by a defined fail-over mechanism, which covers the definition of the exchanged diagnostic data and the agreement between the units to ensure only one active unit at any point of time.

3.5.1 Fault hypothesis

As explained in the selected methodology (see Section 3.1.3), the fault hypothesis is split up into HW and SW related faults.

Hardware faults

The definition of FCRs with respect to hardware faults is based on the high level component decomposition of the ADS-CP (see Figure 3.9) and the 1oo2D architectural patten (see Figure 3.16).

Since a mission controller in the presented approach contains shared physical resources (power management, controller, power supply), primary and secondary mission controllers including their power supplies are considered as separate FCRs. Both mission controllers form a FTU, providing FT capabilities. Any fault affecting a shared physical resource is very likely to result in a failure of the complete mission controller, so in accordance with the criteria for a FCR [4], the definition is done as shown in Figure 3.17.

The assumption that the hardware components of the mission controller FCRs fail independently requires a sufficient degree of diversity regarding design and component selection in order to prevent common mode failures, similar to the approach in [53]. The diversity requirement must be considered in the SAD (see Section 3.5.2).

Each communication channel (bus 1, bus 2) is defined as separate FCR. To fulfil the definition criteria, both FCRs again must be designed considering sufficient diversity.

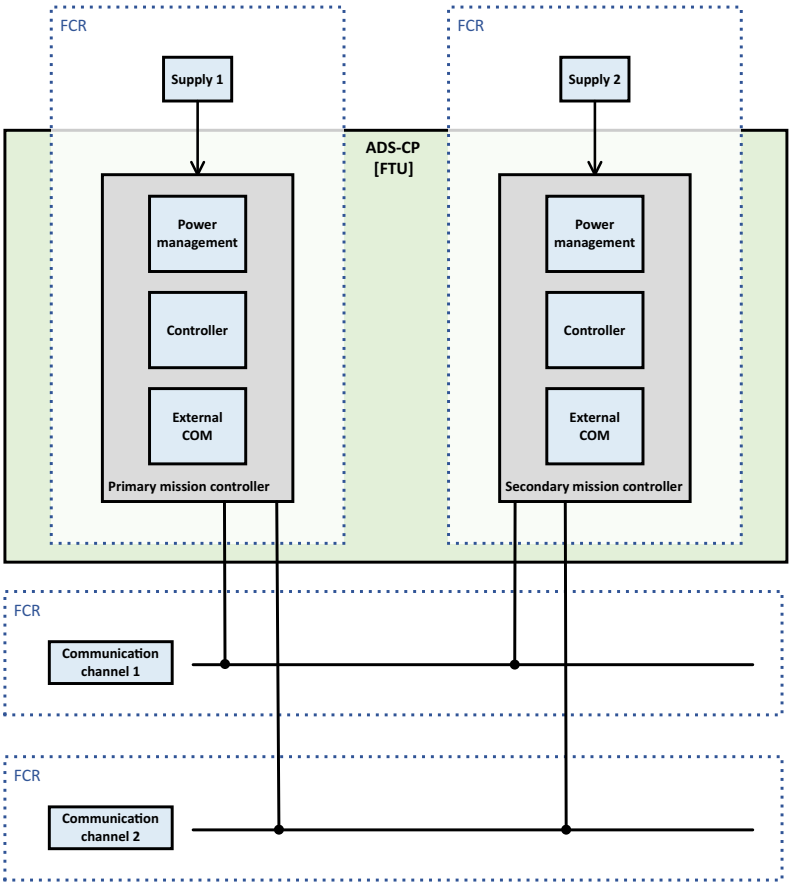

Figure 3.17: FCR decomposition of the 1oo2D approach (hardware components)

The following assumptions regarding failure modes are defined for defining the fault hypothesis:

- The occurrence of a single fault is assumed. More than one fault at the same time are considered to be a rare event (uncovered fault). This is commonly accepted and used e.g. in [4] and [53].
- The temporal classification of the covered faults (see Table 3.6) are considered to be either transient or permanent.
- All other faults (not covered in Table 3.6) are assumed to be rare events. A NGU strategy is defined to react on these uncovered faults (see Section 3.7).
- The FCR containing a communication channel is assumed to not create correct frames spontaneously. Further the communication channels won't add any arbitrary delay when sending/receiving data [54].

The covered hardware faults within the fault hypothesis, considering the defined assumptions, are listed in Table 3.6.

#	Component	Fault	Impact
DCHWF1	Supply 1, Supply 2	Voltage level of power supply below the specified range	Primary or secondary mission controller not supplied anymore, no output can be produced
DCHWF2	Supply 1, Supply 2	Voltage level of power supply above the specified range	Primary or secondary mission controller must shut down to prevent potential damage, no output produced anymore
DCHWF3	Power management	Power management fails to provide the required voltage levels for an arbitrary component apart from the safety MCU	Affected mission controller must perform a transition to a safe state to prevent possible incorrect results
DCHWF4	Controller	Controller provides no result on all interfaces (fail-silent behaviour)	Affected mission controller must perform a transition to a safe state to prevent possible incorrect results
DCHWF5	External COM	External communication component fails to forward received data from the bus to the controller	Affected mission controller must perform a transition to a safe state to prevent possible incorrect calculations
DCHWF6	External COM	External communication component fails to forward received data from the controller to the bus	Affected mission controller must perform a transition to a safe state to prevent possible incorrect calculations
DCHWF7	Bus 1, Bus 2	Communication channel not ready to transmit data due to fault on the physical level	Transition to a degraded mode must be performed by the ADS-CP

Table 3.6: 1oo2D fault hypothesis: covered hardware faults

Software faults

In addition to the high level component decomposition of the ADS-CP (see Figure 3.9) and the 1oo2D architectural pattern (see Figure 3.16) as input for the software related FCR decomposition, a software platform is considered to be used on all hosts between the underlaying OSs and the application layer (see Figure 3.18).

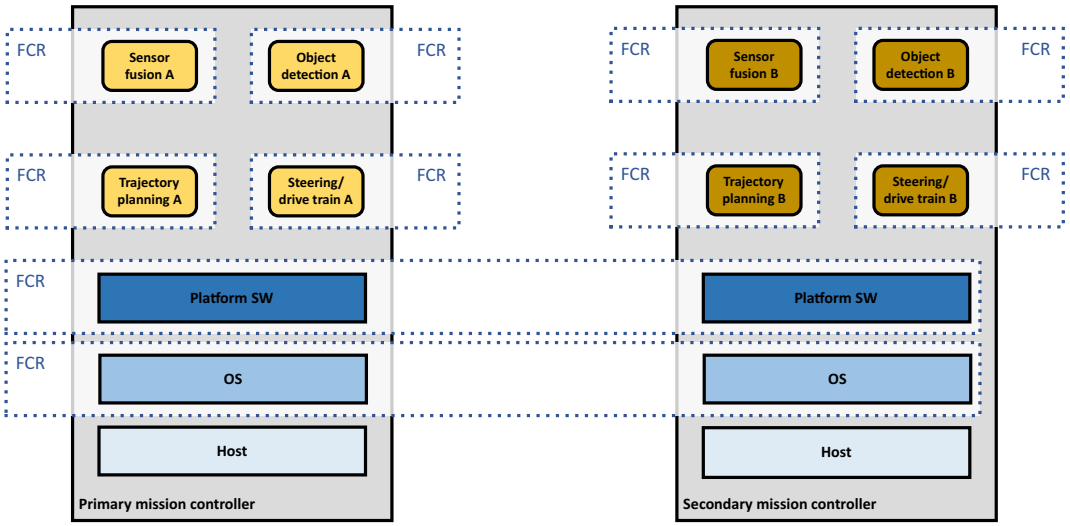

Figure 3.18: FCR decomposition of the 1oo2D approach (software components)

For the software FCR decomposition two different types are distinguished:

- System SW (OS and platform SW)
- Application software (Sensor fusion, Object detection, Trajectory planning, Steering/drive train SWCs)

It is assumed, that primary and secondary mission controllers use the same OS and platform SW due to economic reasons[5]. Since defining the system SW as separate FCRs on both mission controllers would require sufficient degree of diversity regarding their implementations, OS and platform SW form single FCRs. As a consequence, the absence of faults in the system SW is assumed in this approach [53].

On application level it might be feasible to develop various implementations of specific SWCs, ensuring sufficient diversity by e.g. performing the development by independent teams and avoiding the usage of common SW modules to avoid common mode failures. However, according to [53], even utilizing different development teams might not be sufficient in terms of common mode failure avoidance. In the ongoing investigations in this thesis it is assumed that diverse, independent implementations of the SWCs are available and therefore it can be justified, that the main and redundant SWC implementations form separate FCRs.

[5]Developing complex systems like a (safety qualified) OS or platform SW requires extremely high efforts and therefore just a few candidates are available for specific needs.

The following assumptions with respect to software faults are defined:

- The occurrence of a single fault is assumed. More than one fault at the same time are considered to be a rare event (uncovered fault). This is commonly accepted and used e.g. in [4] and [53].
- The temporal classification of the covered software faults (see Table 3.7) are permanent faults by definition.
- All other faults (not covered in Table 3.7) are assumed to be rare events. A NGU strategy is defined to react on these uncovered faults (see Section 3.7).
- The FCRs containing the platform SW and the OS are assumed to be fault-free (as assumed e.g. in [53]).
- The system SW handles the communication between SWCs and between the mission controllers. Since the absence of faults in the system software is assumed, failure modes like arbitrary timing/value message failures in the communication system don't have to be considered.

Based on these assumptions, Table 3.7 lists the covered software faults of the defined fault hypothesis:

#	Component	Fault	Impact
DCSWF1	All SWCs (application level)	SWC produces no output data	Affected mission controller must perform a transition to a safe state to prevent possible incorrect calculations
DCSWF2	All SWCs (application level)	SWC produces incorrect output data (detectable by consumer)	Affected mission controller must perform a transition to a safe state to prevent possible incorrect calculations
DCSWF3	All SWCs (application level)	SWC fails to fetch input data	Affected mission controller must perform a transition to a safe state to prevent possible incorrect results

Table 3.7: 1oo2D fault hypothesis: covered software faults

3.5.2 SAD

All inputs defined in the used methodology for the SAD (see Section 3.1.3) are considered in the following investigations and a separation into HW and SW architecture is done.

HW architecture

Even though common mode failures caused by spatial proximity are not considered in the fault hypothesis (see Section 3.5.1), the ADS-CP in this approach is composed of two separate ECUs, acting as primary and secondary mission controller. If one ECU is used instead (containing both mission controllers), a justification regarding a sufficient degree of independence, and therefore the absence of common mode failures must be argued.[6] The architecture of the HW related high level decomposed parts (see Section 3.2) are described in the following paragraphs.

i) *Power management:* A power management unit (e.g. implemented by using one or more PMICs) is responsible for providing all required voltage levels to the controllers and communication components. In this approach it is also used to perform a mode switch of a mission controller to a safe state, e.g. by disabling the voltage supplies for all components (depending on the used peripheral components). The mode switch can be triggered either by the SH or by specific external events detected by the power management unit (e.g. a missing keep-alive signal from SH or a battery voltage outside the specified range).

   Both mission controllers are supplied by a dedicated battery (1:1 connection). Connecting the batteries to both mission controllers instead (1:2 connection) would introduce a risk for the occurrence of common mode failures (e.g. over voltage of 1 battery could affect both mission controllers).

ii) *Controller:* The controlling part of a mission controller is split up into a safety related (covered by SH) and computationally intensive (covered by PHs) scope. This is done, because state of the art PHs don't provide necessary safety features yet, and therefore the safety concept (see Section 3.5.3) envisages monitoring features on different levels (external watchdog, SH, PH). Since the maximum ASIL of state of the art PHs is up to ASIL B, the capability of hosting ASIL D SWCs requires an ASIL decomposition. According to [8], an ASIL D component can be decomposed to two ASIL B components, where the ASIL of the corresponding safety goal is denoted in parenthesis: ASIL B(D). The decomposition requires a sufficient degree of independency between the components [8].

   Considering both mission controllers, the decomposition results in a requirement to use 4 different PHs and 2 different SHs, again ensuring a sufficient degree of independency among them.

   The communication between all hosts on one ECU is realized via an Ethernet backbone. Therefore an Ethernet switch is used to connect both PHs, the SH and the Ethernet transceiver (used for vehicle communication). All traffic (including external Ethernet and CAN communication) is sent to the SH and - if required - forwarded to the PHs via the Ethernet backbone. The switch on the primary and secondary mission controllers again must be sufficiently independent to avoid common mode failures.

[6] It is assumed, that a complete fault hypothesis would have been done in this case, considering also failure modes related to spatial proximity.

iii) *External COM:* For enabling the ADS-CP to provide FT capability, two redundant and diverse communication channels must be used, since each of the channels form a FCR (see Section 3.5.1) and a fault occurring in any FCR must be tolerated [4]. In this approach CAN and Ethernet are used for communication between both mission controllers and between the ADS-CP and the vehicle[7].

Therefore diverse transceivers (for CAN and Ethernet) are used on both mission controllers. These transceivers must provide fail-safe capability to ensure that no data is sent once a fault is detected. The transition to the safe state is assumed to be done by disabling the voltage supply for the transceivers (handled by the power management).

[7]It is assumed that these technologies are sufficiently diverse to avoid common mode failures.

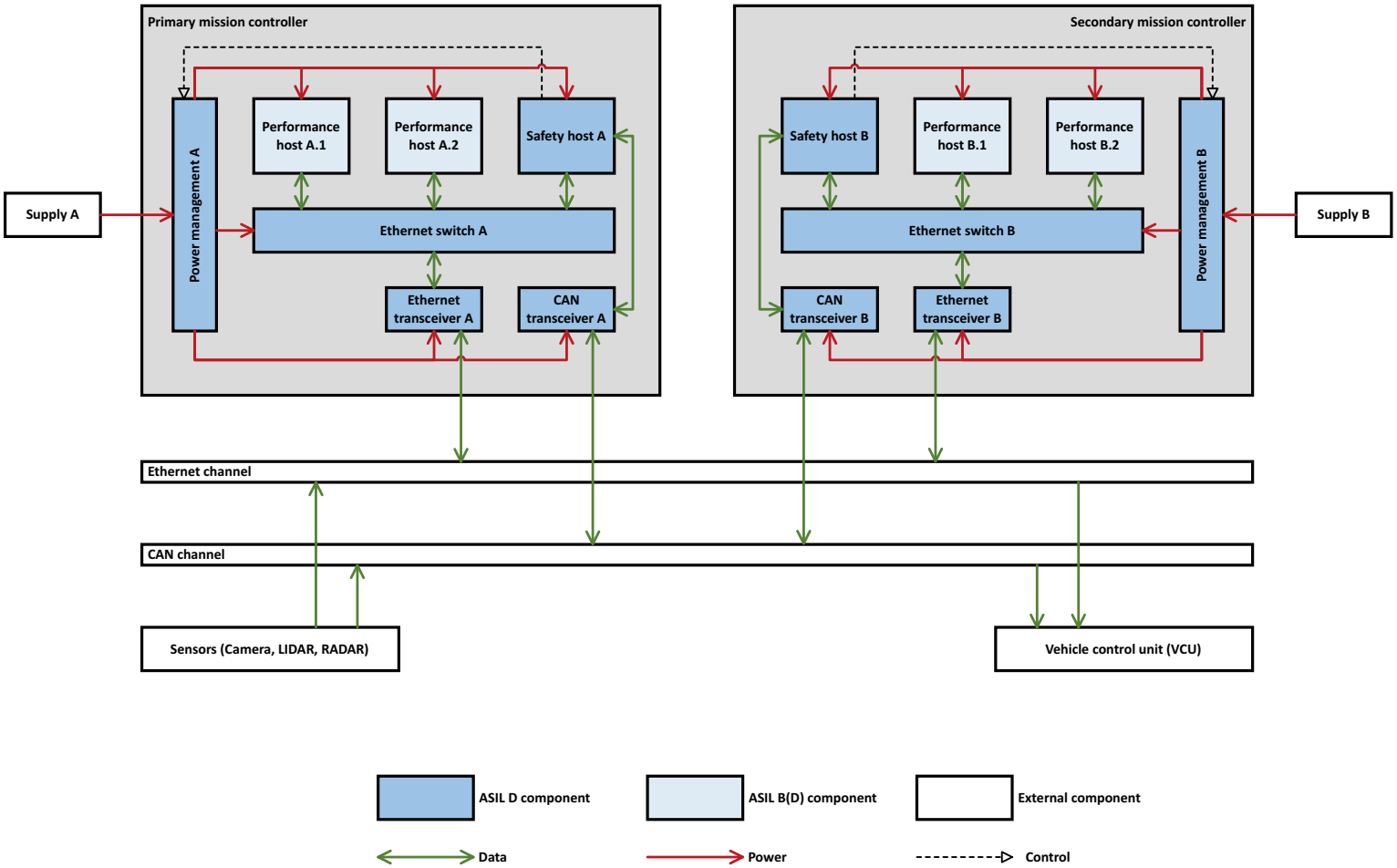

Figure 3.19: Hardware architecture of the 1oo2D approach

SW architecture

The SW architecture section covers the SWC design (i), dataflow between SWCs, system SW and vehicle (ii), and the chosen approach of handling activating a redundant/deactivating a faulty mission controller, referred to as fail-over mechanism (iii).

i) *SWC design:* A layered architecture is chosen in the SW design, where the lowest SW layer (on top of silicon) in this approach is the OS. On top of the OSs, a platform SW is added, offering specific services to the application layer (highest layer). The main services offered by this hypothetical platform SW are communication services (handling communication between SWCs and/or vehicle buses) and scheduling services (scheduling of SWC's tasks). Used OSs are on performance host side a POSIX compliant OS (e.g. QNX [55]) and on the safety host a AUTOSAR [45] OS (e.g. Microsar [56]).

   Each functional SW block according to the high level component decomposition (see Section 3.2) is implemented as separate SWC on top of the platform SW (sensor fusion, object recognition, trajectory planning, steering/drive train data). All SWCs must be developed according ASIL D quality standard. Since on HW level an ASIL decomposition was performed, resulting in 2x ASIL B(D) performance hosts, two options on SW level arise:

   - Development of 1 ASIL D SWC per functional SW block, hosted on both performance hosts
   - Development of 2 ASIL B(D) SWCs per functional SW block

   The selected approach regarding SWC design is the development of 2 diverse ASIL B(D) SWCs per mission controller for sensor fusion, object detection, trajectory planning and steering/drive train functionality. A runtime monitor SWC, hosted on the safety host and developed according to ASIL D quality standard is responsible for monitoring the performance host's SWCs' status and the fail-over handling. Figure 3.20 shows the layered SW architecture of the primary mission controller of the 1oo2D approach. The diverse SWCs are marked with suffices A.1 and A.2, the counterparts on the secondary mission controller must be developed ensuring sufficient degree of independence to avoid common mode failures. Also the runtime monitor SWC on both mission controllers must fulfil the independence criteria to stick to the categorization regarding FCRs (see Section 3.5.1).

ii) *Dataflow definition:* The dataflow between SWCs and external components (sensors, VCU) of the ADS-CP is displayed in Figure 3.21. Originating from the sensor set, consisting of camera, LIDAR and RADAR, the data is transmitted via redundant communication channels (Ethernet, CAN) to the Runtime monitor SWCs on the mission controller's SHs. Once basic checks on the sensor data were successful, it is forwarded to the sensor fusion components simultaneously and further processed by the ADF algorithms. The calculated result (steering and drive train control data) again is checked by the runtime monitor SWCs before the active mission controller transmits it to the VCU via the redundant vehicle communication channels.

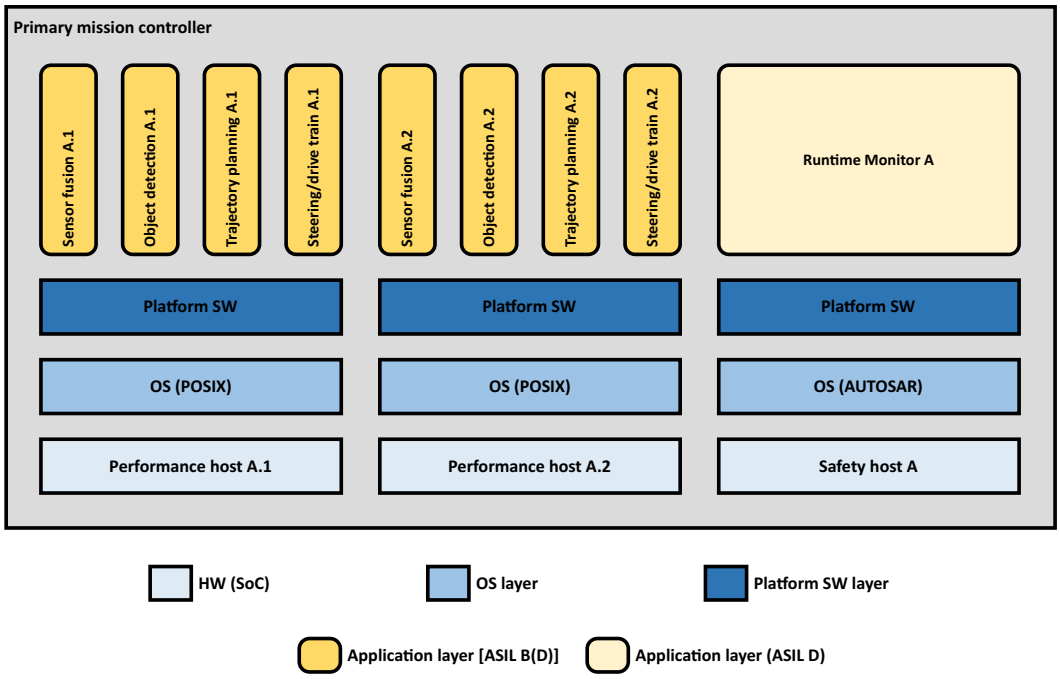

Figure 3.20: Software architecture of the 1oo2D approach (SWCs per host)

The approach of processing complex sensor data (like the sensor set data) in more than two or more diverse streams must deal with the problem of replica indeterminism. Concepts like approximate agreement or inexact voting are potential solutions to mitigate replica indeterminism [42]. Since this is out of scope of this thesis, it is assumed, that two diverse processing streams (suffix A.1/A.2 and B.1/B.2 in Figure 3.21) produce exactly the same output for the same input data in the same point of time (in the fault-free case).

In addition to the checks performed on the sensor datasets and on the calculated steering/drive train data by the runtime monitor, the status of each SWC on the mission controller are monitored continuously. Besides that, the mission controller wide status is exchanged between both mission controllers to agree on the active and passive states. This is covered by the fail-over mechanism, described in the next paragraph.

iii) *Fail-over mechanism:* Based on the state of the art research on fail-over mechanisms (see Section 2.2) a customized hierarchical heartbeat approach is defined for the 1oo2D FO architecture.

According to the defined system requirements (see Section 3.2), no maximum fail-over time is specified, however, to minimize the fail-over time a hot-standby approach is chosen. Implementing a cold-standby fail-over mechanism is critical in terms of

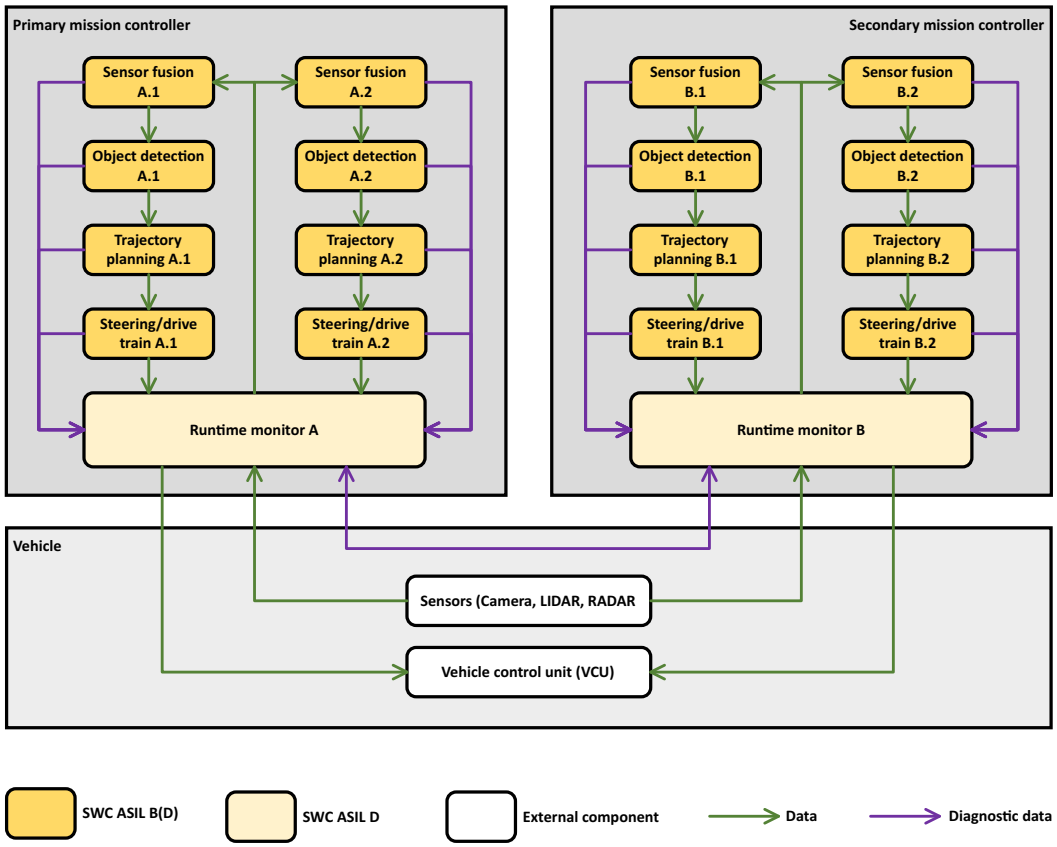

Figure 3.21: Software architecture of the 1oo2D approach (dataflow)

fulfilling the required fail-over time limits, because of considerably high RTOS boot times compared to the application specific timing requirements[8].

The selected fail-over mechanism can be categorized as outlined in Figure 3.22. The heartbeats are sent periodically by the observed SWCs without being triggered by the monitor, hence it is a *push* configuration. It is a *stateless* mechanism, because explicit data is being sent (heartbeats) instead of tracking states using application specific exchanged information. There are two levels of monitoring:

- Monitoring of SWCs hosted on the PHs (sensor fusion, object detection, trajectory planning, steering/drive train) by the runtime monitor SWCs hosted on the SHs
- Both runtime monitor SWCs, hosted on the primary and secondary mission controller are monitoring each other by observing exchanged heartbeats

[8]E.g. in [24] a requirement regarding a maximum fail-over time of 300ms is defined.

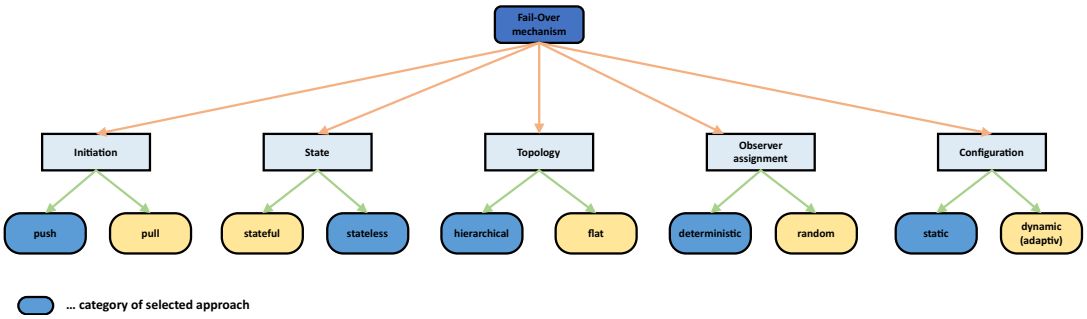

Figure 3.22: Categorization of the fail-over mechanism

Compared to a *flat* topology, where an observer is monitoring other components while not being monitored by a component on another level, a 2-level *hierarchical* approach is defined here. The observer assignment is done in a *static*, *deterministic* way to cope with safety critical determinism and simplicity requirements.

The key components that implements the described fail-over mechanism are the runtime monitor SWCs hosted on the SH on the mission controllers. Main tasks of the runtime monitor is to maintain the mission controller's ECU state (active, inactive). In the initial configuration of the ADS-CP the primary mission controller is in active state (sending the calculated steering/drive train data is activated), whereas the secondary mission controller is inactive (sending of calculated data is disabled)[9]. In Table 3.8 all tasks to be performed by the runtime monitor SWCs are listed.

If an error is detected on a mission controller a mode switch to safe state (inactive) is performed (no outputs are provided anymore), as covered in Section 3.5.3. By not sending heartbeats to the redundant mission controller anymore, it is notified that an error occurred and that the redundant mission controller has to take over operation (mode switch to active). The timing behaviour of the fail-over mechanism is highly determined by the diagnostic measures and the static schedule of the (time-triggered) system. For the PHs one monitoring and one processing task are defined per SWC. The processing task executes the SWC's main functionality (sensor fusion, object detection, trajectory planning, calculation of steering and drive train data) and is not further investigated here. The monitoring task performs diagnostic checks on the corresponding processing task and sends the heartbeat to the runtime monitor SWC on the SH in case the checks were passed. The static schedule of all monitoring tasks is represented in Figure 3.23.

[9] Another approach of handling the fail-over is to move the voting activity to the consuming component (VCU), while primary and secondary mission controllers continuously send their calculated results in distinct time slots (considering a time triggered architecture). The consuming component uses the results of the primary mission controller, as long as data is received, otherwise the results of the secondary mission controller are used. This approach reduces complexity from the ADS-CP and adds complexity to the VCU instead.

#	Component
T1	Periodic reception of sensor data via CAN and Ethernet communication links, verification of the sensor data and distribution to the sensor fusion SWCs hosted on the PH
T2	Periodic reception and verification of results from steering/drive train SWCs (problem of replica-indeterminism is neglected) and distribution to the VCU
T3	Periodic reception of diagnostic information (heartbeats) from all SWCs hosted on the PH
T4	Periodic reception of diagnostic information (heartbeats) from the redundant mission controller's runtime monitor via CAN and Ethernet communication links
T5	Periodic transmission of diagnostic information (heartbeats) to the redundant mission controller's runtime monitor via CAN and Ethernet communication links
T6	Monitoring of all supply voltages provided by the power management component

Table 3.8: Tasks of the runtime monitor SWCs

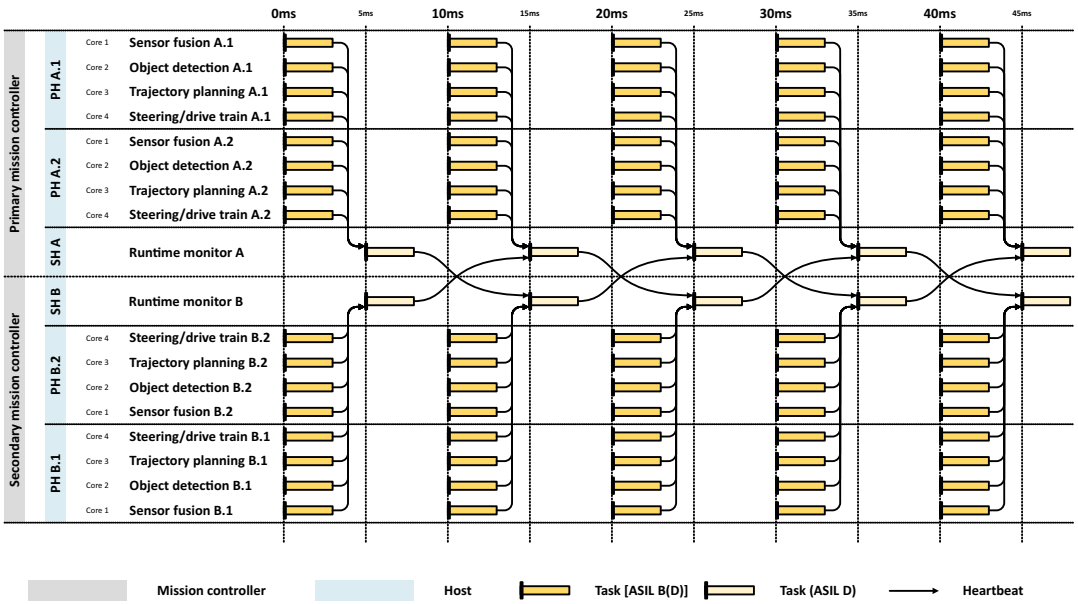

Figure 3.23: Static schedule of tasks contributing to fail-over mechanism

All monitoring tasks are configured to be scheduled periodically every $10ms$. All SWCs on a PH run on distinct cores, hence the tasks can be executed in parallel at the same trigger points. The offset of the triggering points for the runtime monitor tasks (on the SHs) is $5ms$ compared to the triggering points for all monitoring tasks on the PHs, to optimize the event chain formed by the heartbeat transmission. The vertical line labelled with $0ms$ is the starting point of the scheduler, which implicates that the system SW on all hosts is powered up, (time) synchronized and the schedulers are started. The monitoring tasks on the PHs perform diagnostic checks on the processing components and transmit heartbeats to the responsible runtime monitor task if no error was detected. The runtime monitor tasks receive and verify the heartbeats from the monitoring tasks and initiate another heartbeat transmission to the runtime monitor tasks scheduled on the redundant mission controller in case no error was detected. Once the runtime monitor tasks are triggered for the second time ($25ms$) they also receive and check the heartbeats from the redundant mission controller's runtime monitor. The calculation of the worst case FOTI is derived based on Figure 3.24.

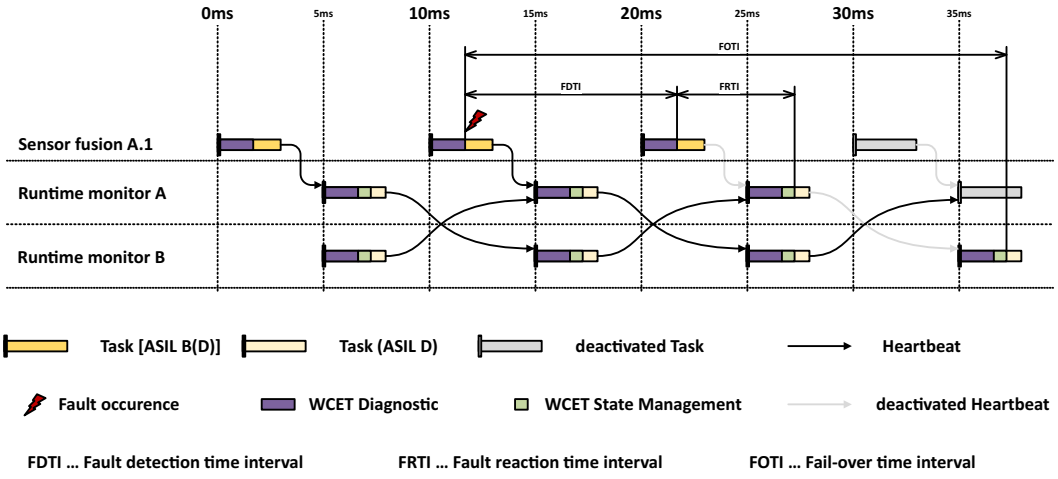

Figure 3.24: Visualization of the worst case fail-over time

The Fault Detection Time Interval (FDTI) [8] encompasses its maximum value if the fault occurs right after finishing a diagnostic procedure. Hence, the worst case FDTI is equal to the configured period t_{SF_period} of the associated monitoring task (see Figure 3.24, Equation 3.1).

$$FDTI = t_{SF\_period} \tag{3.1}$$

The time period between the detection of a fault and the associated reaction on

it is defined as Fault Reaction Time Interval (FRTI) [8]. The fault reaction on mission controller level is considered to be a mode transition to a safe state, whereas the fault reaction of the ADS-CP is the activation of the redundant mission controller. Equation 3.2 covers the calculation of the FRTI, which depends on the triggering point offset between monitoring task on the PHs and runtime monitor task on the SHs ($t_{SF\_RM\_offset}$), the difference between the Worst Case Execution Times (WCETs) of the diagnostic procedures inside the monitoring tasks ($\text{WCET}_{RM\_diag} - \text{WCET}_{SF\_diag}$) and the WCET of the state management procedure inside the runtime monitoring task ($WCET_{RM\_st\_mgmt}$).

$$FRTI = \begin{array}{ll} t_{SF\_RM\_offset} & +WCET_{RM\_diag} \\ -WCET_{SF\_diag} & +WCET_{RM\_st\_mgmt} \end{array} \tag{3.2}$$

The fault reaction of the overall system (ADS-CP) is covered implicitly in the FOTI (see Equation 3.3). It is composed of parameters FDTI and FRTI and the time period between deactivating the faulty mission controller and activating the redundant one, which is the configured period of the runtime monitor tasks (t_{RM_period}).

$$FOTI = FDTI + FRTI + t_{RM\_period} \tag{3.3}$$

The model for the calculation of FDTI, FRTI and FOTI neglects parameters related to time synchronization between the hosts (e.g. jitter) and latencies between finishing runtime monitor task and achieving the new state (active/inactive), as well as introduced overhead by the system SW (e.g. latency introduced by the OS or communication stack). In practice, using this approach of immediately assuming a fault after missing only one heartbeat might result in an unacceptable number of false positives, depending on the robustness of the system (e.g. reliability of used communication links). Hence, assuming a fault after n missing heartbeats might be a better approach. However, this would result in an increased FDTI ($+n * FDTI$) and thus also in an increased FOTI.

The exchanged heartbeats among all components contributing to the fail-over mechanism are based on UDP datagrams on the transport layer since the potential overhead introduced by using Transmission Control Protocol (TCP) instead delays the heartbeat transmission [57], [41].

3.5.3 FSC

In the FSC the high level safety measures of Table 3.3 are refined and traced to the safety goals and the fault hypothesis' covered faults.

SWC development according to ISO26262

To satisfy the determined ASILs as a result of the HARA (see Section 3.3), all SWCs would have to be developed according ASIL D. Since the SWCs implementing ADFs are hosted on PHs with ASIL B capability, an ASIL decomposition of the SWCs according to [8] is done.

The runtime monitor SWC, hosted on the SHs is developed according to ASIL D, since the hosting HW has ASIL D capability and therefore no ASIL decomposition is required here.

Covered safety goals: **SG7-SG17**

HW development according to ISO26262

In order to be able to host ASIL D SWCs, the selected HW components (PMIC, SH, PH, CAN transceivers, Ethernet transceivers, Ethernet switch) must be capable and the development must be done according to ASIL D (decomposed to ASIL B for PHs).

Covered safety goals: **SG7-SG17**

Functional safety measures implemented in SWCs

To compensate loss of sensor data appropriate measures (e.g. interpolation) must be implemented in the ADF SWCs. If a permanent loss of sensor data (at least 2 types) occurs, an emergency operation must be implemented in the trajectory planning SWC (controlled breaking manoeuvre until standstill).

Covered safety goals: **SG1**, **SG2**, **SG4**

E2E protection of data sent over communication channels

All data sent over any communication channel must be checked to ensure its integrity. Adding an E2E protection (CRC, sequence counter) to all data being sent (CAN messages, Ethernet frames) ensures proper error detection.

Covered safety goals: **SG1-SG3**, **SG7-SG8**

Redundancy and diversity

On vehicle level the ADS-CP's power supplies, the sensor sets and the communication channels provided to the ADS-CP must be implemented in a redundant and diverse way (CAN, Ethernet). Each sensor set uses a dedicated communication channel for data transmission and each mission controller is supplied by a dedicated power supply to minimize the risk of common cause faults.

The ADS-CP must provide independent interfaces to the redundant and diverse communication channels (CAN, Ethernet) on vehicle level.

Covered safety goals: **SG4**, **SG5**, **SG7**, **SG8**

State management

The 1oo2D approach requires specific ECU state management for both mission controllers and a fail-over mechanism to provide FT capabilities. The defined measures to fulfil the requirements regarding state management are:

- **Voltage monitoring:** All voltages provided to the mission controller and provided by the PMIC must be monitored by the SH. In case any voltage level deviates from the specified range, a transition to a safe state (no outputs are produced anymore) is initiated.
- **Host monitoring:** The host monitoring is done on various levels. The PMIC monitors the SH (e.g. question/answer protocol) and the PHs are monitored by the SH (e.g. question/answer protocol). If any host does not respond according to the specified behaviour, a transition to a safe state (no outputs are produced anymore) is initiated.
- **Shutdown path:** The PMIC is responsible of maintaining the safe state, either if the SH does not respond or the SH notifies the PMIC to perform a switch to the safe state. This requires the deployment of highly reliable transceivers, capable of ensuring a safe state of the ECU by not occupying the communication channels, even if not supplied anymore.
- **Fail-over mechanism:** If an error is detected on the active mission controller, a fail-over to the redundant unit must be performed. This is handled by the runtime monitor SWC, hosted on the SHs (see Section 3.5.2).

Covered safety goal: **SG6**

Trace to fault hypothesis: **DCHWF1-DCHWF7**, **DCSWF1-DCSWF3**

Degradation strategy

Once a fail-over to the redundant mission controller was performed because an error was detected, the driver must be alerted that the vehicle is operating in a degraded mode [8]. If the error on the faulty unit disappears after a restart (transient fault), the ADS-CP is able to operate in normal mode again and an appropriate notification to the driver must be initiated again.

Trace to fault hypothesis: **DCHWF7**

3.6 Hybrid architecture

In Section 2.1.5 some proposals for utilizing a hybrid architectural approach to be used for ADSs are outlined based on a state of the art research. The generic pattern of the hybrid architecture is displayed in Figure 3.25.

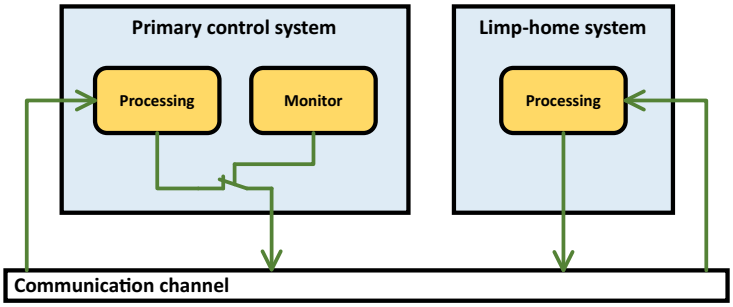

Figure 3.25: Hybrid architectural pattern

A primary control system, equipped with a high diagnostic coverage, is designed as fail-silent system. As soon as a fault is detected on the primary unit, it transitions into a safe state and stops sending data to the communication channel. A limp-home system, designed with less features, continuously sends its calculated results to the actuator. The voting activity regarding which data to be used is done on the actuator. As long as data from the primary control system is available on the actuator it is used. Once data from the primary control system is not received, the results from the limp-home system are used and the vehicle operates in a degraded mode.

3.6.1 Fault hypothesis

The separation of the fault hypothesis into HW and SW related faults is performed according to the selected methodology in Section 3.1.3.

Hardware faults

The high level component decomposition of the item definition (see Figure 3.9) is a major input for the separation into FCRs. Compared to the fully redundant 1oo2D architecture (see Section 3.5) the primary unit (primary control system) and fallback unit (limp-home system) use dedicated communication channels (see Figure 3.26).

Any fault in the power supply, power management, controller, external communication or communication channel components affect the operation of a complete control system (either primary control system or limp-home system), hence, the FCRs are defined according to Figure 3.26 and contain all aforementioned components.

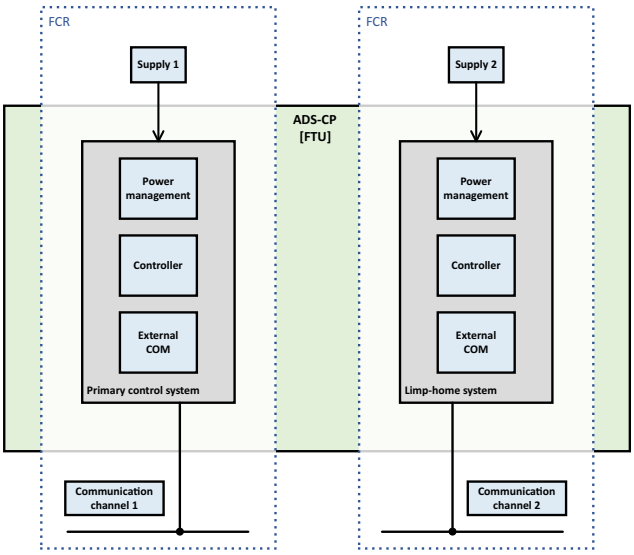

Figure 3.26: FCR decomposition of the Hybrid approach (hardware components)

Several assumptions for the HW related part of the fault hypothesis are defined:

- A single fault hypothesis is used. More than one fault at the same time are considered to be rare events (uncovered faults). This is commonly accepted and used e.g. in [4] and [53].
- The temporal classification of the covered faults (see Table 3.9) are considered to be either transient or permanent.
- All other faults (not covered in Table 3.9) are assumed to be rare events. A NGU strategy is defined to react on these uncovered faults (see Section 3.7).
- The communication channels don't create correct frames spontaneously. Further they won't add any arbitrary delay when sending/receiving data [54].

The covered hardware faults are listed in Table 3.9.

Software faults

The architectural pattern of the Hybrid approach (see Figure 3.25) and the high level component decomposition (see Figure 3.9) are the main inputs for the SW related definition of the fault hypothesis. Similar to the previously described approaches, the application layer is located on top of a platform SW and OSs. The considered layers for the definition of the FCRs are:

- System SW (OS and platform SW)

#	Component	Fault	Impact
HCHWF1	Supply 1, Supply 2	Voltage level of power supply below the specified range	Primary control system/Limp-home system not supplied anymore, no output can be produced
HCHWF2	Supply 1, Supply 2	Voltage level of power supply above the specified range	Primary control system/Limp-home system must shut down to prevent potential damage, no output produced anymore
HCHWF3	Power management (primary control system)	Power management fails to provide the required voltage levels for an arbitrary component	Primary control system should perform a transition to a safe state
HCHWF4	Power management (Limp-home system)	Power management fails to provide the required voltage levels for an arbitrary component	Data integrity must be ensured by HW design
HCHWF5	Controller (primary control system)	Controller provides no result on all interfaces (fail-silent behaviour)	Primary control system must perform a transition to a safe state to prevent possible incorrect results
HCHWF6	Controller (Limp-home system)	Controller provides no result on all interfaces (fail-silent behaviour)	Data integrity must be ensured by HW design
HCHWF7	External COM	External communication component fails to forward received data from the bus to the controller	Primary control system/Limp-home system must perform a transition to a safe state to prevent possible incorrect calculations
HCHWF8	External COM	External communication component fails to forward received data from the controller to the bus	Primary control system/Limp-home system must perform a transition to a safe state to prevent possible incorrect calculations
HCHWF9	Bus 1, Bus 2	Communication channel not ready to transmit data due to fault on the physical level	Transition to a degraded mode must be initiated

Table 3.9: Hybrid fault hypothesis: covered hardware faults

- Application SW (sensor fusion, object detection, trajectory planning, steering/drive train, monitor)

Primary control system and limp-home system both use the same system SW, hence the defined FCRs for the platform SW and the OS (see Figure 3.27) are assumed to be free of any type of faults to remove the risk of a single point of failure in the ADS-CP.

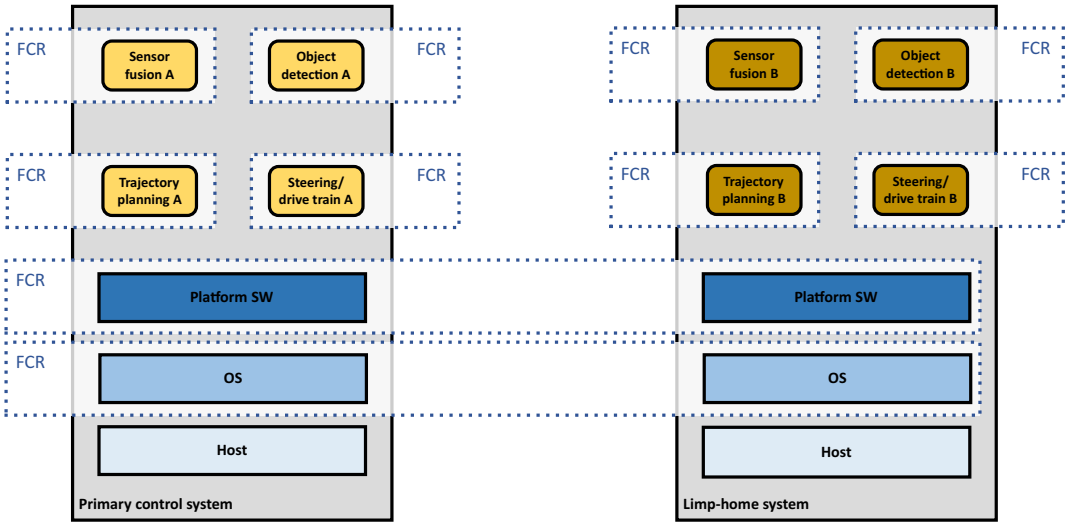

Figure 3.27: FCR decomposition of the Hybrid approach (software components)

All SWCs on application level form separate SWCs, since a detected fault is contained within a SWC and not affecting other SWCs. It must be ensured that the SWCs hosted on the primary control system and on the limp-home system are developed sufficiently independent to avoid the risk for common mode failures.

Assumptions to be considered for the definition of the covered SW faults are:

- A single fault hypothesis is used. More than one fault at the same time are considered to be a rare event (uncovered fault). This is commonly accepted and used e.g. in [4] and [53].
- The temporal classification of the covered software faults (see Table 3.10) are permanent faults by definition.
- All other faults (not covered in Table 3.10) are assumed to be rare events. A NGU strategy is defined to react on these uncovered faults (see Section 3.7).
- The FCR containing the system SW is assumed to be fault-free (as assumed e.g. in [53]).

- The system SW handles the communication between SWCs and between primary control system/limp-home system and vehicle. Since the absence of faults in the platform SW and in the OSs is assumed, failure modes like arbitrary timing/value message failures in the communication system don't have to be considered.

Following these assumptions, the defined covered SW faults are listed in Table 3.10.

#	Component	Fault	Impact
HCSWF1	All SWCs (application level)	SWC produces no output data	Primary control system: transition to safe state; Limp-home system: data integrity must be ensured
HCSWF2	All SWCs (application level)	SWC produces incorrect output data (detectable by consumer)	Primary control system: transition to safe state; Limp-home system: data integrity must be ensured
HCSWF3	All SWCs (application level)	SWC fails to fetch input data	Primary control system: transition to safe state; Limp-home system: data integrity must be ensured

Table 3.10: Hybrid fault hypothesis: covered software faults

3.6.2 SAD

As outlined in Section 3.1.3, output artefacts from several design steps (item definition, HARA, architectural pattern, fault hypothesis, FSC) are used for the SAD. Again, HW and SW related parts are described separately.

HW architecture

Following to the hybrid architectural pattern, the HW architecture of the primary control system (i) and the limp-home system (ii) are described in the following paragraphs. Both control systems are deployed on the same ECU, which is accepted here because faults related to spatial proximity are not covered in the fault hypothesis.

i) *Primary control system:* The computational part of the primary control system is composed of high performance processing units and a safety controller. Since PHs with ASIL B capability are available, an ASIL decomposition must be performed to satisfy the requirement for hosting ASIL D SWCs [8]. All hosts of the primary control system (2x PH, 1x SH) are connected via an Ethernet backbone, realized via an Ethernet switch (see Figure 3.28). A power management unit (Power management A) is responsible for providing all necessary voltage levels to the deployed components forming the primary control system. The power management unit is controlled

by the safety controller, enabling mode transitions to fail-silent. Compared to the Simplex (Section 3.4) and 1oo2D (Section 3.5) approaches, the primary unit is not connected via redundant communication channels to the vehicle. Primary and fallback node use distinct, diverse communication channels in the Hybrid approach.

ii) *Limp-home system:* The limp-home system of the ADS-CP using the Hybrid approach is not intended to host SWCs of the same ASIL as the hosted SWCs on the primary control system, hence no ASIL decomposition must be performed for the fallback system's high performance computational part (assuming that an ASIL B capability is sufficient). This can be justified by the reduced operation time of the vehicle in degraded mode once a fault on the primary control system occurred. In order to satisfy the diversity requirement related to independency of the used communication links, CAN is used for the limp-home system (compared to Ethernet on the primary control system). The power management for the redundant system (Power management B) provides all required voltage levels for the PH and the CAN transceiver (see Figure 3.28) and is not explicitly controlled by any component, since no fail-silent behaviour is required here[10].

SW architecture

The derived SWCs (i) from the high level component decomposition (see Section 3.2), their assignment to hosts and the dataflow between SWCs, system SW and vehicle (ii) are covered in the following paragraphs.

i) *SWC design:* The SW architecture of the primary control system in the Hybrid approach is similar to the primary and secondary channel's architecture of the 1oo2D approach.

As already described for the Simplex (Section 3.4) and 1oo2D (Section 3.5) approach, a layered architecture, consisting of OS layer, platform SW and application layer is used. On the PHs a POSIX compliant OS is deployed (e.g. QNX [55]), while a AUTOSAR [45] OS (e.g. MICROSAR [56]) is utilized on the SH. Basic services like communication and scheduling are handled by the platform SW on top of the OSs. The limp-home system's SWCs (see Figure 3.29), hosted on top of the system SW (OS and platform SW) on Performance host B, are developed to satisfy a safety integrity level of ASIL B. To ensure that the primary control system and the limp-home system fail independently with respect to SW faults, the SWCs (sensor fusion, object detection, trajectory planning, steering/drive train) on all PHs must be developed guaranteeing a sufficient level of diversity.

The Monitor SWC, hosted on the primary control system's SH (see Figure 3.30) verifies the results of its PHs and is responsible for the mode management of the

[10] Since the voting activity is shifted to the actuator, received data from the fallback unit is not used as long as the primary unit operates correctly.

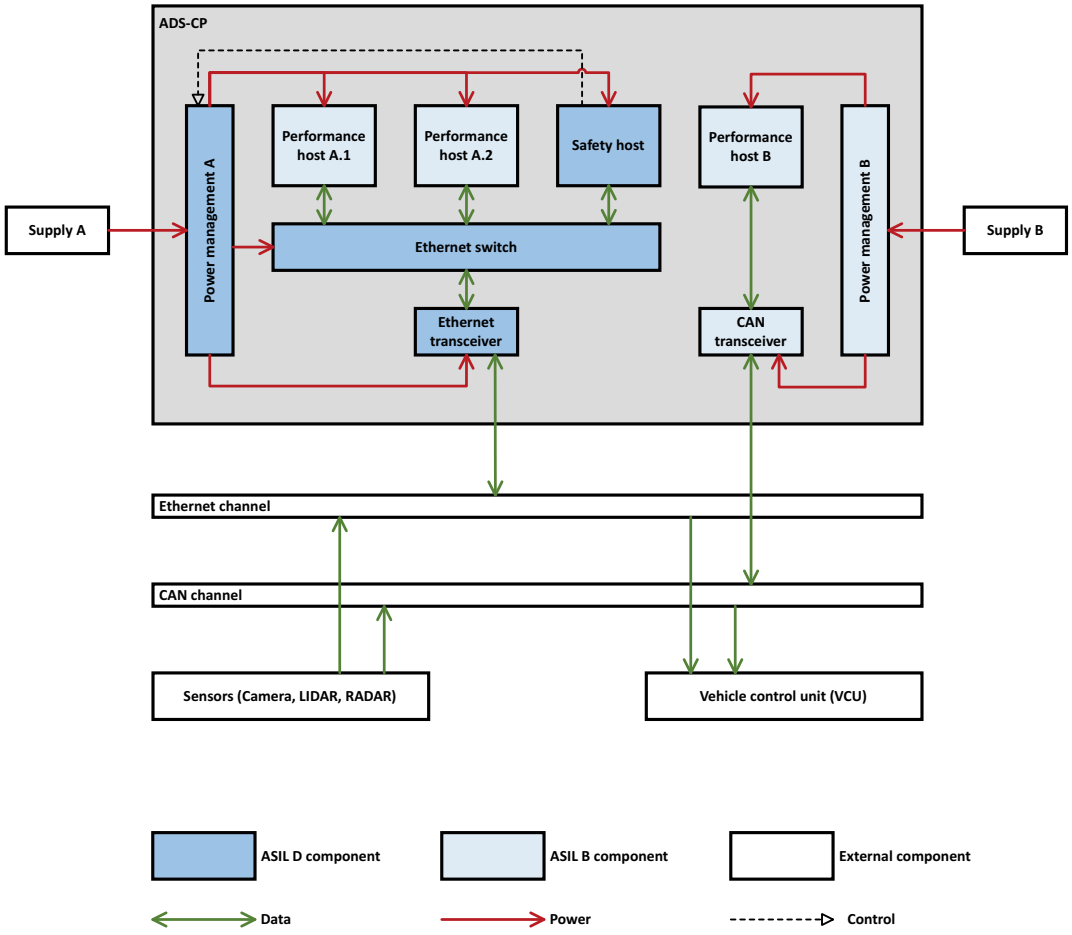

Figure 3.28: Hardware architecture of the Hybrid approach

primary node. The problem of replica indeterminism is neglected here, as already done in the Simplex (Section 3.4) and 1oo2D (Section 3.5) approaches.

ii) *Dataflow definition:* Camera, LIDAR and RADAR data is sent via Ethernet to the primary control system's sensor fusion SWCs (Sensor fusion A.1, Sensor fusion A.2). The sensor set data is further processed by the PHs and sent to the monitor SWC to be verified (see Figure 3.31)[11]. The monitor SWC sends the successfully verified results to the VCU via Ethernet. If the verification was not successful, the primary control system performs a transition to a safe state (no data sent anymore to the VCU).

In parallel, the limp-home system's sensor fusion SWC (Sensor fusion B) receives

[11]The problem of replica indeterminism is neglected in this investigation. Possible mitigation strategies are proposed in [42].

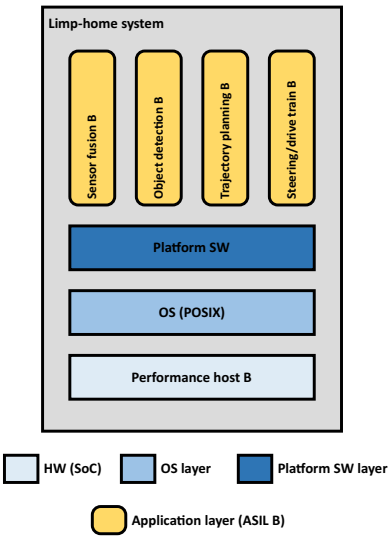

Figure 3.29: Software architecture of the Hybrid approach (SWCs per host) - limp-home system

sensor data via a CAN communication channel. In the end of the processing chain (sensor fusion, object detection, trajectory planning, steering/drive train) the results are sent to the VCU using the CAN interface (see Figure 3.31).

3.6.3 FSC

According to the used methodology (Section 3.1.3), the high level safety measures (defined in Table 3.3), are refined and referenced to the fault hypothesis (covered faults) and to the safety goals.

SWC development according to ISO26262

Several safety goals are satisfied by developing SWCs according to the functional safety standard ISO 26262 [8], targeting ASIL D. On the primary control system, the SWCs (hosted on the PHs) require a ASIL decomposition because of the PHs' safety integrity level (see Section 3.6.2) to ASIL B(D).

On the limp-home system, SWC development according to ASIL B is sufficient for the Hybrid approach, which can be argued with the limited vehicle operation time in the degraded mode.

Covered safety goals: **SG7-SG17**

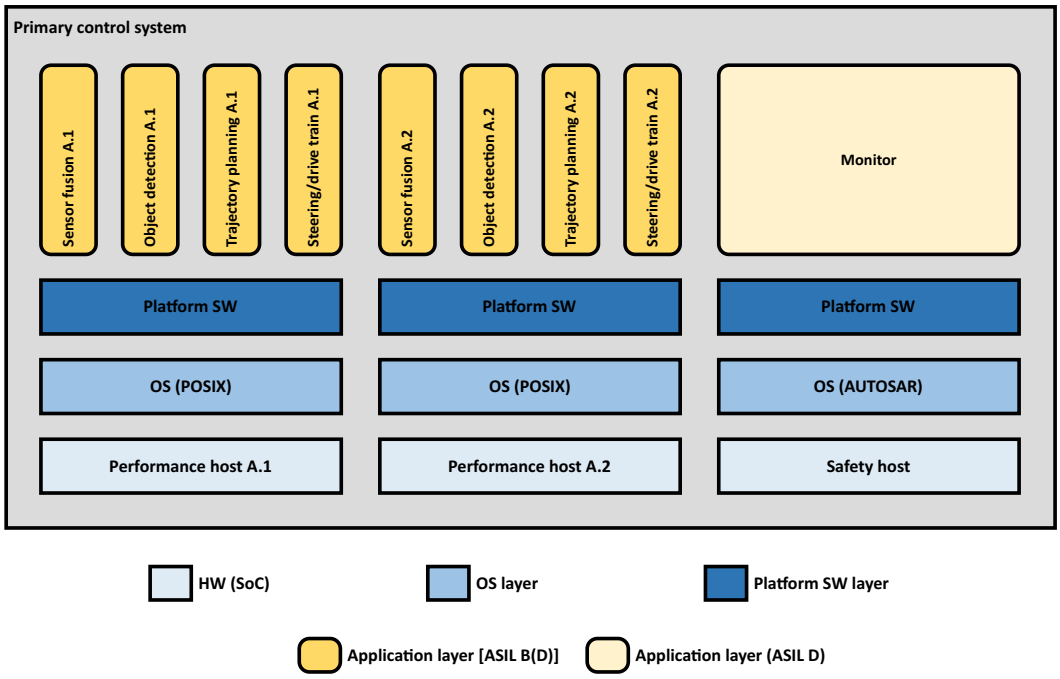

Figure 3.30: Software architecture of the Hybrid approach (SWCs per host) - primary control system

HW development according to ISO26262

The primary control system must be capable of hosting ASIL D SWCs, hence the HW (Power management A, Safety host, Ethernet switch, Ethernet transceiver) must be developed according to ASIL D, too, while the PHs are decomposed to ASIL B(D).

The limp-home system's HW (Power management B, Performance host B, CAN transceiver) is sufficient to be developed according to ASIL B, since the vehicle operation time is reduced due to the degraded mode.

Covered safety goals: **SG7-SG17**

Functional safety measures implemented in SWCs

To compensate loss of sensor data appropriate measures (e.g. interpolation) must be implemented in the ADF SWCs. If a permanent loss of sensor data (at least 2 types) occurs, an emergency operation must be implemented in the trajectory planning SWC (controlled breaking manoeuvre until standstill).

Covered safety goals: **SG1**, **SG2**, **SG4**

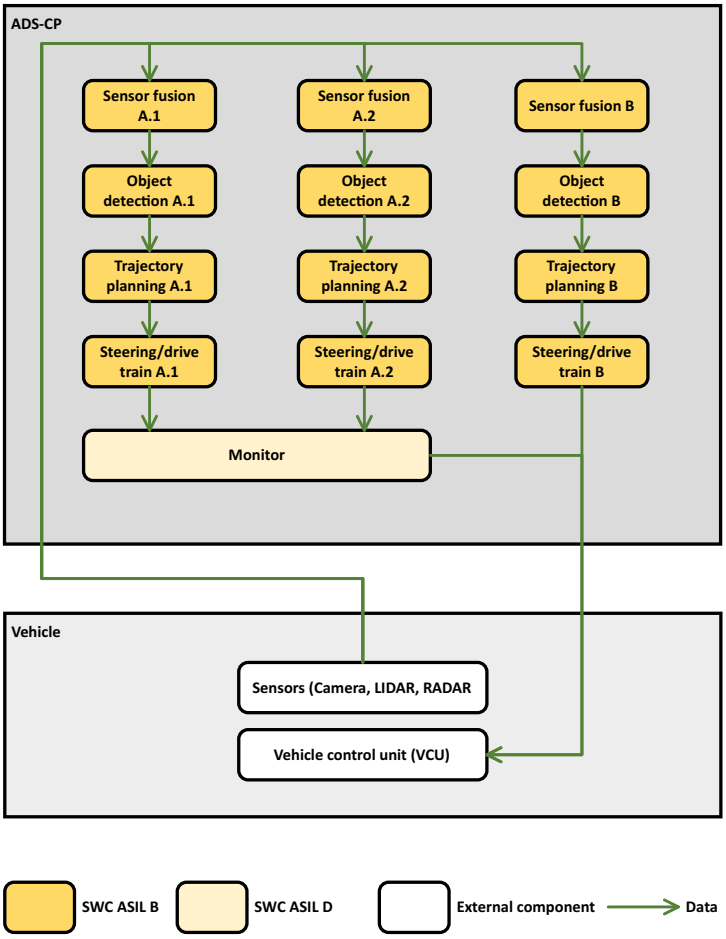

Figure 3.31: Software architecture of the Hybrid approach (dataflow)

E2E protection of data sent over communication channels

All data sent over any communication channel must be checked to ensure its integrity. Adding an E2E protection (CRC, sequence counter) to all data being sent (CAN messages, Ethernet frames) ensures proper error detection.

Covered safety goals: **SG1-SG3**, **SG7-SG8**

Redundancy and diversity

On vehicle level the ADS-CP's power supplies, the sensor sets and the communication channels provided to the ADS-CP must be implemented in a redundant and diverse way (CAN, Ethernet). Each sensor set uses a dedicated communication channel for data transmission and primary control and limp-home system are supplied by dedicated power

supplies to minimize the risk of common cause failures.

The ADS-CP must provide independent interfaces to the redundant and diverse communication channels (CAN, Ethernet) on vehicle level, where each control system uses a dedicated communication channel. The limp-home system is added as redundant system to the primary control system (Hybrid architecture) and continues vehicle operation in a degraded mode once a fault occurred on the primary control system. The voting activity is shifted from the ADS-CP to the actuator (VCU). The primary control system must ensure fail-silent behaviour in case a fault is detected, otherwise the voting component on the VCU uses the primary control system's invalid results. Monitoring of the supply voltage levels is one measure to satisfy the fault hypothesis with respect to covered faults. The power management components on both controllers must ensure proper safety measures to react on over voltage of the power supplies (i.e. shutdown of the controller). The data integrity of the steering/drive train data, sent by the limp-home system must be ensured by selecting proper HW components.

Covered safety goals: **SG4-SG8**

Trace to fault hypothesis: **HCHWF1-HCHWF8**, **HCSWF1-HCSWF3**

Degradation strategy

Once a fault occurred on the primary control system, the driver must be alerted that the vehicle operates in a degraded mode in terms of features and remaining operation time [8]. If the error of the primary control system disappears after a restart (transient fault), the ADS-CP is able to operate in normal mode again and an appropriate notification to the driver must be initiated.

Trace to fault hypothesis: **HCHWF9**

3.7 Common NGU strategy

If a fault occurs that is not covered in the fault hypothesis, a NGU strategy should be defined, e.g. on vehicle level. The proposed NGU strategy to mitigate rare events (i.e. not-covered faults) affecting the ADS-CP is to manoeuvre the vehicle into a safe state by braking while maintaining the last received steering angle. This could be initiated by the VCU if no data from the ADS-CP is received at all, or if a applied integrity check on the received data fails persistently. Performing a reboot of the ADS-CP is not considered to be a feasible approach because of stringent timing requirements of the (critical) steering/drive train data[12].

[12] In [24] an exemplary maximum fail-over time of $300ms$ is required.

4 FO demonstrator

In Chapter 3 three FO architectures, suitable for ADSs, were investigated. The 1oo2D approach is selected (Section 4.2) and used for implementing a specific use case, defined in Section 4.1. This first proof of concept is developed according to prototype quality level, but automotive processes and toolchains are used. Finally, the prototypical implementation, performed in Section 4.3 is evaluated with respect to FO relevant parameters in Section 4.4.

4.1 Use case definition

Prior to the selection of a FO architecture to be implemented, a simple use case for the prototypical ADS-CP is defined. Therefore it is assumed, that a single SWC, requiring high computational performance, shall be executed on a suitable host, while FO behaviour is ensured by choosing an appropriate FO architecture.

The SWC to be hosted could implement one of the identified software related functional blocks of the high level component decomposition (sensor fusion, object detection, trajectory planning, steering/drive train), as performed in Section 3.2. Since the prototypical implementation focusses on the FO behaviour (and not e.g. on high computational performance), a pseudo implementation of a AI component is used. This SWC periodically generates and transmits random data to a Personal Computer (PC) used for evaluation purposes. The evaluation makes use of certain fault injection methods, performed on the prototypical ADS-CP, to trigger the investigated fail-over activity.

4.2 Selection of a FO architecture

Selecting one of the investigated FO architectures (see Chapter 3) is driven by the availability of prototypical ECUs on the one hand[1] and the applicability of investigating the fail-over mechanism of the prototypical ADS-CP on the other hand.

The fail-over mechanism of the Simplex architecture (see Section 3.4) is implemented implicitly in the SH's decision module SWC. Since all fail-over activities are managed inside one SWC and are not measurable via external interfaces of the ADS-CP per default, the Simplex architecture is not considered for the prototypical ADS-CP implementation.

Triggering and evaluating a fail-over activity is a key aspect in Section 4.3 and Section 4.4. The voting activity in the Hybrid architecture (Section 3.6) is shifted from the

[1]Since developing a prototypical ECU (HW) is far beyond out of scope of the thesis, Commercial Off-The-Shelf (COTS) components are the preferable choice for implementing the ADS-CP.

ADS-CP to the actuator, hence the Hybrid architecture is not chosen for the prototypical ADS-CP implementation, since this thesis is focussing on the ADS-CP level rather than on the vehicle level.

The 1oo2D approach (see Section 3.5) fits best to the criteria for selecting a FO architecture to be implemented. Since the fail-over mechanism of the 1oo2D architecture is based on exchanging heart beats between the primary and secondary mission controllers, the requirement for a simple fail-over activity measurement is met. However, the prototypical ADS-CP does not fully comply with the defined SAD in Section 3.5.2 due to the following reasons:

- The prototypical implementation of the ADS-CP does not require development according to any ASIL, hence no ASIL decomposition has to be applied on the PH.
- The main focus of the evaluation is on the fail-over mechanism, hence there is no need to apply HW or SW diversity on the primary or secondary mission controller.
- No redundant communication channel is required for the prototypical implementation of the ADS-CP.
- Instead of the SW related functional blocks, defined in the high level component decomposition (see Figure 3.9), one dummy AI SWC is implemented, periodically sending random data.

The simplifications compared to the 1oo2D architecture in Section 3.5 are specified in the following sections.

4.2.1 Simplified SAD

The simplified HW architecture of the ADS-CP prototype is composed of two separate ECUs. Since no diversity is required among both mission controllers, the same COTS prototypical ECU can be used for the primary and secondary mission controllers (see Figure 4.1).

No ASIL requirements are considered for the prototypical implementation, so no ASIL decomposition needs to be performed for the PH. Hence, the high computational part of the mission controller is composed of one PH.

Ethernet is used as communication channel between both mission controllers and a PC, used for evaluation tasks. No redundant and diverse communication channel (e.g. CAN) or power supply is considered for the prototype. Both ECUs are connected to a power supply and a fault injection on a mission controller can be performed e.g. by disconnecting it from the power supply.

Compared to the SW architecture of the 1oo2D approach investigated in Section 3.5.2 only one SWC is hosted on the PHs, denoted with Fusion A and Fusion B. For the

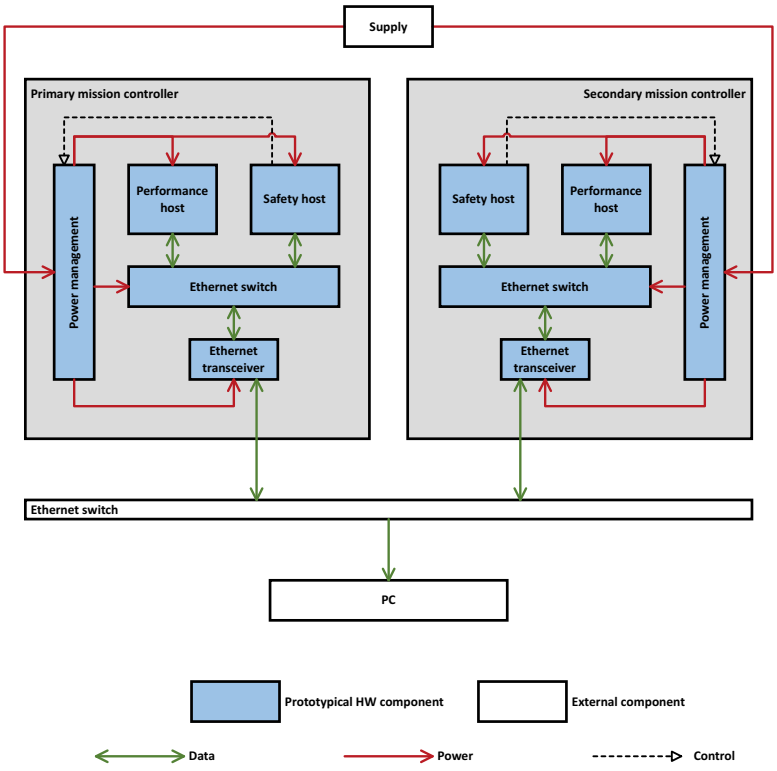

Figure 4.1: HW architecture of the ADS-CP prototype

prototypical implementation of the ADS-CP no ASIL related requirements are applicable, so all SWCs in Figure 4.2 (Fusion, Runtime monitor) are developed with prototype quality.

The dataflow between the SWCs and the PC is represented in Figure 4.3. The fail-over management is implemented through the Runtime monitor SWCs by retrieving diagnostic information from the monitored Fusion SWCs and by exchanging periodic heartbeats. The Fusion SWC of the active mission controller periodically sends random data to the PC used for evaluating the fail-over mechanism.

The fail-over concept to be implemented is based on the investigated high level concept in Section 3.5.2 and defined in more detail in Section 4.3.

4.2.2 Reliability model

Assumptions regarding failure rates of the assigned failure modes are used as input for the reliability model. For investigating the reliability of the selected 1oo2D approach the simplified model, described in Table 4.1 is used.

Four distinct system states are defined for the markov model (see Figure 4.4), using the

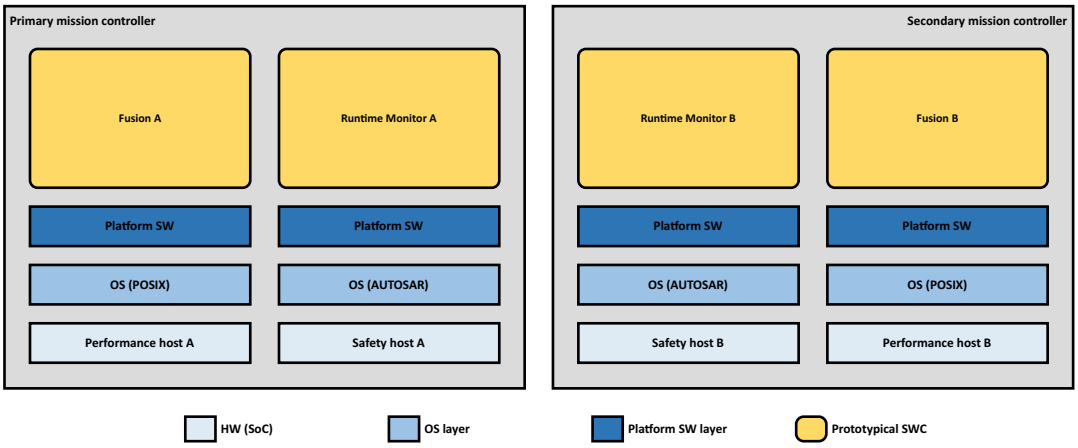

Figure 4.2: SW architecture of the ADS-CP prototype (SWCs per host)

Symbol	Description	Value
λ_P	Permanent, independent failure rate (HW and SW)	1.000 Failure In Time (FIT)
λ_T	Transient, independent failure rate (HW and SW)	100.000 FIT
λ_{CCF}	Common cause failure rate (HW and SW)	100 FIT
c	Assumption coverage	90%
μ	Repair rate of transient faults	$\{0,1,10\}\frac{1}{h}$

Table 4.1: Markov parameters of the reliability model

parameters defined in Table 4.1 as input:

- S0: State of primary and secondary unit is **OK**
- S1: Transient fault occurred on primary or secondary unit - **Degraded** state (recoverable)
- S2: Permanent fault occurred on primary or secondary unit - **Degraded** state (non recoverable)
- S3: State of primary and secondary unit is **Not OK**

The model-checking tool *PRISM* [58] is used to evaluate the markov model and prove that the overall reliability of the selected FO architecture is higher than the reliability of

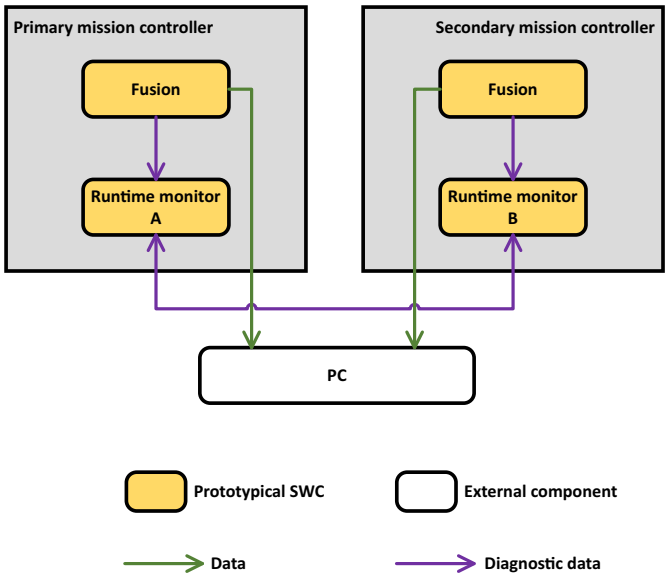

Figure 4.3: SW architecture of the ADS-CP prototype (dataflow)

a single unit.

The reliability of the overall system using three different values for the repair rate μ is displayed in Figure 4.5, where the dependency of the reliability on the repair rate is evident (the higher the repair rate, the higher the reliability).

Besides the reliability graph of Figure 4.5, the MTTF is a common parameter to characterize a system's reliability. Evaluating the markov model of Figure 4.4 using *PRISM*, the following values for the MTTF and the duration of running and degraded states are determined in Table 4.2.

μ	MTTF	t_{OK}	$t_{degraded\_t}$	$t_{degraded\_p}$
$0\frac{1}{h}$	$13,854.53h$	$4,948.05h$	$8,818.30h$	$88.18h$
$1\frac{1}{h}$	$46,025.19h$	$45,211.35h$	$8.14h$	$805.75h$
$10\frac{1}{h}$	$46,051.97h$	$45,244.81h$	$0.81h$	$806.34h$

Table 4.2: Model evaluation using *PRISM*

The evaluation of the model verifies that increasing the repair rate (e.g. from $1\frac{1}{h}$ to $10\frac{1}{h}$) does not necessarily increase the MTTF significantly. This is because at a certain point of time the permanent (unrecoverable) degraded state prevails and therefore is mainly determining the MTTF.

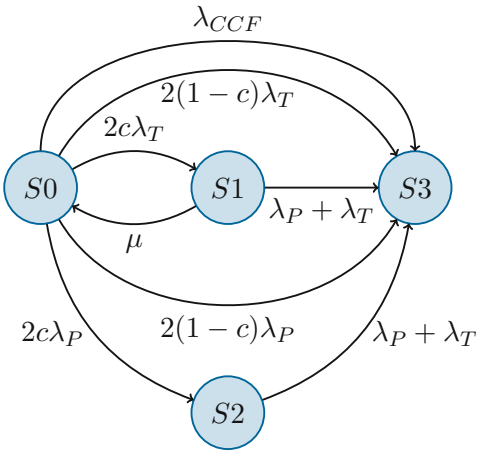

Figure 4.4: Markov model of the 1oo2D approach

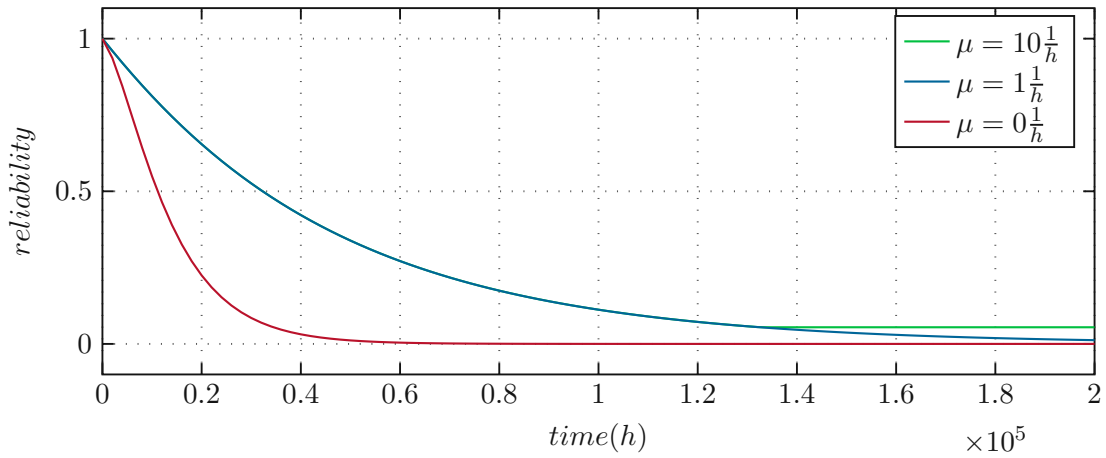

Figure 4.5: Evaluation of the reliability

4.3 Implementation

For the prototypical implementation of the ADS-CP TTTech's Advanced Driver Assistance System (ADAS) ECU RazorMotion [48] is used. Primary and secondary mission controller are composed of one RazorMotion each, however, only the identified components of the simplified SAD in Section 4.2.1 are utilized.

As platform SW TTTech's MotionWise [47] is implemented on all hosts, where communication and scheduling services are deployed. For the OS layer the AUTOSAR implementation of Vector (Microsar [56]) is deployed on the SHs while on the PHs a Linux distribution of the Yocto project [59] is used.

The selected HW and SW components are implemented using the automotive toolchain described in Figure 4.6, separated into the following phases:

- System model definition
- System model integration
- Application development
- Building and flashing
- Static schedule visualization[2]

The system model definition is the initial phase of the defined approach and is based on the (simplified) SW related artefacts of the SAD (see Section 4.2.1). In this phase the SWCs, their runnables, interfaces and parameters like runnable periods are defined using a 3rd party AUTOSAR modelling tool, following a platform specific modelling guide.

According to the SW architecture, defined in the SAD, two SWCs are defined per mission controller:

- Runtime monitor (hosted on the SH)
- Fusion (hosted on the PH)

The diagnostic data (heartbeats), exchanged between the SWCs, is composed of one data element (sequence counter), implemented as an unsigned integer (32 bit). This sequence counter is periodically incremented inside the monitored SWC (Fusion or Runtime monitor A/B) before the heartbeat is sent to the observing SWC (Runtime monitor A/B). The observer periodically checks the received sequence counter to verify whether the monitored SWC is still alive or not.

The data sent by the (active) Fusion SWC consists of an array (120 elements) of unsigned integer (8 bit) values. It is periodically filled with random numbers and sent via

[2]The optional visualization of the generated static schedule is used for evaluating the implemented fail-over mechanism.

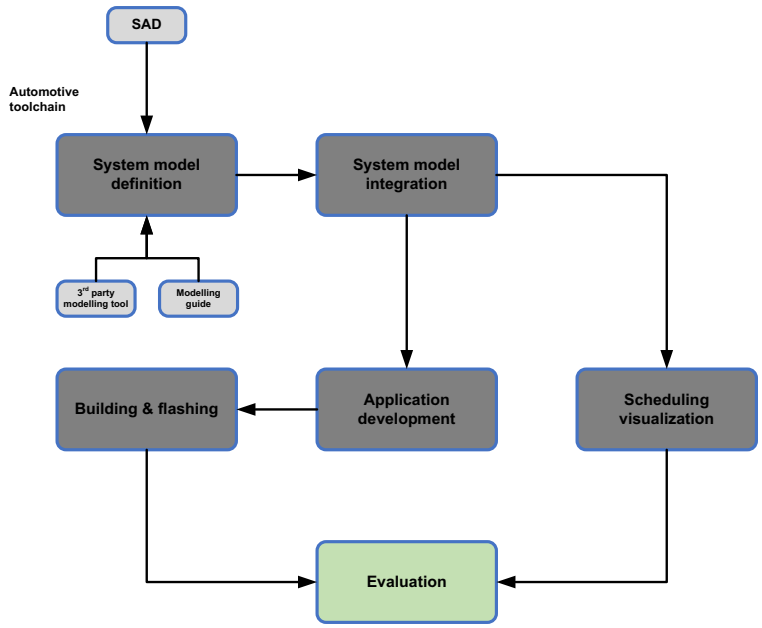

Figure 4.6: Implementation according to an automotive toolchain approach

Ethernet to the PC to evaluate the fail-over mechanism.

Both SWCs include cyclically triggered runnables (RRuntimeMonitor, RFusion) with a configured period of 10ms. No additional scheduling constraints (e.g. earliest activation time, deadline, ...) are configured for the evaluation, since optimizing the fail-over mechanism is not the main focus here.[3]

Once the system model definition is finalized, the model must be integrated into the platform SW. MotionWise offers a toolchain to integrate the AUTOSAR conform system model, assuming it complies with the provided modelling guide. The platform toolchain includes all necessary steps to generate implementation templates for the SWCs, configuration and generation of the AUTOSAR stack (using a 3rd party AUTOSAR configuration tool) and configuration and generation of the platform specific embedded code [47].

The implementation templates, generated in the system model integration phase, are the entry point for the application development. The templates include all defined runnables and provide an API to access the modelled interfaces. For the ADS-CP prototype, the Runtime monitor and Fusion SWCs must be developed.

[3] By systematically applying scheduling constraints on the runnables (including e.g. runnables of the AUTOSAR Basic Software (BSW)), the event chain from the transmission of the Fusion's heartbeat to the transmission of the Runtime monitor's heartbeat can be optimized, hence, the FOTI can be minimized.

The implementation of the Fusion SWC (runnable RFusion) generates 120 random numbers of type uint8 (unsigned integer 8-bit) each time the execution of the runnable is triggered (10ms period). Those random numbers are sent via Ethernet to the PC if the mission controller is active. Otherwise the Ethernet communication SWC (part of the platform) is deactivated to ensure the fail-silent behaviour of the redundant unit.

Runnable RRuntimeMonitor (part of SWC Runtime monitor) implements the fail-over concept defined in Section 3.5.2. A maximum error counter of 2 is defined for this prototype, meaning that after the 2nd missing heartbeat (received from the Fusion SWC or from the redundant Runtime monitor SWC) a state change is triggered.

As soon as the application development is finalized, the binaries can be built and finally flashed onto the targets (RazorMotions). Additionally, a visualization of the generated static schedule is created and used in Section 4.4 to compare it with the measurements for evaluation purposes.

4.4 Evaluation

To evaluate the fail-over mechanism of the implemented (simplified) 1oo2D architecture a test setup as shown in Figure 4.7 is established. Primary and secondary mission controllers, implemented on TTTech's RazorMotion ECUs, are connected via Ethernet using an unmanaged Ethernet switch. In addition, a PC is connected to sniff the active mission controller's generated packets using Wireshark [60].

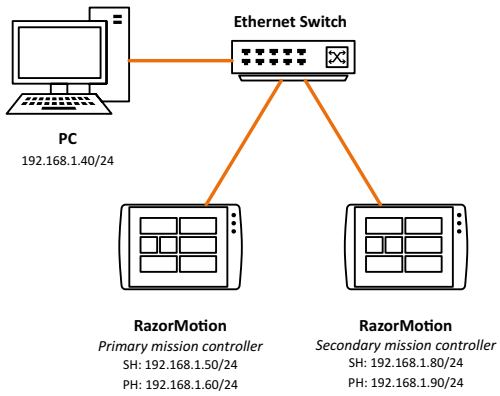

Figure 4.7: Test setup of the prototypical ADS-CP

The Internet Protocol (IP) addresses of all hosts are statically configured according to Figure 4.7.

To compare and evaluate the measured results, the output of the scheduling visualization step of Section 4.3 is used to calculate the FOTI (best case and worst case scenario). Since

in the implemented prototypical ADS-CP primary and secondary mission controllers are not synchronized, the best case and worst case FOTI not only depends on the timing of the fault occurrence, but also on the time offset between the primary and secondary mission controller's schedule. In Figure 4.8 the best case scenario regarding fault timing and schedule offset between both mission controllers for the minimal FOTI is displayed.

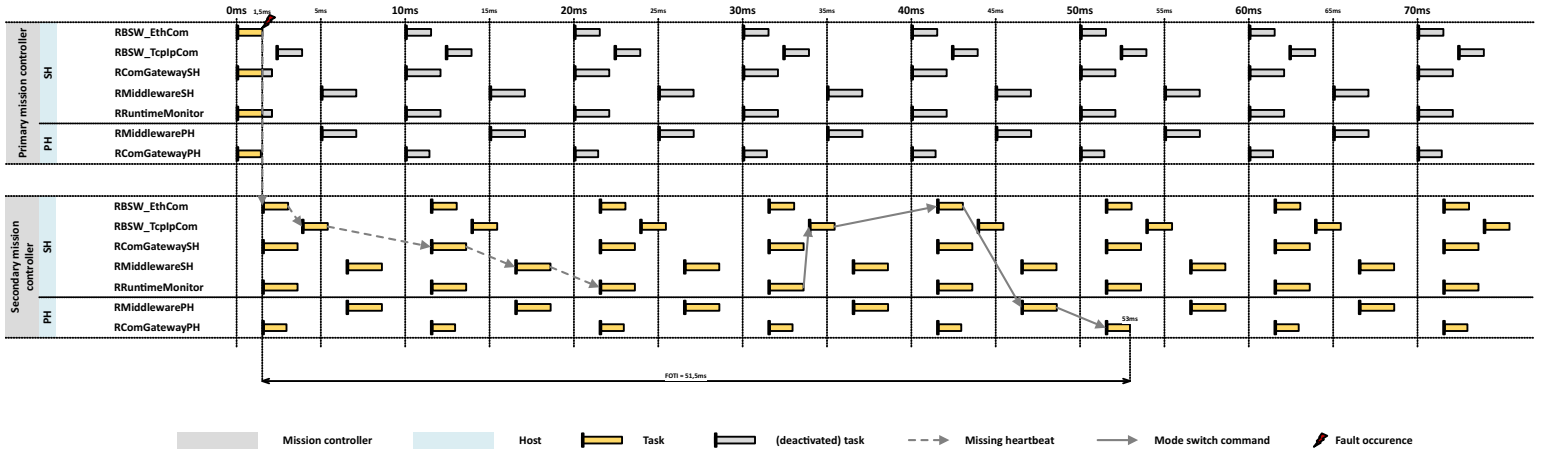

Figure 4.8: Static schedule and FOTI visualization (best case scenario)

This visualized schedule only considers relevant runnables for the calculation of the FOTI. These runnables are further described in Table 4.3.

Runnable name	Type	Host	Period	Description
RBSW_EthCom	AUTOSAR BSW	SH	10ms	AUTOSAR BSW specific runnable handling the Link layer [61] related communication (Receive (Rx) and Transmit (Tx))
RBSW_TcpIpCom	AUTOSAR BSW	SH	10ms	AUTOSAR BSW specific runnable handling the Transport layer (TCP, UDP) and Internet layer (IP) [61] related communication (Rx and Tx)
RComGatewaySH	Middleware	SH	10ms	Gateway SWC between AUTOSAR communication stack and application SWCs
RMiddlewareSH	Middleware	SH	10ms	Core SWC of the used middleware (MotionWise [47])
RRuntimeMonitor	Application	SH	10ms	Monitoring SWC of the implemented FO approach
RMiddlewarePH	Middleware	PH	10ms	Core SWC of the used middleware (MotionWise [47])
RComGatewayPH	Middleware	PH	10ms	Gateway SWC between OS communication stack and application SWCs

Table 4.3: Relevant runnables for the FOTI calculation

The minimal FOTI is experienced if the fault occurs immediately before a heartbeat is sent by the primary mission controller on the one hand, and if the offset between both mission controller's schedule yields to a minimal delay between sending and receiving a heartbeat via runnables *RBSW_EthCom* on the other hand. The secondary mission controller's runtime monitor (runnable *RRuntimeMonitor*) detects the missing heartbeat after the delay introduced by the event chain labelled as *missing heartbeat*. A maximum error counter of 2 is defined (see Section 4.3), so the mode switch on the secondary mission controller (inactive to active) is triggered after the second missing heartbeat, detected by the runtime monitor SWC. The event chain to enable the external Ethernet communication on the PH's runnable *RComGatewayPH* is labelled as *mode switch command* in Figure 4.8. The time period between the first sent Ethernet frame of the secondary mission controller's Fusion SWC (via runnable *RComGatewayPH*) and the occurred fault on the primary mission controller adds up to a calculated minimal FOTI of $FOTI_{min} = 51.5ms$.

The worst case scenario regarding fault timing and offset between primary and secondary mission controller's schedule, leading to the maximum FOTI, is displayed in Figure 4.9. Similar to the calculation of the minimal FOTI, event chains for the (missing) heartbeat and the mode switch command are used to visualize the time period between the occurrence of a fault on the primary mission controller and the activation of sending Ethernet frames by the secondary mission controller.

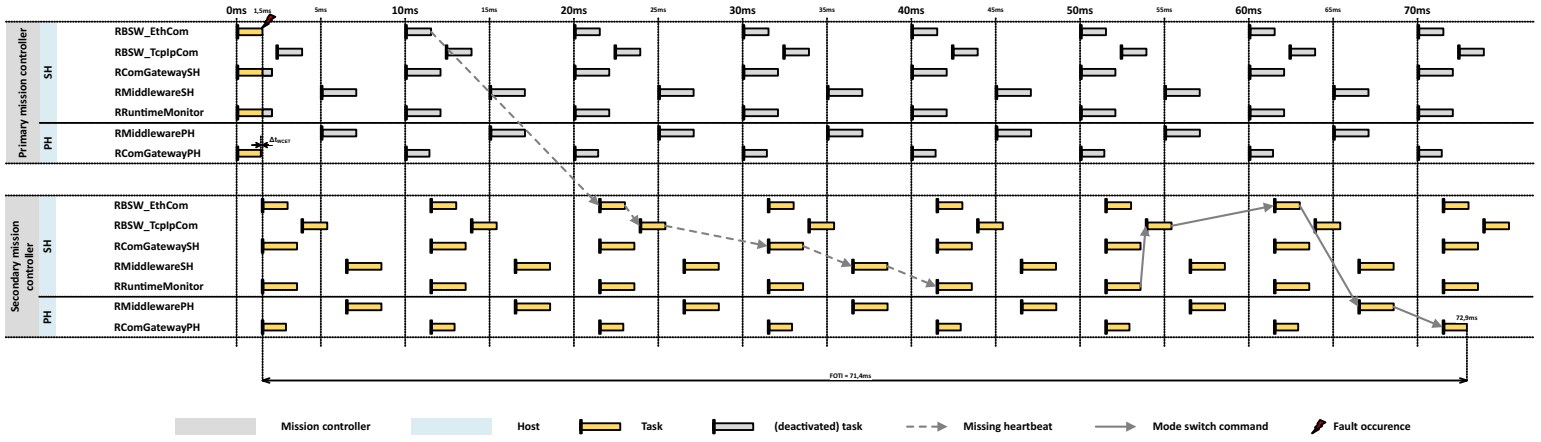

Figure 4.9: Static schedule and FOTI visualization (worst case scenario)

To calculate the maximum FOTI, the fault must occur immediately after the heartbeat is sent by the primary mission controller (runnable *RBSW_EthCom*). Additionally, the scheduling offset between both mission controllers must maximize the delay between heartbeat transmission by the primary mission controller and heartbeat reception by the secondary mission controller. This is experienced if runnable *RBSW_EthCom* on the secondary mission controller starts right before runnable *RBSW_EthCom* on the primary mission controller ends. Based on the event chains in Figure 4.9, $FOTI_{max} = 71.4ms$ is calculated.

The calculation of the minimum and maximum FOTI neglects the following parameters:

- Transmission delay and jitter of the Ethernet frames exchanged between primary and secondary mission controllers (via runnables *RBSW_EthCom*)
- Delay introduced by the OS's communication stack on the PHs
- Scheduling jitter (activation time jitter)
- Timing differences (offset, drift) between SHs and PHs[4]
- Only WCETs of the runnables are used for the calculations

The measurement is expected to result in a FOTI between the calculated time interval:

$$FOTI_{min} = 51.5ms <= FOTI_{measured} <= FOTI_{max} = 71.4ms \tag{4.1}$$

Table 4.4 shows the relevant section of the sniffed network traffic, where the fail-over between primary and secondary mission controllers is experienced. The delta time between the first sent Ethernet frame by the secondary mission controller's PH (source IP address 192.168.1.90) and the last sent Ethernet frame by the primary mission controller's PH (source IP address 192.168.1.60) is $60.355ms$.

Since this measured time interval allocates the period between the last (complete) execution of the primary mission controller's runnable *RComGatewayPH* and the first execution of the secondary mission controller's runnable *RComGatewayPH* (in active mode), it's not meaningful to directly compare this value to the calculated FOTIs (which considers the time instant where the fault occurs). The exact timing of the injected fault is not known for the performed measurement, since the HW voltage buffering delays the voltage drop on the ECU after disconnecting it from the power supply. However, the difference between the assumed WCET parameters of runnables *RBSW_EthCom* and *RComGatewayPH* (Δt_{WCET} in Figure 4.9) is small compared to the calculated FOTIs ($< 0.1ms$) and therefore it is neglected. Hence, the measured delta time can be considered as measured FOTI and fits to the calculated range of FOTIs.

[4]In the implemented prototypical ADS-CP, Generalized Precision Time Protocol (gPTP) is used to synchronize the SH's and PH's local clocks. The quality of the synchronization is not further evaluated here and (unavoidable) differences between the local clocks are neglected.

Time (sec)	Δ Time (sec)	Source	Destination	Protocol	Length (Byte)
...					
61.014852	0.009692	192.168.1.60	192.168.1.40	UDP	120
61.025136	0.010284	192.168.1.60	192.168.1.40	UDP	120
61.035044	0.009908	192.168.1.60	192.168.1.40	UDP	120
61.045246	0.010202	192.168.1.60	192.168.1.40	UDP	120
61.055089	0.009843	192.168.1.60	192.168.1.40	UDP	120
61.115444	0.060355	192.168.1.90	192.168.1.40	UDP	120
61.125648	0.010204	192.168.1.90	192.168.1.40	UDP	120
61.135404	0.009756	192.168.1.90	192.168.1.40	UDP	120
61.145550	0.010146	192.168.1.90	192.168.1.40	UDP	120
61.155417	0.009867	192.168.1.90	192.168.1.40	UDP	120
...					

Table 4.4: Ethernet traffic of the ADS-CP captured with Wireshark [60]

The calculated and measured maximum FOTI of 71.4ms (see Equation 4.1) implies, that a minimum application specific deadline of 71.4ms can be handled while fulfilling the minimum QoS value of 1 (see Figure 3.1). An application deadline higher than 71.4ms results in a QoS value greater than 1 with the implemented approach, while a smaller application deadline would result in a QoS value smaller than 1 and hence, (critical) deadline misses would occur.

5 Conclusion

In this thesis FO strategies enabling ADSs of SAE levels 3+ were analysed. At first a state of the art research on existing standards and taxonomies related to AD, on FO architectures (including fail-over mechanisms) and on existing AD platforms was performed. Based on that, three FO architectures (Simplex, 1oo2D and Hybrid) suitable to host ADFs were selected and investigated in detail, following a proposed methodology approach, based on existing standards and publications.

The 1oo2D architecture was implemented in a proof of concept prototypical setup, following an automotive toolchain approach. The experiments performed in this thesis show in general that the proposed concepts implemented are feasible for the application in practice and thus contributing to the development of autonomous vehicles of the future.

5.1 Comparison of FO architectures

In order to simplify and harmonize the investigations of the selected FO architectures several assumptions are defined in the fault hypothesis of the approaches. These (restrictive) assumptions are mainly used to compare the approaches and evaluate their applicability to enable ADSs.

The architectural pattern of the *Simplex* approach defines a voting component, selecting either the mission controller's or the base controller's output. This is a major weakness of this architecture for safety critical systems like ADSs, since the voter introduces a single point of failure. To mitigate this drawback, further redundancy and diversity must be added to the voting component (e.g. via a TMR approach), since the assumption of deploying a voter free of any types of faults is not valid in practice. Furthermore, the assumption that the platform software is fully verified and therefore free of faults is likely unrealistic. This must be checked in existing platform software's safety manuals. The fact that the voting component forms a single point of failure if no additional measures are applied currently eliminates this approach to be utilized in ADSs.

The most restrictive assumption in the 1oo2D and Hybrid architectures' fault hypothesis considered to be not valid in practice is the free-of-faults assumption regarding the system SW. This can be mitigated by using diverse OSs and platform SW implementations in both approaches. However, this must be investigated individually anyway for each (series targeting) project, since this thesis only provides a high level overview and comparison of the architectures. Other than this limiting factor, the 1oo2D and Hybrid architectures are considered to be capable of enabling ADSs, hosting ADFs of SAE levels 3+. Since the

1oo2D approach utilizes fully redundant and equivalent channels, the valid operation time in degraded mode is increased compared to the degraded mode of the Hybrid approach.

5.2 Future challenges

Currently, a major challenge is to improve existing or develop new standards to be applied in particular in the AD domain. The existing functional safety standard *ISO26262* [8] does not cover the ADS lifecycle sufficiently (see Section 1.2.2). The *J3016* taxonomy [6] is vague e.g. with respect to timing specifications (e.g. required minimum fail-over time for specific SAE levels of driving automation), too. This should be covered by upcoming standards (e.g. *ISO/CD TR 4804* [10]). Providing standards with a maximum meaningful coverage of AD related topics is of interest in industry to provide equal opportunities among the OEMs and suppliers.

In order to satisfy requirements related to a minimal power consumption of ADSs, a pure hot standby approach might not be feasible, hence further research must be done to achieve required fail-over times while the startup time of PHs currently is quite high.

The evaluation of the prototypical ADS-CP showed that implementing the fail-over mechanism on application level is not optimal, since the latency introduced by the system SW is quite high. Future work should focus on adding fault-tolerance capabilities to a platform SW and potentially utilize HW features of the deployed hosts to minimize the FOTI. Since the proposed FO architectures in this thesis also highly depend on a high fault detection coverage it is essential to deploy suitable failure detection mechanisms to reduce the error detection latency and increase the fault detection coverage to a maximum.

Besides the mentioned challenges related to standardization and technical issues, several other challenges (e.g. legislation, social aspects, etc.) require synchronized future research and development of several stakeholders in order to enable automated driving of SAE driving automation levels 3 and above.

A Minimal HARA of a ADS-CP

#	Malfunction	Operational scenario	Impact	Exposure	Severity	Controllability	ASIL	Safety Goal
1	Single camera/RADAR/LIDAR frame not received by ADS-CP (other 2 sensor type frames still available)	Highway/Urban/Rural	response time (motion control) slightly decreased - for one frame period	E4	S3	C3	ASIL D	A single invalid, corrupt or lost frame of 1 sensor type must not lead to any unsafe situation
2	No camera/RADAR/LIDAR frames received by ADS-CP anymore (other 2 sensor type frames still available)	Highway/Urban/Rural	fusion capabilities and therefore quality of motion control significantly decreased; manoeuvre into safe state possible with degraded functionality	E4	S3	C3	ASIL D	Permanent invalid, corrupt or lost frames of 1 sensor type must not lead to any unsafe situation
3	Single invalid camera/RADAR/LIDAR frame received by ADS-CP (other 2 sensor type frames still available)	Highway/Urban/Rural	response time (motion control) slightly decreased - for one frame period;	E4	S3	C3	ASIL D	A single invalid, corrupt or lost frame of 1 sensor type must not lead to any unsafe situation
4	Multiple invalid camera/RADAR/LIDAR frames received by ADS-CP (other 2 sensor type frames still available)	Highway/Urban/Rural	fusion capabilities and therefore quality of motion control significantly decreased	E4	S3	C3	ASIL D	Permanent invalid, corrupt or lost frames of 1 sensor type must not lead to any unsafe situation
5	Single, valid camera/RADAR/LIDAR frame received by ADS-CP containing wrong data (other 2 sensor type frames still available)	Highway/Urban/Rural	fusion capabilities and therefore quality of motion control slightly decreased for a single frame period	E4	S3	C3	ASIL D	Invalid or corrupt frames must be detected
6	Multiple camera/RADAR/LIDAR frames received by ADS-CP containing wrong data (other 2 sensor type frames still available)	Highway/Urban/Rural	quality of motion control decreases significantly; possible calculation of bad trajectories causing catastrophic events	E4	S3	C3	ASIL D	Invalid or corrupt frames must be detected
7	2 out of 3 sensor type frames not received by ADS-CP anymore (only frames for 1 sensor type still received)	Highway/Urban/Rural	quality of motion control decreases to a very low level, normal operation not possible anymore; possible catastrophic hazards;	E4	S3	C3	ASIL D	Permanent loss of frames of at least 2 sensor types must trigger the emergency operation, i.e. an emergency brake using the last calculated steering angle
8	no sensor data received anymore by ADS-CP	Highway/Urban/Rural	no sufficient quality of motion control possible anymore	E4	S3	C3	ASIL D	Permanent loss of frames of at least 2 sensor types must trigger the emergency operation, i.e. an emergency brake using the last calculated steering angle

Table A.1: HARA, functional block *External communication*

#	Malfunction	Operational scenario	Impact	Exposure	Severity	Controllability	ASIL	Safety Goal
9	ADS-CP power supply interrupted	Highway/Urban/Rural	ADAS functionality can't be provided anymore; may lead to catastrophic events	E4	S3	C3	ASIL D	The power supply of the ADS-CP must not be interrupted
10	ADS-CP power supply voltage below lower threshold	Highway/Urban/Rural	ADAS functionality might not be able to be provided for a necessary amount of time anymore; may lead to catastrophic events if no safe state is reached	E4	S3	C3	ASIL D	The power supply must conform to the specified boundaries
11	ADS-CP power supply voltage exceeds upper threshold	Highway/Urban/Rural	ADAS functionality might not be able to be provided for a necessary amount of time anymore; may lead to catastrophic events if no safe state is reached	E4	S3	C3	ASIL D	The power supply must conform to the specified boundaries

Table A.2: HARA, functional block *Power management*

#	Malfunction	Operational scenario	Impact	Exposure	Severity	Controllability	ASIL	Safety Goal
12	steering data not sent by ADS-CP	Highway/Urban/Rural	intended trajectory not followed because missing steering data; leads to catastrophic hazards	E4	S3	C3	ASIL D	Correct steering data must be sent continuously
13	wrong steering data sent by ADS-CP	Highway/Urban/Rural	safe trajectory not followed; leads to catastrophic hazards	E4	S3	C3	ASIL D	Correct steering data must be sent continuously

Table A.3: HARA, functional block *Steering data transmission*

#	Malfunction	Operational scenario	Impact	Exposure	Severity	Controllability	ASIL	Safety Goal
14	drive train/brake data not sent by ADS-CP	Crossover/grade crossing	no acceleration/decceleration according to driving situation possible; might lead to catastrophic hazard (e.g. standing on a grade crossing);	E4	S3	C3	ASIL D	Correct drive train/brake data must be sent continuously
15	wrong drive train/brake data sent by ADS-CP	Highway/Urban/Rural	safe trajectory not followed regarding speed; leads to catastrophic hazards	E4	S3	C3	ASIL D	Correct drive train/brake data must be sent continuously

Table A.4: HARA, functional block *Acceleration/braking data transmission*

#	Malfunction	Operational scenario	Impact	Exposure	Severity	Controllability	ASIL	Safety Goal
16	fusion algorithm does not respond (permanent error)	Highway/Urban/Rural	input data for subsequent algorithms cannot be provided; no trajectory calculation possible	E4	S3	C3	ASIL D	Sensor fusion algorithm mustn't be in a incorrect, not working state for more than 3 frames per second
17	fusion algorithm calculates invalid data (permanent error)	Highway/Urban/Rural	subsequent algorithms cannot perform any calculations because of invalid input data; no trajectory calculation possible	E4	S3	C3	ASIL D	Sensor fusion data mustn't contain incorrect, corrupt or missing data for more than 3 frames per second
18	fusion algorithm calculates wrong data (permanent error)	Highway/Urban/Rural	wrong input data might lead to wrong (unsafe) trajectory planning	E4	S3	C3	ASIL D	Sensor fusion data mustn't contain incorrect, corrupt or missing data for more than 3 frames per second
19	fusion algorithm fails to calculate data and responds with error (permanent error)	Highway/Urban/Rural	input data for subsequent algorithms cannot be provided; no trajectory calculation possible	E4	S3	C3	ASIL D	Sensor fusion algorithm mustn't be in a incorrect, not working state for more than 3 frames per second
20	fusion algorithm provides no/invalid/wrong data for 3 frames at most (transient error)	Highway/Urban/Rural	quality of trajectory calculation temporary decreased	E4	S2	C3	ASIL C	Sensor fusion data mustn't contain incorrect, corrupt or missing data for more than 3 frames per second

Table A.5: HARA, functional block *Sensor fusion*

A Minimal HARA of a ADS-CP

#	Malfunction	Operational scenario	Impact	Exposure	Severity	Controllability	ASIL	Safety Goal
21	object recognition algorithm does not respond (permanent error)	Highway/Urban/Rural	input data for subsequent algorithms cannot be provided; no trajectory calculation possible	E4	S3	C3	ASIL D	Object recognition algorithm mustn't be in a incorrect, not working state for more than 3 frames per second
22	object recognition algorithm fails to execute and responds with error (permanent error)	Highway/Urban/Rural	input data for subsequent algorithms cannot be provided; no trajectory calculation possible	E4	S3	C3	ASIL D	Object recognition algorithm mustn't be in a incorrect, not working state for more than 3 frames per second
23	object is recognized although no object is present (false positive) for some frames	Highway/Urban/Rural	quality of trajectory calculation temporary decreased	E4	S0	C3	QM	Object recognition algorithm mustn't produce more than 6 frames per second containing false positives
24	object is not recognized although object is present (false negative) for some frames	Highway/Urban/Rural	quality of trajectory calculation temporary decreased	E4	S0	C3	QM	Object recognition algorithm mustn't produce more than 3 frames per second containing false negatives
25	object is recognized although no object is present (false positive) permanently	Highway/Urban/Rural	user experience significantly decreased; might also lead to injuries due to unnecessary brake manoeuvres	E4	S1	C3	ASIL B	Object recognition algorithm mustn't produce more than 6 frames per second containing false positives
26	object is not recognized although object is present (false negative) permanently	Highway/Urban/Rural	not detecting obstacles might lead to catastrophic events	E4	S3	C3	ASIL D	Object recognition algorithm mustn't produce more than 3 frames per second containing false negatives
27	object recognition algorithm provides no/invalid/wrong data for 3 frames at most (transient error)	Highway/Urban/Rural	quality of trajectory calculation temporary decreased	E4	S2	C3	ASIL C	Object recognition data mustn't contain poor quality data (no/invalid/wrong data) for more than 3 frames per second

Table A.6: HARA, functional block *Object detection*

#	Malfunction	Operational scenario	Impact	Exposure	Severity	Controllability	ASIL	Safety Goal
28	trajectory planning algorithm does not respond (permanent error)	Highway/Urban/Rural	no trajectory planning leads to catastrophic events	E4	S3	C3	ASIL D	Trajectory planning algorithm mustn't be in an incorrect, not working state for more than 1 frame per second
29	trajectory planning algorithm doesn't find any safe trajectory (permanent error)	Highway/Urban/Rural	no safe trajectory (including full braking) planning leads to catastrophic events	E4	S3	C3	ASIL D	Trajectory planning algorithm must provide a trajectory with minimum threat to human lives (pedestrians, driver, other drivers) in case no safe trajectory could be found
30	trajectory planning algorithm fails to execute and responds with error (permanent error)	Highway/Urban/Rural	no trajectory planning leads to catastrophic events	E4	S3	C3	ASIL D	Trajectory planning algorithm mustn't be in an incorrect, not working state for more than 1 frame per second
31	trajectory planning algorithm calculates unsafe trajectory (permanent error)	Highway/Urban/Rural	using an unsafe trajectory leads to catastrophic events	E4	S3	C3	ASIL D	Trajectory planning algorithm mustn't provide no or an unsafe trajectory for more than 1 frame per second
32	Trajectory calculation provides no or an unsafe trajectory for some frames (transient error)	Highway/Urban/Rural	a missing or an unsafe trajectory for some frames might already lead to catastrophic events	E4	S3	C3	ASIL D	Trajectory planning algorithm mustn't provide no or an unsafe trajectory for more than 1 frame per second

Table A.7: HARA, functional block *Trajectory planning*

A Minimal HARA of a ADS-CP

Acronyms

1oo2D	1-out-of-2 with Diagnostics
2oo2D	2-out-of-2 with Diagnostics
2oo3	2-out-of-3
AD	Automated Driving
ADAS	Advanced Driver Assistance System
ADF	Automated Driving Feature
ADS	Automated Driving System
ADS-CP	Automated Driving System Computing Platform
AI	Artificial Intelligence
API	Application Programming Interface
ARU	Analytic Redundant Unit
ASIL	Automotive Safety Integrity Level
AUTOSAR	AUTomotive Open System ARchitecture
BSW	Basic Software
CAN	Controller Area Network
COM	Communication
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DDT	Dynamic Driving Task
E2E	End-to-End
ECU	Electronic Control Unit
E/E	Electrical and/or Electronic
FCR	Fault Containment Region
FDTI	Fault Detection Time Interval
FIT	Failure In Time
FMEA	Failure Mode and Effects Analysis
FO	Fail-Operational
FOTI	Fail-Over Time Interval

FRTI	Fault Reaction Time Interval
FSC	Functional Safety Concept
FT	Fault-Tolerance
FTU	Fault-Tolerant Unit
GPS	Global Positioning System
gPTP	Generalized Precision Time Protocol
HARA	Hazard Analysis and Risk Assessment
HAZOP	Hazard and Operability Analysis
HD	High-Definition
HW	Hardware
IP	Internet Protocol
IMU	Inertial Measurement Unit
LIDAR	Light Detection and Ranging
MCU	Microcontroller Unit
MRC	Minimal Risk Condition
MTTF	Mean Time to Failure
NGU	Never-Give-Up
ODD	Operational Design Domain
OEDR	Object and Event Detection and Response
OEM	Original Equipment Manufacturer
OS	Operating System
PC	Personal Computer
PH	Performance Host
PMIC	Power Management Integrated Circuit
POSIX	Portable Operating System Interface
QoS	Quality of Service
RADAR	Radio Detection and Ranging
RT	Real-Time
RTOS	Real Time Operating System
Rx	Receive
SAD	System Architectural Design
SAE	Society of Automobile Engineers
SES	Safely Embedded Software
SH	Safety Host

SW	Software
SWC	Software Component
TCP	Transmission Control Protocol
TMR	Triple Modular Redundancy
TSC	Technical Safety Concept
TTA	Time-Triggered Architecture
Tx	Transmit
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
V2X	Vehicle-to-Everything
VCU	Vehicle Control Unit
WCET	Worst Case Execution Time

Bibliography

[1] T. Schmid, "Safety Analysis for highly automated driving," in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2018, pp. 154–157.

[2] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[3] A. Mehmed, M. Antlanger, and W. Steiner, "The Monitor as Key Architecture Element for Safe Self-Driving Cars," in *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, 2020, pp. 9–12.

[4] H. Kopetz, "On the Fault Hypothesis for a Safety-Critical Real-Time System," in *Automotive Software - Connected Services in Mobile Networks*, M. Broy, I. H. Krüger, and M. Meisinger, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 31–42, ISBN: 978-3-540-37678-1.

[5] R. Isermann, R. Schwarz, and S. Stolzl, "Fault-Tolerant Drive-by-Wire Systems," *IEEE Control Systems Magazine*, vol. 22, no. 5, pp. 64–81, 2002.

[6] SAE J3016, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, Norm, 2018.

[7] ISO/PAS 21448:2019, *Road vehicles - Safety of the intended functionality*, Norm, 2019.

[8] ISO 26262, *Road vehicles - Functional safety*, Norm, 2018.

[9] ISO/SAE DIS 21434, *Road vehicles - Cybersecurity engineering*, Norm.

[10] ISO/CD TR 4804, *Road vehicles - safety and cybersecurity for automated driving systems - design, verification and validation methods*, Norm.

[11] SAE International, [Online]. Available: `https://www.sae.org/` (visited on 04/13/2020).

[12] IEC 61508:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems*, Norm, 2010.

[13] T. Stolte, G. Bagschik, A. Reschka, and M. Maurer, "Hazard Analysis and Risk Assessment for an Automated Unmanned Protective Vehicle," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1848–1855.

[14] R. E. Lyons and W. Vanderkulk, "The Use of Triple-Modular Redundancy to Improve Computer Reliability," *IBM Journal of Research and Development*, vol. 6, no. 2, pp. 200–209, 1962.

[15] J. Braun and J. Mottok, "Fail-Safe and Fail-Operational Systems safeguarded with Coded Processing," in *Eurocon 2013*, 2013, pp. 1878–1885.

[16] Y. C. Yeh, "Triple-Triple Redundant 777 Primary Flight Computer," in *1996 IEEE Aerospace Applications Conference. Proceedings*, 1996, 293–307 vol.1.

[17] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangiovanni-Vincentelli, M. Peri, and S. Pezzini, "Fault-Tolerant Platforms for Automotive Safety-Critical Applications," Jan. 2003, pp. 170–177.

[18] A. Kohn, M. Käßmeyer, R. Schneider, A. Roger, C. Stellwag, and A. Herkersdorf, "Fail-Operational in Safety-Related Automotive Multi-Core Systems," in *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2015.

[19] L. Sha, "Dependable System Upgrade," in *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279)*, 1998, pp. 440–448.

[20] T. L. Crenshaw, E. Gunter, C. L. Robinson, L. Sha, and P. R. Kumar, "The Simplex Reference Model: Limiting Fault-Propagation due to Unreliable Components in Cyber-Physical System Architectures," in *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, 2007, pp. 400–412.

[21] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha, "The System-Level Simplex Architecture for Improved Real-Time Embedded System Safety," in *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2009, pp. 99–107.

[22] F. Abdi, R. Mancuso, R. Tabish, and M. Caccamo, "Restart-Based Fault-Tolerance: System Design and Schedulability Analysis," in *2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2017, pp. 1–10.

[23] F. Abdi, R. Tabish, M. Rungger, M. Zamani, and M. Caccamo, "Application and System-Level Software Fault Tolerance Through Full System Restarts," in *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*, 2017, pp. 197–206.

[24] T. Ishigooka, S. Honda, and H. Takada, "Cost-Effective Redundancy Approach for Fail-Operational Autonomous Driving System," in *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*, 2018, pp. 107–115.

[25] A. Ruiz, G. Juez, P. Schleiss, and G. Weiss, "A safe generic adaptation mechanism for smart cars," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, 2015, pp. 161–171.

[26] P. Schleiss, C. Drabek, G. Weiss, and B. Bauer, "Generic Management of Availability in Fail-Operational Automotive Systems," in *Computer Safety, Reliability, and Security*, S. Tonetta, E. Schoitsch, and F. Bitsch, Eds., Cham: Springer International Publishing, 2017, pp. 179–194, ISBN: 978-3-319-66266-4.

[27] J. Wolf, "Is This What the Future Will Look Like? Implementing fault tolerant system architectures with AUTOSAR basic software," *Elektronik automotive*, Nov. 2015.

[28] C. Temple and A. Vilela. (2014). Fehlertolerante Systeme im Fahrzeug - von fail-safe zu fail-operational, [Online]. Available: `https://www.elektroniknet.de/elektronik-automotive/assistenzsysteme/fehlertolerante-systeme-im-fahrzeug-von-fail-safe-zu-fail-operational-110612.html`
(visited on 07/11/2020).

[29] J. Meunier. Automotive Functional Safety: The Evolution of Fail Safe to Fail Operational Architecture, [Online]. Available: `https://blog.nxp.com/automotive/automotive-functional-safety-the-evolution-of-fail-safe-to-fail-operational-architecture`
(visited on 04/21/2020).

[30] C. Helpa. (2017). Fail-Operational-Plattform für vollautomatisiertes Fahren, [Online]. Available: `https://www.all-electronics.de/325813-2-fail-operational-vollautomatisiertes-fahren-plattform-ecu/`
(visited on 07/12/2020).

[31] M. Ghadhab, M. Kuntz, D. Kuvaiskii, and C. Fetzer, "A Controller Safety Concept Based on Software-Implemented Fault Tolerance for Fail-Operational Automotive Applications," in *Formal Techniques for Safety-Critical Systems*, C. Artho and P. C. Ölveczky, Eds., Cham: Springer International Publishing, 2016, pp. 189–205, ISBN: 978-3-319-29510-7.

[32] N. Druml, G. Macher, M. Stolz, E. Armengaud, D. Watzenig, C. Steger, T. Herndl, A. Eckel, A. Ryabokon, A. Hoess, S. Kumar, G. Dimitrakopoulos, and H. Roedig, "PRYSTINE - PRogrammable sYSTems for INtelligence in AutomobilEs," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 618–626.

[33] 32-bit AURIX Microcontroller based on TriCore, [Online]. Available: `https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/`
(visited on 07/12/2020).

[34] A. Schnellbach, M. Hirz, and J. Fabian, "Comparison of fail-operational software architectures from the viewpoint of an automotive application," *e & i Elektrotechnik und Informationstechnik*, pp. 283–293, Sep. 2016.

[35] G. Bosilca, A. Bouteiller, A. Guermouche, T. Herault, Y. Robert, P. Sens, and J. Dongarra, "A failure detector for HPC platforms," *The International Journal of High Performance Computing Applications*, vol. 32, pp. 139–158, Jan. 2018.

[36] X. Defago, N. Hayashibara, and T. Katayama, "On the Design of a Failure Detection Service for Large-Scale Distributed Systems," Apr. 2004.

[37] M. Bertier, O. Marin, and P. Sens, "Implementation and performance evaluation of an adaptable failure detector," in *Proceedings International Conference on Dependable Systems and Networks*, 2002, pp. 354–363.

[38] F. A. Arshad, G. Khanna, S. Bagchi, and I. Laguna, "Stateful Detection in High Throughput Distributed Systems," in *IEEE Symposium on Reliable Distributed Systems*, IEEE Computer Society, Oct. 2007, pp. 275–287.

[39] P. Croll and P. Griffiths, "Modelling Real-Time Behaviour of Parallel and Distributed Systems Under Failure Conditions," *IFAC Proceedings Volumes*, vol. 28, no. 5, pp. 543–550, 1995.

[40] A. S. M. Noor and M. M. Deris, "Extended Heartbeat Mechanism for Fault Detection Service Methodology," in *FGIT-GDC*, 2009.

[41] J. Na, D. Lee, M. Zorigbold, D. Lee, and S. Moon, "Simple and Low-Cost Heartbeat-Based Dual Modular Redundant Systems for Wireless Sensor Networks," in *Advances in Computer Science and Ubiquitous Computing*, J. J. Park, V. Loia, G. Yi, and Y. Sung, Eds., Singapore: Springer Singapore, 2018, pp. 546–552, ISBN: 978-981-10-7605-3.

[42] A. Mehmed, W. Steiner, M. Antlanger, and S. Punnekkat, "System Architecture and Application-Specific Verification Method for Fault-Tolerant Automated Driving Systems," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 39–44.

[43] Apollo Auto, [Online]. Available: `http://www.apollo.auto/index.html` (visited on 08/31/2020).

[44] Nvidia Drive, [Online]. Available: `https://developer.nvidia.com/drive` (visited on 08/31/2020).

[45] AUTOSAR (AUTomotive Open System ARchitecture), [Online]. Available: `https://www.autosar.org/` (visited on 06/21/2020).

[46] H. Kopetz and G. Bauer, "The Time-Triggered Architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[47] MotionWise, [Online]. Available: `https://www.tttech-auto.com/products/automated-driving/motionwise/` (visited on 07/22/2020).

[48] ADAS ECU - RazorMotion, [Online]. Available: `https://www.tttech-auto.com/products/automated-driving/razormotion-tttech-auto/` (visited on 07/22/2020).

[49] AMAZON Autonomous Driving, [Online]. Available: `https://aws.amazon.com/automotive/autonomous-driving/` (visited on 08/31/2020).

[50] AutonomouStuff, [Online]. Available: `https://autonomoustuff.com/` (visited on 08/31/2020).

[51] H. Martin, K. Tschabuschnig, O. Bridal, and D. Watzenig, "Functional Safety of Automated Driving Systems: Does ISO 26262 Meet the Challenges?" In. Sep. 2017, pp. 387–416, ISBN: 978-3-319-31893-6.

[52] M. Wood, P. Robbel, M. Maass, R. D. Tebbens, and M. Meijs, *Safety first for Automated Driving*, 2019.

[53] R. Obermaisser and P. Peti, "A Fault Hypothesis for Integrated Architectures," in *2006 International Workshop on Intelligent Solutions in Embedded Systems*, 2006, pp. 1–18.

[54] G. Bauer, H. Kopetz, and W. Steiner, "The Central Guardian Approach to Enforce Fault Isolation in the Time-Triggered Architecture," in *The Sixth International Symposium on Autonomous Decentralized Systems, 2003. ISADS 2003.*, 2003, pp. 37–44.

[55] QNX Operating System, [Online]. Available: `https://blackberry.qnx.com/en/products/neutrino-rtos/index`
(visited on 06/12/2020).

[56] Vector Microsar Operating System, [Online]. Available: `https://www.vector.com/int/en/products/products-a-z/embedded-components/microsar/`
(visited on 06/21/2020).

[57] J. M. Lozano Domínguez, T. d. J. Mateo Sanguino, and M. J. Redondo González, "Evaluation of a Robust Fault-Tolerant Mechanism for Resilient IoT Infrastructures," in *Broadband Communications, Networks, and Systems*, V. Sucasas, G. Mantas, and S. Althunibat, Eds., Cham: Springer International Publishing, 2019, pp. 3–12, ISBN: 978-3-030-05195-2.

[58] PRISM - Model Checker, [Online]. Available: `https://www.prismmodelchecker.org/`
(visited on 09/15/2020).

[59] Yocto Project, [Online]. Available: `https://www.yoctoproject.org/`
(visited on 07/26/2020).

[60] Wireshark, [Online]. Available: `https://www.wireshark.org/`
(visited on 07/22/2020).

[61] RFC1122, [Online]. Available: `https://tools.ietf.org/html/rfc1122`
(visited on 08/18/2020).

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Vienna, January 2021

Rupert Schorn