Project acronym:	*SafeAdapt*
Project title:	*Safe Adaptive Software for Fully Electric Vehicles*
Grant Agreement number:	608945
Coordinator:	*Dr.-Ing. Dirk Eilers*
Funding Scheme:	*FP7-2013-ICT-GC*

Deliverable 3.3

Specification of ISO 26262 Safety Goals for Self-Adaptation Scenarios

Due date of deliverable:	June 30th, 2015
Actual submission Date:	July 2nd, 2015
Lead beneficiary for this deliverable:	TECNALIA

Dissemination level		
PU	**Public**	X
PP	**Restricted to other programme participants (including the Commission Services)**	
RE	**Restricted to a group specified by the consortium (including the Commission Services)**	
CO	**Confidential, only for members of the consortium (including the Commission Services)**	

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013)

Document Information	
Title	D3.3. Specification of ISO 26262 safety goals for self-adaptation scenarios.
Creator	TECNALIA
Description	Definition of safety goals for the safe adaptation scenarios based on ISO 26262, building the argumentation for retaining of functional safety of a function during the adaptation of the system. It outlines what kind of argumentation is required to provide evidence that the safety goals are fulfilled. It also includes a proposal on changes to the ISO 26262 standard to deal with adaptation issues.
Publisher	TECNALIA
Contributors	Tecnalia: Alejandra Ruiz, Mª Carmen Palacios, Garazi Juez, Maite Álvarez Delphi: Timo Feismann, Martin Lange, Thorsten Rosenthal Duracar: Ken Lam Siemens: Cornel Klein, Jan Sawallisch, Michael Armbruster TTTech: Andreas Eckel *Advisors*: Gereon Weiß, Philipp Schleiß *Revision:* Martin Lange, Jan Sawallisch, Andreas Eckel
Language	en-GB
Creation date	20/04/2014
Version number	01
Version date	30/06/2015
Audience	☐ internal X public ☐ restricted

Table of Contents

List of Figures

List of Tables

Executive Summary

Today´s electronic control systems are prone to new types of failures that do not have effect on purely mechanical and hydraulic systems. More precisely, computation hardware typically fails more frequently and less gracefully than their mechanical counterparts, as for instance, evolving and eminent failures are not perceivable through degradation in quality, but rather occur abruptly. To establish an effective protection against these risks, safety-critical electronic control systems do not only require duplicate installation of identical components, but in fact necessitate sophisticated forms of redundancy and isolation techniques to avoid systematic errors spreading throughout the system. Furthermore, safety measures are defined to avoid or control systematic failures and to detect or control random hardware failures, or mitigate their harmful effects. In addition, dependent failures are addressed by means of physical separation or design diversity. Adaptation poses as a viable solution to increase the reliability of safety-critical applications without relying on mechanical fall-back solutions. Obviously, safety issues and certification demands cannot be neglected when adaptation and dynamic function reallocation are to be applied to future Fully Electric Vehicles (FEVs.)

The static nature of best practice automotive architectures is also reflected by the ISO26262 standard, which provides limited guidance on leveraging adaptive behaviour with the goal to improve safety. Motivated by this non-exploited possibility to improve safety through dynamic and adaptive architectures, this paper sets out to identify and discuss future safety concerns and their relation to current certification practise.

Since the introduction of ISO the 26262 standard in 2011, functional safety assessment in the automotive domain is governed by an industry specific guideline, thus replacing the general IEC 61508 standard. Regarded more closely, adherence to the ISO 26262 standard with respect to the development process for FEV is a challenging assignment, as for instance, no previous experience in defining runtime hazards exists, which in turn is, however, indispensable for defining sound countermeasures. Adaptive systems could be defined as those that are able to modify its behaviour at operating time depending on already defining conditions that trigger this change. The new state of the system after the adaptation could include new hazards also called run time hazards, which are not avoided, detected or sufficiently mitigated.

This document aims at answering to two different questions. Firstly, the challenges of applying ISO 26262 to the novel architecture are identified and new interpretations and means for compliance to the standard are proposed. Secondly, it is explained how the developed adaptation concept of the SafeAdapt project identifies and mitigates the effects of possible adaptation hazards.

1 Introduction

The promising advent of fully electric vehicles also means a shift towards fully electrical control of the vehicle functions. In particular, critical X-by-Wire functions require solutions with sophisticated redundancy. As a result, the overall Electric/Electronic (E/E) architecture of a vehicle is becoming more overly complex and costly.

The main concept behind SafeAdapt (Safe Adaptive Software for Fully Electric Vehicles) is to develop novel E/E architecture based on adaptation to achieve safety, reliability, and cost-efficiency in future FEVs. Thereby, the system complexity will be reduced due to generic and system-wide fault handling and change management mechanisms. Furthermore, and despite failures, the reliability is extended, active safety is improved, and resource utilisation is optimised. This is especially important for increasing reliability and efficiency regarding energy consumption, cost, and design simplicity.

SafeAdapt follows a holistic approach for building adaptive systems in safety-critical environments by comprising methods and tools. The technical approach builds on a Safe Adaptation Platform Core (SAPC), encapsulating the basic adaptation mechanisms for re-allocating and updating functionalities in distributed automotive control systems. Although SafeAdapt does not endeavour to certify developed software or hardware, the certification process exceeds the scope of the project. However, the SafeAdapt approach considers functional safety with respect to the ISO 26262 standard to support potential certification issues of safety-critical e-vehicle systems.

SafeAdapt provides an integrated approach for engineering such adaptive, complex and safe systems, ranging from tool chain support, reference architectures, modelling of system design and networking, up to early validation and verification. For realistic validation of the adaptation and redundancy concepts, an actual vehicle prototype with different and partially redundant applications is developed.

1.1 Document Scope

The purpose of this document is to specify the safety goals for the SafeAdapt solution, which is an architecture that is capable of changing dynamically in response to different situations that can be reached when some vehicle functions are not working properly or on decisions triggered by energy levels that can end up in dangerous situations.

We also outline what kind of argumentation is required to provide evidence that these safety goals are fulfilled.

Moreover, this deliverable includes a “quasi guideline” explaining which part of the certification need to be newly treated / implemented and which can be omitted due to the novel approach.

1.2 Document Outline

The remainder of the report is structured as follows:

- Section 2: Describes the outputs of the hazards analysis and risk assessment done at vehicle level of the functions selected to be included on the SafeAdapt demonstrators.

- Section 3: Describes the challenges identified in order to apply the ISO 26262 functional safety standard to the self-adaptive systems such as SafeAdapt is proposing.
- Section 4: Includes a brief description of the functionality of the Safe Adaptation Platform Core.
- Section 5: Describes the process followed to identify the safety goals derived from the adaptation hazards and how they are avoided or mitigated by applying the safety concept design.
- Section 6: Includes proposals to ISO 26262 in order to maintain the same objectives but applying them to the self-adaptive systems.

2 Outputs of the vehicle hazard analysis

On the deliverable D2.1 an excerpt of the Hazard Analysis and Risk Assessment work was included which was part of the analysis done to define the project use cases.

An abstract of that analysis defining the vehicle functions, hazards effect and ASIL assigned in **Table 1**.

Function	**Hazard**		**Potential Effect**	**ASIL**
	ID	**Description**		
ACC	ACC1	The driver may lose control of the vehicle if sudden unintended acceleration occurs. Departure of the lane may lead to accidents with road users/objects.	ACC-acceleration > 2 m/s^2	**B**
ACC	ACC2	Life-threatening injuries, possible rear-end collision with vehicle in front.	Without driver intervention the vehicle would reach max. speed	**B**
ACC	ACC3	Possible rear-end collision with vehicle in front.	It takes too long to reach the target speed	**QM**
ACC	ACC4	Another car is following the ACC-vehicle. Rear end collision with the ACC-vehicle.	Worst Case: unintended deceleration until standstill due to brake intervention	**C**
ACC	ACC5	Oncoming traffic or obstacles on the side, driving in a curve with max. speed, vehicle close to destabilisation. Destabilisation of the vehicle, departure of the lane, collision with objects.	Unintended to strong deceleration of the vehicle	**B**
ACC	ACC6	Vehicle approaching pedestrian crossing with extreme low speed, driver not pressing brake- or clutch pedal.	Unintended acceleration	**A**
ACC	ACC7	Driving backwards, persons close behind the vehicle.	Unintended acceleration, starting from standstill	**A**
ACC	ACC8	City traffic, traffic jam, pedestrian forces his way on the pedestrian crossing between the ACC vehicle and the car in front.	Unintended driveway of the vehicle	**A**
AEB	AEB1	City traffic, pedestrian in front of the car crossing the road, daytime, and car does not brake automatically. Possible collision.	Vehicle will not brake automatically	**A**
AEB	AEB2	Destabilisation of the vehicle, departure of the lane, collision with objects.	Worst Case: unintended deceleration until standstill due to brake intervention	**C**
SA	SA1	The driver may lose control of the vehicle if follows sleep.	Worst Case: The driver is not alerted and fall sleep	**A**
SA	SA2	The driver may lose control of the vehicle if alert is too loud.	Driver is annoyed by continuous alert	**QM**
BMS	BMS1	Driver and passengers can be damaged by a fire or explosion.	Overcharging of Battery- degassing fire, explosion (depending on cell chemistry)	**D**
BMS	BMS2	Drivers, passengers, pedestrian can be damaged by a fire or explosion..	Overcharging of Battery- degassing fire, explosion (depending on cell chemistry)	**C**

BMS	BMS3	Drivers, passengers, pedestrian can be damaged by a fire or explosion.	Overcharging of Battery- degassing fire, explosion(depending on cell chemistry), local heating due to overcurrent	**C**
MIW	MIW1	The driver may (partial) lose vehicle control.		**C**
SBW	SBW1	The driver may (partial) lose vehicle control.		**D**
BBW	BBW1	The driver may (partial) lose vehicle control.		**D**
BBW	BBW3	The driver will lose vehicle control.	Vehicle can't stop	**C**
Non-Driving function	NDF	Passenger and the person charging the battery could be injured.	Overheating of battery-> degassing, fire, explosion(depending on cell chemistry), aging	**C**

Table 1 Vehicle hazard analysis

For each of the applications different particularities appear but for all of them, the intention for the safety goal definition is to avoid malfunction within the vehicle. As a result of this hazard analysis the following common safety state has been identified:

In case of failure detection, adaptation is triggered.

It has been decided to follow a common strategy when a failure is detected which takes advantage of the adaptation capabilities so as to define a new configuration where the failure does no impact on the vehicle behaviour.

3 Challenges for Compliance with ISO 26262

ISO 26262 [ISO26262 2011] is a quite novel standard that appears on November 2011, which can be described as an objective standard. It does include quite innovative concepts from the assurance point of view such as model based engineering or the application of formal methods for validation and verification. Nevertheless, it does not take into account adaptive systems or autonomously operating.

In order to detect which are the parts of the standard which are more challenging to be fulfilled on adaptive systems various brainstorming meetings have been conducted in the context of SafeAdapt project where different points of ISO 26262 were analysed in order to apply them to a specific adaptive architecture. As a result of the brainstorming the following main areas have been identified where the application might result difficult:

- Adaptation as a safety mechanism or as functionality
- Hazard Analysis and Risk Assessment (HARA)
- Automotive Safety Integrity Level (ASIL) decomposition
- Safety Analysis
- HW-SW Interface
- Requirement Management
- Verification and Validation (V&V)
- Maintenance
- Safety Element out of Context (SEooC)

After defining this list, we have produced a survey between different practitioners (24 replies) either experts on adaptive systems or experts on safety and standards compliance or both. The objective of this survey was to validate and enhance the first findings.

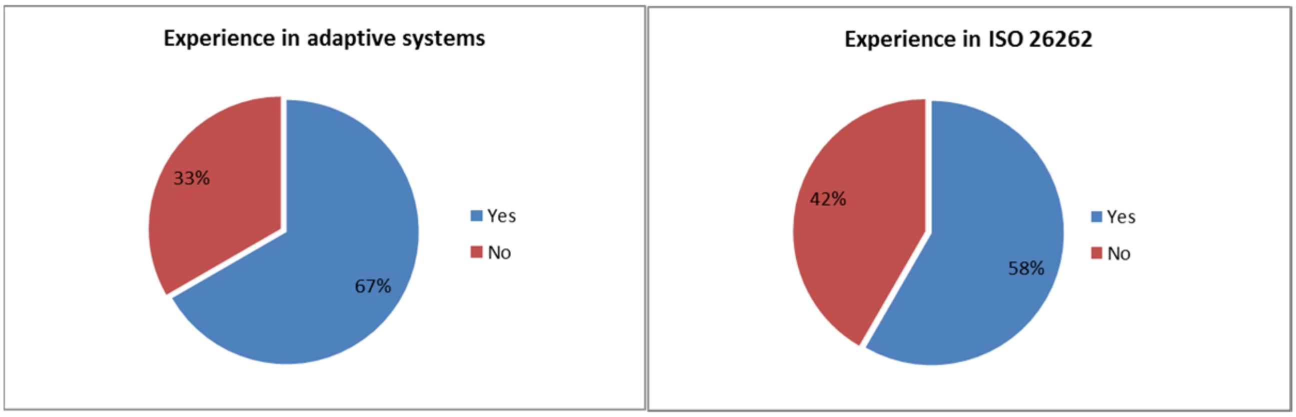


Fig. 1 Statistics of experiences on adaptive systems and ISO 26262

3.1 Adaptation as a safety mechanism or as functionality

This has been a very controversial point when analysing the feasibility of applying ISO 26262 to adaptive systems. The first challenge we came across is the definition of Adaptation. In [Whittle et al 2009] Whittle defines Self-adaptive systems as those which have the capability to autonomously modify their behaviour at run-time in response to changes in their environment. Whittle presents adaptation as a general feature that deals with the availability rather than safety. However, in

[Gudemann et al 2006], adaptation is presented in a way where the systems takes advantage to make the function of the system available again. Gudemann indicates that “if the hazard occurs and the reconfiguration succeeds, then the system will come back into working mode some time later. So the occurrence of the hazard is only ‘critical’ if the system cannot repair itself anymore. Intuitively the hazard is only critical, if the system cannot repair itself by reconfiguration. In other words: if the self-x capabilities may not compensate the failures any more”. In this case, the adaptation is seen as a system level safety mechanism to reach fail-operational behaviour.

In SafeAdapt project adaptation is seen with both approaches. On the one hand, SafeAdapt aims to develop novel architecture concepts based on adaptation so as to be able to handle failures in safety-relevant systems by adaptation/reconfiguration, especially failures where current systems do not degrade gracefully. On the other hand, the Safe Adapt Platform Core (SAPC) is also meant to optimize the energy consumption in automotive E/E architectures. This second purpose it is not directly linked with the avoidance of a hazard, it just deals with the modification of the behaviour so as to make it more efficiently.

In the conducted survey, similar results were obtained as it can be seen in **Fig. 2**: safety mechanism (prevent hazards by adaptation) or as a common functionality (increase the efficiency).

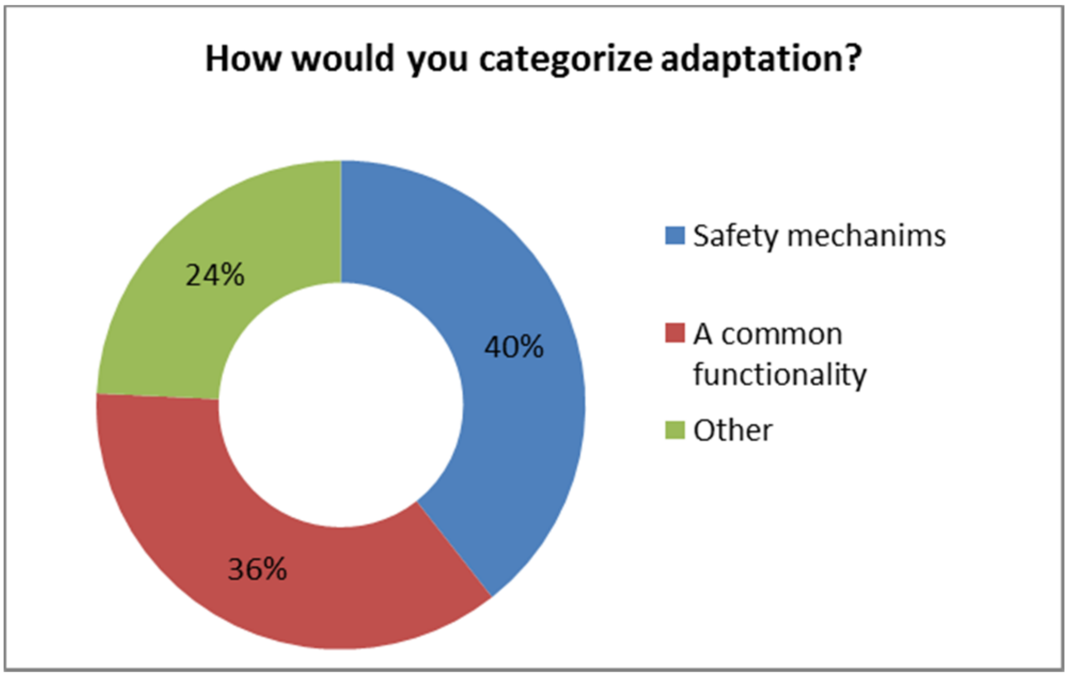


Fig. 2 Results for the question about adaptation category

When we think about how to apply ISO 26262, the main interest is the functional safety. In this context, adaptation can be used to ensure the overall safety as a system level safety mechanism so as to avoid possible hazards. This would apply for adaptation used as a common functionality when it is used to increase the efficiency of the vehicle’s energy consumption. However, if adaptation for efficiency reasons produces the introduction of new hazards, then ISO26262 compliance should be ensured.

3.2 Hazard Analysis and Risk Assessment (HARA)

According to ISO 26262 [ISO 26262-3:7.4.1.2] the item without internal safety mechanisms shall be evaluated during hazard analysis and risk assessment. This means that if a vehicle subsystem is considered, we should analyse it alone and detect the possible hazards it could create and thereafter define the adaptation as a safety mechanism put in order to avoid specific hazards.

Safety mechanisms of the item that are intended to be implemented or that have already been implemented are incorporated as part of the functional safety concept. Afterwards, safety mechanisms are refined, specified in detail and allocated to hardware and software at Technical Safety Requirements and System Design Level.

However, if adaptation is done systematically across the vehicle in order to avoid hazards derived from malfunctions then, adaptation could also lead to new hazards not considered on the previous hazard analysis.

During the survey 79% of the replies indicate the need of a change in the HARA so as to include adaptation on the analysis.

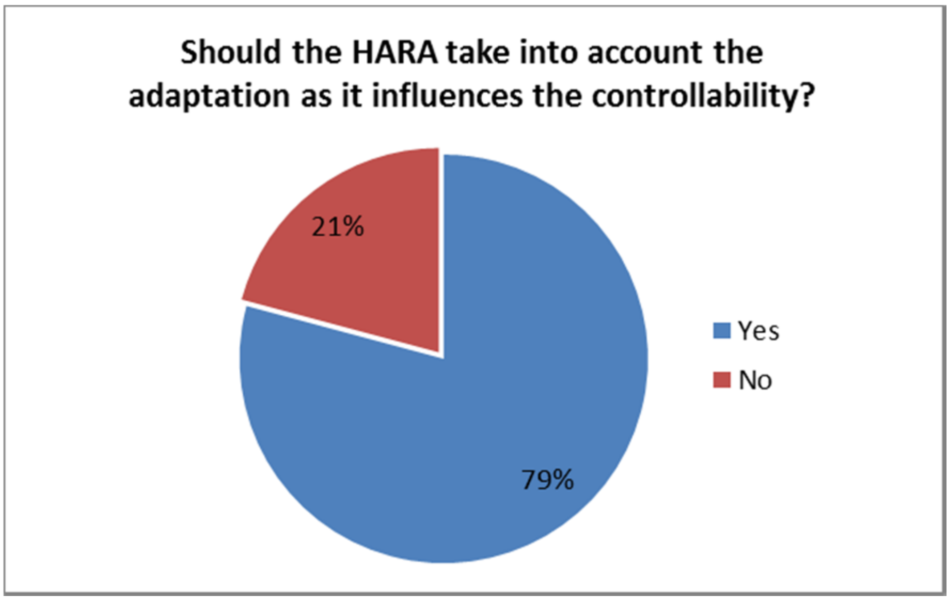


Fig. 3 Survey's answers to the question about HARA modification

In [Gudemann et al 2006] Gudemann says:" Standard methods for reliability analysis like FMEA, FTA and DCCA are not directly applicable, because they try to find only cause-consequence relationships between component failures and system failures."

3.3 ASIL decomposition

ASIL (Automotive safety Integrity Level) is defined as one of four levels to specify item's or element's necessary requirements of the ISO 26262 safety measures to apply for avoiding an unreasonable residual risk with D representing the most stringent and A the least stringent level [ISO26262 2011].

As a result of the survey about this issue, 83% of the respondents affirmed that ASIL decomposition should take adaptation into consideration.

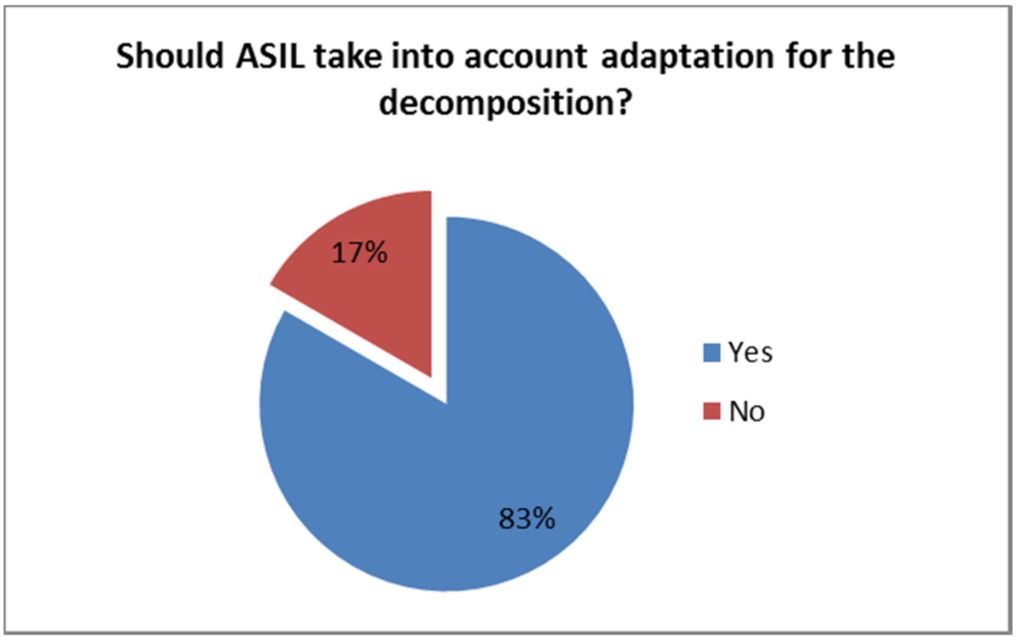


Fig. 4 Survey's answer about ASIL decomposition

Adaptation implies a harmonized mean for error handling. Failures of different system's elements are handled in a systematic and common way offering a better control of the failure effect. It decreases the options of the failure to produce a real hazard.

Related to ASIL decomposition we also asked in the survey about whether we should limit the applications to be adapted based on their criticality. 79% of the answers indicated that adaptation might be used to adapt ASIL D applications.

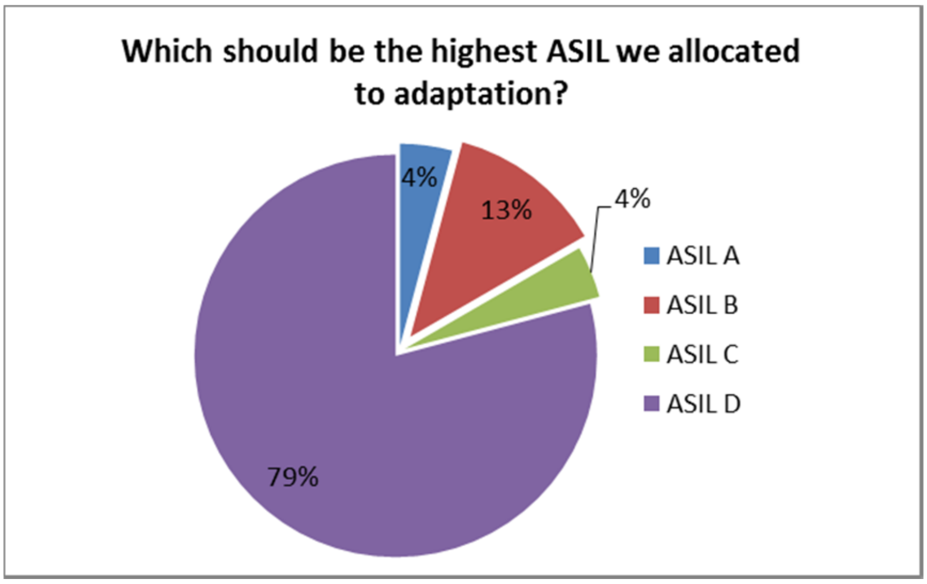


Fig. 5 Survey's answer about ASIL allocation to adaptation

The implication of introducing adaptation to all levels of critical applications is comparable to the multicore challenge running mixed critical applications. [Pop et al. 2013] indicates that when several safety functions of different criticality share resources such as running on the same multicore, the standards require hardware and software to be developed at the highest critical level among the levels of the safety functions implicated.

ASIL decomposition when dealing with adaptation should be treated in the same way as the mixed criticality issue. The first way to address the previous concern would be to assign the highest ASIL for all the functions affected by the adaptation. Otherwise, it should be ensured that no interference could occur between the functions impacted by the adaptation. Ensuring freedom of interference could be quite challenging, however, it is really a must to guarantee safety of the system.

3.4 Allocation to hardware and software

ISO 26262 includes the requirement of allocation of the functional safety requirements to preliminary elements of the architecture for the safety concept. This allocation should be enhanced once we get the technical safety requirement derived. This occurs when the allocation is done to hardware, software or both, although the allocation is difficult when we are considering adaptation. This process implies a modification of the behaviour so as to handle an error. This behaviour alteration implies that a function could run on different hardware or software elements.

Adaptation implies that a piece of software executing a function which runs on a specific hardware element due to an adaptation is migrated to another piece of hardware different from the previous one. We can even have other cases where it is not the hardware which changes but the software, the implementation or have a degraded behaviour.

Should ISO 26262 requirements be modified so the allocation is done for a normal mode and also all the possible allocations it could have once it is adapted? That could only be done if the adaptation is programmed or prearranged meaning that the functions could only be adapted to already specified configuration and not others. This solution would leave all those systems outside the standards which provide adaptation in a dynamic way.

This makes a real challenge to come up with a clear outcome on how this issue should be overcome in SafeAdapt. Considering preconfigured adaptation scenarios, all the possible allocation possibilities could be fixed for a specific case. On the contrary, since there is no a clear and consistent conclusion on how this should be treated when adaptation is carried out in a really dynamic way, we consider that much research should be performed in this area.

3.5 Hardware-Software Interface (HSI)

In terms of ISO 26262, the HSI “shall include the component’s hardware devices controlled by software and hardware sources that support the execution of software”. The information for the HSI should include characteristics such as shared and exclusive use of hardware resources, the access mechanisms to hardware devices or timing constraints for each service.

With adaptation, the hardware and or software should be capable to trigger the adaptation and dynamically execute different applications with different level of criticality and degradation. It is not only needed to specify how software controls the hardware, but also the adaptation implications what implies a more complex specification. The access mechanism to hardware devices will imply also the adaptation capability. Timing constrains should also consider the adaptation timing needs.

HSI as it is defined in ISO 26262 does not include any reference to the need to include the specification of the adaptation interface. ISO 26262-4:2011 7.4.6.3 clause indicates that relevant diagnostic capabilities such as diagnostic features concerning hardware to be implemented in software shall be included in the HSI specification. This might be interpreted as an adaptation triggering feature. In spite of the presence of some example contents in Annex B of Part 4, it does not reference to any adaptive behaviour.

3.6 Safety Analysis

As stated in ISO 26262, safety analyses such as system, design and process Failure Mode and Effects Analysis (FMEA), ETA or Markov modelling (inductive analysis methods) and Fault Tree Analysis (FTA) or Reliability Block Diagrams (RBD) (deductive analysis) shall be accomplished. Among other objectives, by means of these methods both the validation of the safety goals and the verification of safety concepts and requirements are performed. Moreover, safety analyses are carried out at the appropriate level of abstraction during the concept and product development phases.

It should be noted that the aforementioned techniques are not directly applicable for adaptive systems. They try to find only cause-consequence relationships between component failures and system failures [Gudmann et al 2006]. When the question about this issue was raised, 92% of the replies indicated that safety analysis should be modified to include adaptation on their scope.

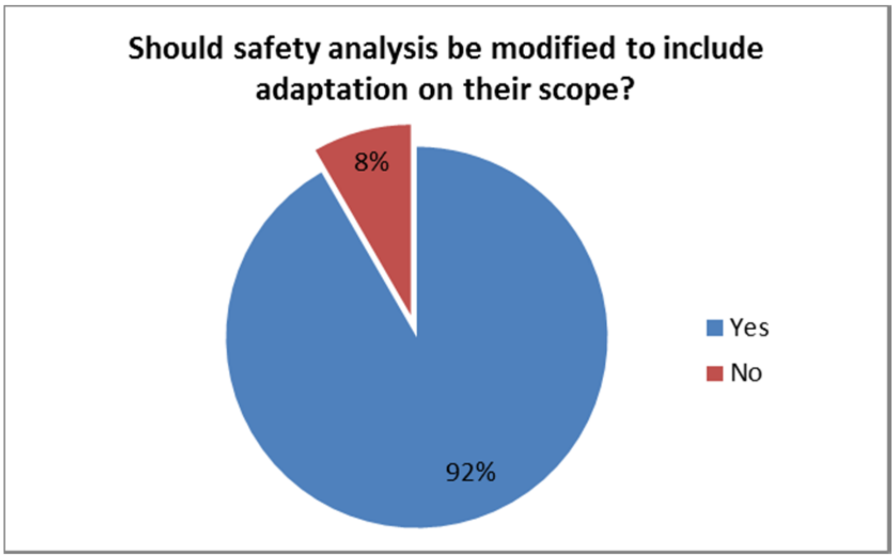


Fig. 6 Survey's answers about adaptation impacts on safety analysis

3.7 Adaptation requirement management

ISO 26262 requires tracing safety requirements from high level functional safety requirement into technical requirements and then to derive to software and hardware requirements.

When we questioned this requirement during the survey the answer is not clear. Only 46 % of the respondents answer if the adaptation requirements should be trace separately from the safety requirements.

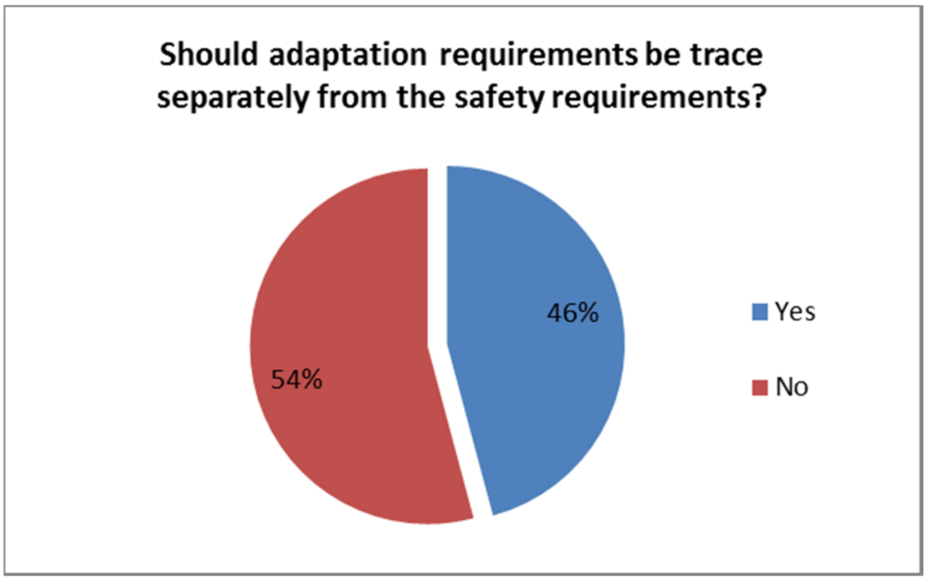


Fig. 7 Survey's answer about the adaptation requirements management

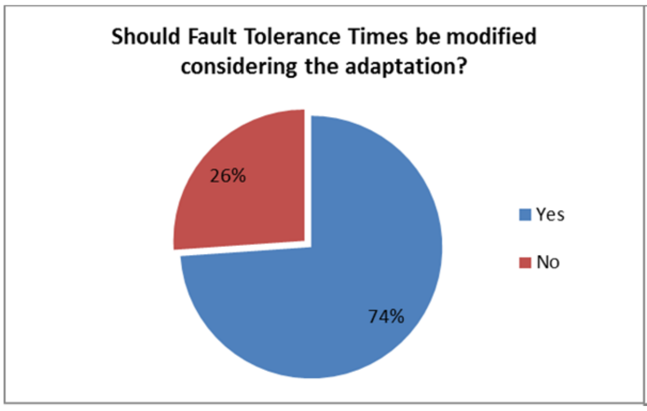
Such an unclear answer makes us think how we should manage the adaptation requirements needs some further analysis.

3.8 Adaptation Verification & Validation

In terms of ISO26262 different hardware metrics need to be achieved depending on the required ASIL level. Adaptation could significantly lead to a change of HW metrics considerations and they should be available for all the possible adaptation scenarios having to be parameterized to take in account those adaptation procedures. Furthermore, Diagnostic Coverage (DC) would be more complex to either calculate or verify as well. In other words, it could influence whether certain hardware parts contribute to Safety Goal violation and, consequently, the metrics could have a different value.

Several literature studies such as [Eghbal et al 2009] [Alexandersson and Karlsson 2011] refer to the previous concept stating how different workloads running on the same microprocessor at different times lead to different diagnostic coverage. This concept has been validated by means of fault injection dependability validation technique [Arlat et al 1993] [Mariani et al 2006]. This problematic could be addressed by the SEooC approach.

Other parameters such as Fault Tolerant Time Interval (FTTI) would also need to take adaptation into account. The verification of this parameter could also be a challenge since the different reconfiguration/adaptation scenarios shall be taken in account.



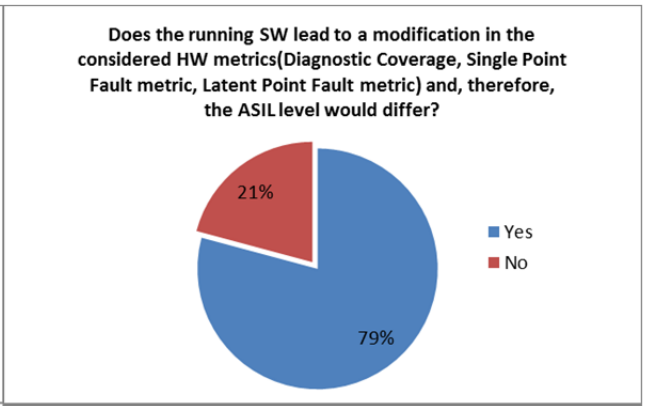


Fig. 8 Survey's answer about FTTI and hardware metrics

In the survey we asked about Fault Tolerant Time Interval measurement and 74% of the respondents answered that the measure needs some modifications as adaptation does impact on the obtained value. In fact, 79% of the survey answers say that adaptation on software implies a modification on the hardware metrics. We might need more complex methods in order to validate that the initial metrics obtained during the safety analyses evaluation have not changed.

3.9 Maintenance

ISO 26262 part 7 Clause 6 deals with operation, service (maintenance and repair) and decommissioning activities. Processes related to service shall be planned and evaluated. Systems maintenance should also include adaptation maintenance.

If adaptation is considered as mean for repairing or upgrading a function, then this clause should apply, even if it has to be mentioned that this clause was not written having adaptation functionality in mind.

Frtunikj [Frtunikj, 2014] proposes the use of plug and play mechanism as an adaptivity function: "Via Plug&Play an owner should be able to personalize his car, stay up to date by adding new hardware and software components, or upgrade old components with newer software." From the functional safety perspective the use of this mechanism could introduce undetected hazards as possible interference to other components. Responsibilities assignation could differ if this mechanism is applied.

3.10 Safety Element out of Context (SEooC) development

Part 10 of ISO 26262 includes guidelines for the Safety Element out of Context development. SEooC is a safety-related element which is not developed for a specific item.

Ruiz et al. [Ruiz et al 2013] mentioned the difficulties of SEooC development as the context in which the element is going to be integrated is unknown at development time. With the SEooC definition we can interpret that any application which adapts can be catalogued as a SEooC because the adaptation implies the lack of knowledge about in which hardware environment such application might be used or executed.

Part 10 of ISO 26262 mentions as examples of SEooC, AUTOSAR [Autosar] application software modules and AUTOSAR basic software modules. Adaptable applications can be designed as

AUTOSAR application software modules; consequently adaptable functions and SAPC software can be classified as software SEooC. The adaptation feature such as the default error handling mechanism can be developed following the specification of an AUTOSAR basic software module, thus it can be defined as a SEooC.

If adaptable functions and adaptation feature are considered as SEooC, then the assumption identification becomes a more challenging activity. Different contexts, in which functions could be executed, should be considered as assumptions. During design time, the different environments where the function is expected to be executed, running should be assumed. As these contexts are difficult to identify before operational time, assumptions are not specified as possible context in which the application will run but instead specify as environment constraints defining the minimum requirements in order to run the applications.

4 Safe Adaptation Platform Core (SAPC)

In SafeAdapt, adaptation will be achieved with the help of the Safe Adaptation Platform Core (SAPC). The SAPC is software that is hosted on different computing platforms (RACE and TMDP), which in turn are commonly referred to as core nodes. Each instance of the SAPC controls the adaptation for its local core node, but also guarantees to reach a correct system configuration with respect to all other instances of the SAPC. It is important that the SAPC follows safety standards and limits its impact on the effort for designing the resulting system. For this, each SAPC regularly sends information about the current state of its platform to the other SAPCs. When combining this volatile information with the predefined requirements of the vehicle, the SAPC is capable of calculating a configuration to address the newly occurred need for adaptation.

Fig. 9 shows the intended concept of the hardware architecture as far as it is relevant for the SAPC. The figure uses boxes for gateway nodes, round cornered boxes for core nodes, and circles for switches (see glossary). Gateway nodes provide access to aggregates, such as sensors and actuators, which are not designed to be directly connected to an Ethernet network. To ensure the required level of fault-tolerance, the power supply of the nodes has to be designed in a way that the loss of one power line still allows a safe operation. This is depicted through red and blue colored borders denoting the two separate power circuits used in the demonstrator car (see D5.1). The demonstrator core is based on results of the research project RACE [RACE] and as such uses multiple overlapping rings to connect the nodes in the system. In SafeAdapt, it is planned to replace the inner ring (core network) containing the main computing platforms (core nodes) of the RACE vehicle with a Time-Triggered Ethernet (TTE) based data communication. The wiring of the outer rings with the gateway nodes can, however, remain unchanged.

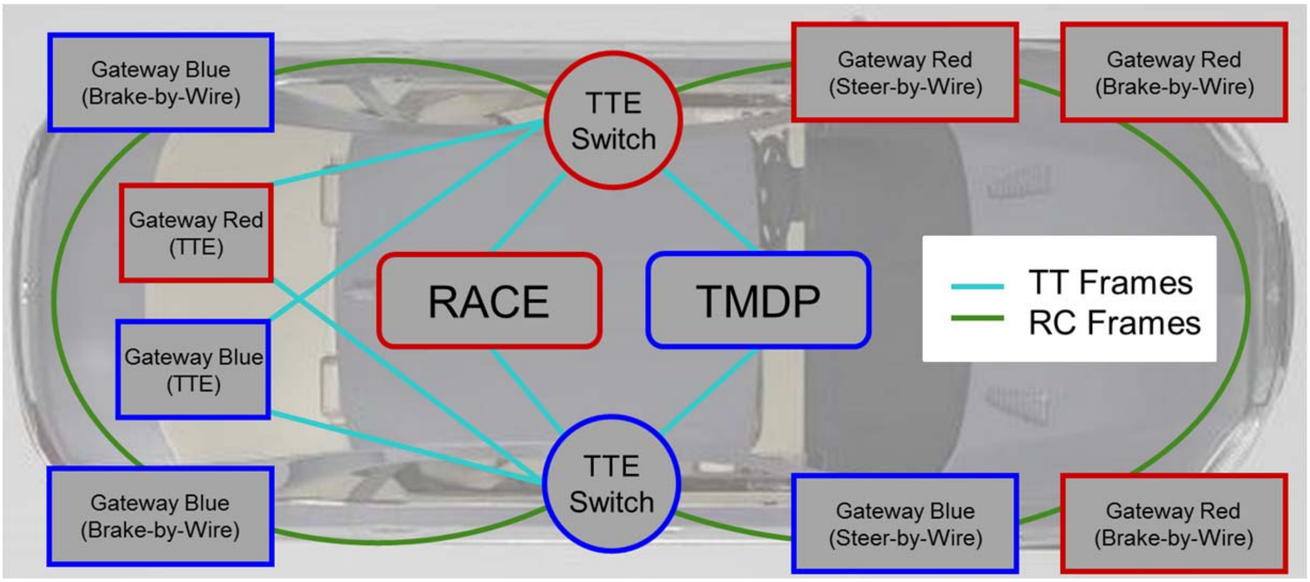

Gateway Blue (Brake-by-Wire)
TTE Switch
Gateway Red (Steer-by-Wire)
Gateway Red (Brake-by-Wire)
Gateway Red (TTE)
RACE
TMDP
TT Frames
RC Frames
Gateway Blue (TTE)
Gateway Blue (Brake-by-Wire)
TTE Switch
Gateway Blue (Steer-by-Wire)
Gateway Red (Brake-by-Wire)

Fig. 9 Overview of Hardware Architecture

The SAPC is part of the Information and Communication Technology of a vehicle (ICT) which can be described as follows:

The adaptation core is a logical item which consists of several control-computers, the so called core nodes (see **Fig. 9**). The adaptation core ensures the consistent and unique control of all actuators in any use-case of adaptation mentioned within this project. In the SafeAdapt implementation the adaptation core or CCC comprises a SIEMENS and a DELPHI platform also known as core node or CSCC as illustrated in **Fig. 10**.

The uniqueness of the control is ensured by the safe adaptation core SAPC. The SAPC has a local instance on each core node. This instance is a piece of software deployed onto each core node, thus on the SIEMENS and on the DELPHI platform.

Each instance of the SAPC consists out of a set of algorithms and interfaces to handle:

- detected faults,
- assign faults to fault containment regions (FCRs),
- rate the health-state of each FCR,
- exchange the health state of each FCR with all other instances of the SAPC and to provide a consistent data base on each core node. The database is readable only from the SAPC and it might be writeable by diagnostics,
- decide on the basis of the interactive consistent data base on each core node
  - which application to deploy,
  - which application to execute,
  - which application to run where as master and where as slave.

Acc. to [Butler 2008], fault containment region means:

> *"The primary goal of a FCR is to limit the effects of a fault and prevent the propagation of errors from one region of the system to another. A FCR is a subsystem that will operate correctly regardless of any arbitrary fault outside the region. FCRs must be physically separated, electrically isolated, and have independent power supplies. Physical dispersion limits the effect of physical phenomena such as the impact of a micrometeoroid. Electrical isolation protects against fault propagation from lightning or other forms of static discharge. Power supply isolation prevents a power failure affecting the entire system. The number of FCRs in a system is a primary factor in determining how many faults a system can tolerate without failure."*

In the context of SafeAdapt, various FCRs have been defined for each platform at different levels. It is important to insist on the need to prevent errors to propagate from one FCR to another to a certain degree. For further information, please refer to D3.2 (to be published).

The concept of interactive data-consistency is described in [Butler 2008] and [Armbruster 2009] at which the latter also explains the master/slave concept.

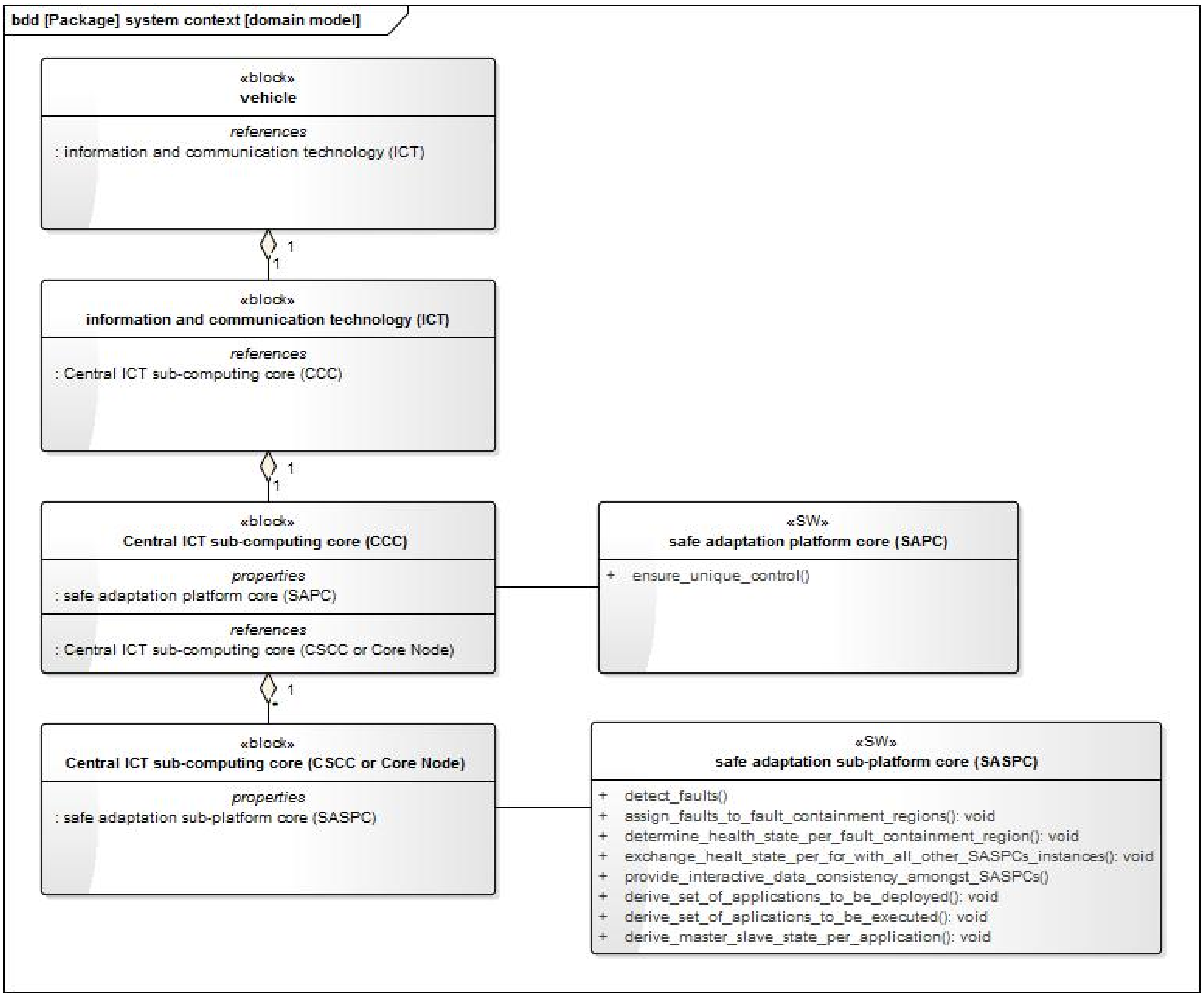

bdd [Package] system context [domain model]
«block»
vehicle
references
: information and communication technology (ICT)
1
1
«block»
information and communication technology (ICT)
references
: Central ICT sub-computing core (CCC)
1
1
«block»
Central ICT sub-computing core (CCC)
properties
: safe adaptation platform core (SAPC)
references
: Central ICT sub-computing core (CSCC or Core Node)
«SW»
safe adaptation platform core (SAPC)
+ ensure_unique_control()
1
*
«block»
Central ICT sub-computing core (CSCC or Core Node)
properties
: safe adaptation sub-platform core (SASPC)
«SW»
safe adaptation sub-platform core (SASPC)
+ detect_faults()
+ assign_faults_to_fault_containment_regions(): void
+ determine_health_state_per_fault_containment_region(): void
+ exchange_healt_state_per_fcr_with_all_other_SASPCs_instances(): void
+ provide_interactive_data_consistency_amongst_SASPCs()
+ derive_set_of_applications_to_be_deployed(): void
+ derive_set_of_aplications_to_be_executed(): void
+ derive_master_slave_state_per_application(): void

Fig. 10 Domain-model

5 Safety Assurance for the Safe Adaptation Platform Core

The SafeAdapt Platform Core as it is explained before has resulted to be an advanced and sophisticated mechanism in charge of the default error handling. One of the big concerns at concept and design time of the SAPC is the mitigation of possible hazards that could occur due to the adaptation.

This section includes a description of the methodology followed in order to determine the safety goals specific for the adaptation. This is based on an iterative process described in **Fig. 11**.

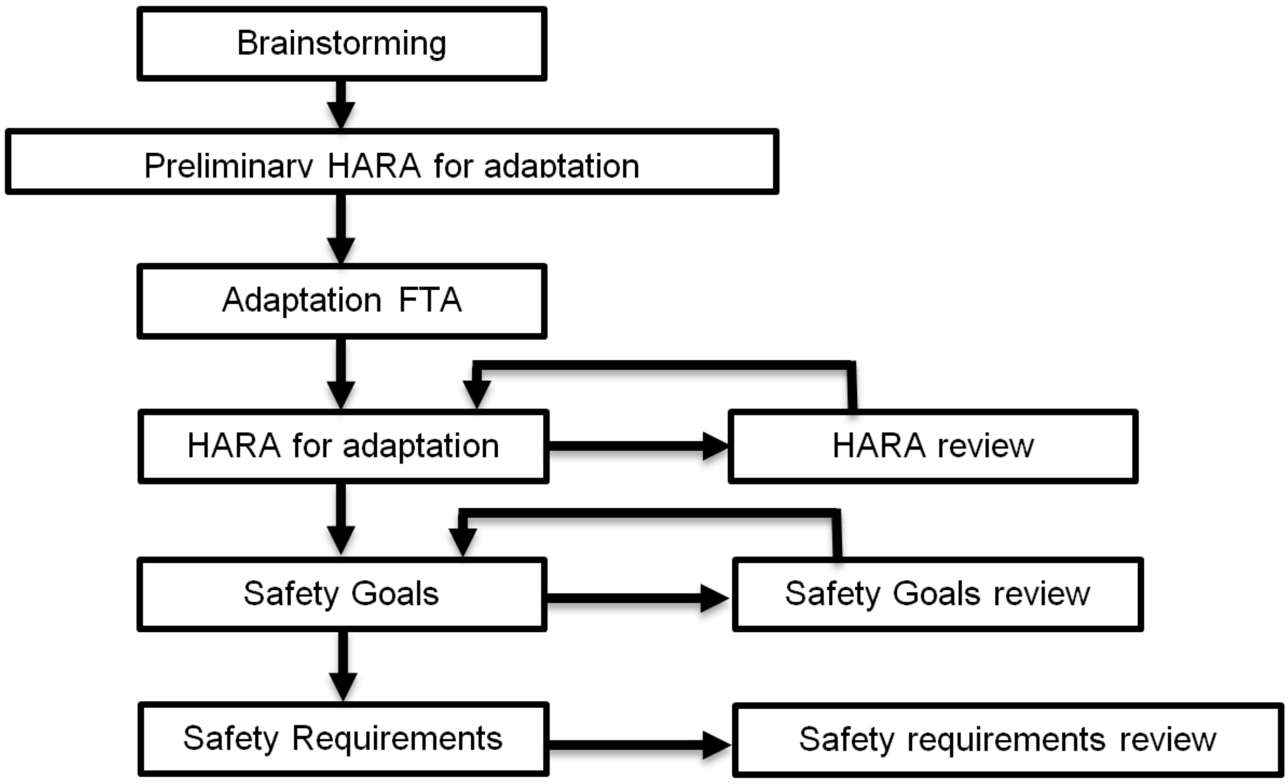


Fig. 11 Process followed for the safety goals definition

The activities executed along the process are the following:

1. Brainstorming session to identify possible error and faults on the adaptation
   Different partner experts were asked about possible failures and effects at vehicle level
2. As a result of the brainstorming session, we were able to define FTA for the adaptations functions
3. HARA for the malfunctions identified on previous phases
4. Review of the HARA from different partners
5. Safety goals definition
6. Review of the safety goals
7. Safety requirements specification
8. Review of the safety requirements by different partners

To reach the independency level required by ISO 26262, it has been decided to make the work by a group and to review the performed work by other partners.

5.1 Safe Adaptation Platform Core FTA

As a result of FTA activity, we came across with the following high-level FTAs. In this sub section, resulting FTA diagrams are included being the scope of this analysis the SafeAdapt core system. The illustrated fault trees have been defined at a preliminary stage of the design of the SAPC and will be part of the inputs required for an appropriate design.

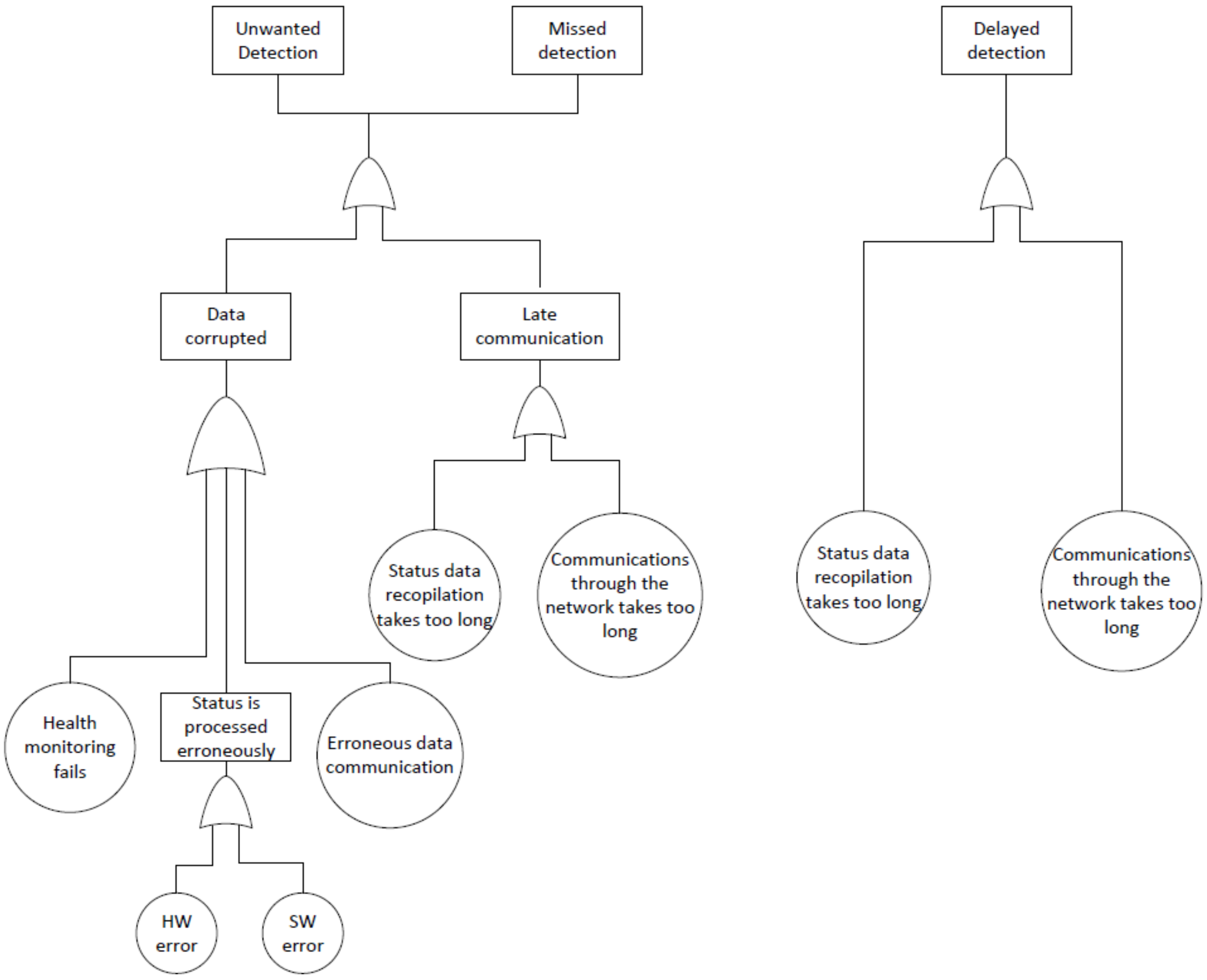

Unwanted Detection
Missed detection
Delayed detection
Data corrupted
Late communication
Status data recopilation takes too long
Communications through the network takes too long
Status data recopilation takes too long
Communications through the network takes too long
Health monitoring fails
Status is processed erroneously
Erroneous data communication
HW error
SW error

Fig. 12 Detection FTA

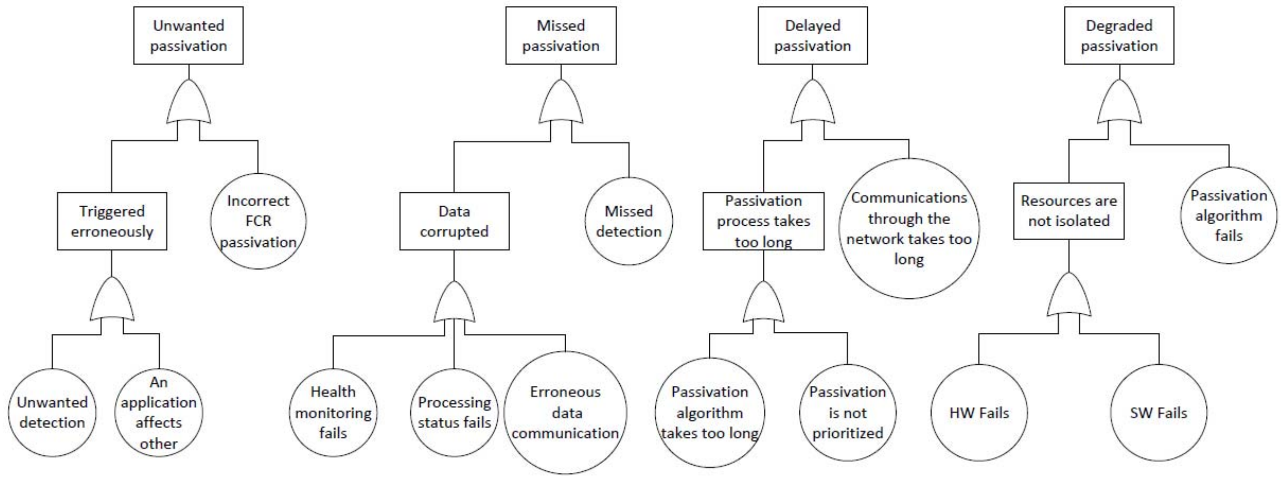

Unwanted passivation
Triggered erroneously
Incorrect FCR passivation
Unwanted detection
An application affects other
Missed passivation
Data corrupted
Missed detection
Health monitoring fails
Processing status fails
Erroneous data communication
Delayed passivation
Passivation process takes too long
Communications through the network takes too long
Passivation algorithm takes too long
Passivation is not prioritized
Degraded passivation
Resources are not isolated
Passivation algorithm fails
HW Fails
SW Fails

Fig. 13 Passivation FTA

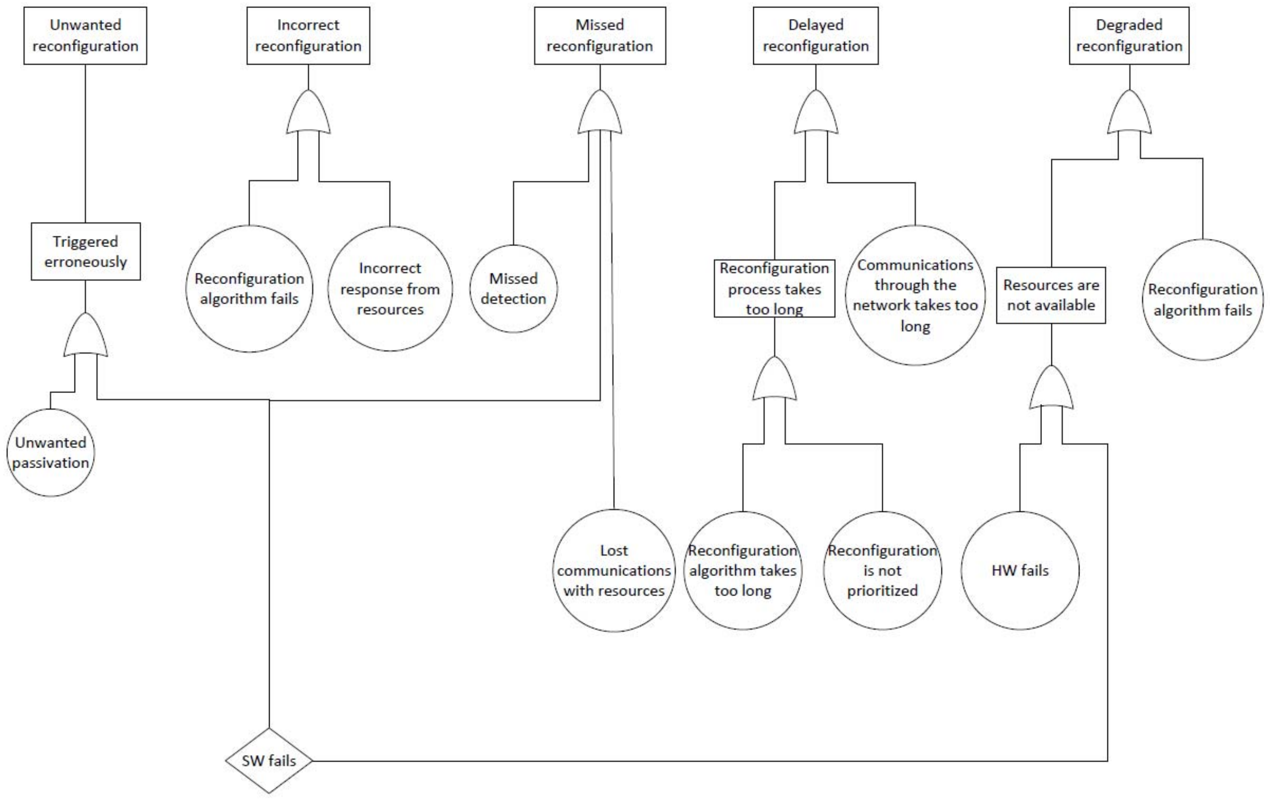

Unwanted reconfiguration
Triggered erroneously
Unwanted passivation
Incorrect reconfiguration
Reconfiguration algorithm fails
Incorrect response from resources
Missed reconfiguration
Missed detection
Lost communications with resources
Delayed reconfiguration
Reconfiguration process takes too long
Communications through the network takes too long
Reconfiguration algorithm takes too long
Reconfiguration is not prioritized
Degraded reconfiguration
Resources are not available
Reconfiguration algorithm fails
HW fails
SW fails

Fig. 14 Reconfiguration FTA

5.2 HARA for adaptation

FTAs were used as inputs for defining possible malfunctions that could trigger hazards when doing the adaptation. Based on these inputs the potential effects were analysed. After these inputs the effects were analysed. ASIL for these hazards were not taken into account as it is completely dependable on the functions to adapt.

Functions	Malfunctions	ID	Hazard Description	Potential Effect
F001-Detection	Unwanted detection	H001	Trigger the need for adaptation unnecessary	Overloading processor and other applications can be degraded or delayed
	Misdetection	H002	Adaptation is not triggered	A fault on a core node will continue and hazardous situations will not be mitigated
	Delayed detection	H003	Adaptation is delayed	There is no time for the driver to control the hazardous situation that triggers the adaptation
F002-Passivation	Unwanted passivation	H004	Application, partition, computing core, communication link are isolated	A critical application is isolated and this could lead to not being able to activate it again creating a hazardous situation
	Miss passivation	H005	Adaptation is not triggered, reconfiguration, reallocation and activation will not be triggered	A fault on a core node will continue and hazardous situations will not be mitigated
	Delayed passivation	H006	Isolation of the fault is delayed	A fault on an application could trigger dependent failure due to the delayed isolation
	Degraded passivation	H007	Isolation of the fault is not completed	A fault on an application could trigger dependent failure due to the not complete isolation
F003-Reconfiguration	Unwanted reconfiguration	H008	Reconfiguration is triggered unnecessarily without being isolated any application (as there is no fault)	Unwanted reallocation will be carried out
	Incorrect reconfiguration	H009	Resources changes are not applied correctly	A resource change into a passivated one could activate a faulty resource and the hazardous situation will not be mitigated.
	Miss reconfiguration	H010	Adaptation is not triggered, reallocation and activation will not be triggered	Even if the fault is isolated, the reallocation will not be carried out and thus, the loss of reactivation of some critical applications could lead to hazardous situations (or the hazardous situation is not mitigated)
	Delayed reconfiguration	H011	The action of changing the resources distribution is	Reallocation phase will be delayed

Functions	Malfunctions	ID	Hazard Description	Potential Effect
			delayed	
	Degraded reconfiguration	H012	Resource distribution change is not completed	The application might not have the resources needed
F004-Reallocation	Unwanted reallocation	H013	Reallocation is triggered unnecessarily without any reconfiguration has been previously carried out	Unwanted activation will be carried out
	Incorrect reallocation	H014	When there is no free space to reallocate the application, a lower ASIL level application replaces a higher ASIL level application.	Higher ASIL level application does not run anymore and it could lead to hazardous situations (or the hazardous situation is not mitigated)
	Miss reallocation	H015	Adaptation is not triggered, activation will not be triggered	Even if the fault is isolated, the reallocation/activation will not be carried out and thus the loss of reactivation of some critical applications could lead to hazardous situations (or the hazardous situation is not mitigated)
	Delayed reallocation	H016	The distribution of applications into resources is delayed	Activation phase will be delayed
	Degraded reallocation	H017	The distribution of applications into resources is not completed	The effect is the same as an incorrect allocation
F005-Activation	Unwanted activation	H018	Activation is triggered unnecessarily without any reallocation has been carried out	If there was no space, an application could be unnecessarily replaced
	Miss activation	H019	Adaptation is not triggered, the application does not operate	The no reactivation of some critical applications could lead to hazardous situations (or the hazardous situation is not mitigated)
	Delayed activation	H020	The application operates late	The delay on the activation could lead to hazardous situations
	Degraded activation	H021	The application does not properly operate	The hazardous situation is not mitigated

Table 2 Hazard analysis for the adaptation

5.3 Safety goals

This section includes the list and description of the high level safety goals for the adaptation.

Preliminary list of safety goals:

SG1: Ensure adaptation is correctly triggered

Adaptation is triggered based on information about a failure on a system element. In the case it is unnecessarily triggered due to an error on the detection, the processor can be overloaded and other applications can be degraded or delayed. One of the main goals for this mechanism is the correct detection of the adaptation need.

SG2: Ensure correct isolation of failures

The failure can affect different elements of the system and therefore the impact of the failure effects might differ. A fault on an application could lead to dependent failures as a result of the not correct isolation. The SAPC should be able to handle fault containment regions of different sizes and correctly isolate the possible outputs of these affected regions.

SG3: Resources are correctly configured

When adaptation is executing, it is important that the resources are rightly configured. Even if the fault is isolated, the adaptation will not be carried out and thus the loss of some critical applications could lead to hazardous situations if the resources are not reconfigured.

The adaptation mechanism not only needs to carry out the reconfigurations but also the initialization of replacement cores and activation of the applications on the defined cores. In addition it must be guaranteed that actuators and sensors are accessible from the replacement core in a proper way. Resources not only need to be available but also the applications should be able to run on those re-configured resources.

SG4: Applications are correctly activated

Once the applications are on the new core, those applications should also need to run in a safe way, being this especially important on high ASIL level application (ASIL D). If those applications are not available and running correctly at the same point as before the adaptation started, it could lead to hazardous situations.

SG5: Adaptation timing should be less that the time required to achieve a safe state

Adaptation should occur at the shortest time as possible. That time should not be more than the fault tolerant time interval (FTTI) of the application or we will run on the loss of control. FTTI [ISO26262 2011] determines the time-span in which a fault or faults can be present in a system before a hazardous occurs.

5.4 Safety Concept

After stating the adaptation related safety goals, this section defines a system level functional safety concept and briefly describes the functional safety concept at core node level so that the adaptation hazards are properly addressed.

Current automotive approaches don't usually support fail-operational applications. [SAFE project D3.3.2]. The system usually would enter in a safe state even if caused by the Quality Management

(QM) part. SafeAdapt functional safety concept follows the direction of future research challenges providing fail-operational behaviour at system level for different automotive functions. To get a common understanding some basic definitions of different failing strategies are explained below.

- **Fail-operational**: the component/system is still operational after one failure. In other words, one failure is tolerated and it will continue to fulfil its intended purpose at least until the driver can take back the control of the corresponding function or stops carefully.
- **Degraded operational (graceful degradation):** In this case, the fail-operational behaviour is degraded with respect to the full performance operation. The application performance could be degraded based on different performance indicators.
- **Fail-safe:** Here, the component/system enters into a safe state after a failure occurs. Thus, a critical failure is prevented. One of the possible examples within this group is the fail-silent behaviour.
- **Fail-silent: a** fail-silent node either functions correctly or stops working (turns into silent) after an internal failure is detected. In this second case, the node would not send any output data to the rest of the system staying externally quiet.
- **Critical failure**: the system goes into an incorrect behaviour which could lead to a catastrophic result.

One of the most common solutions is to combine fail-silent units in a proper way to achieve a fail-operational behaviour.

Despite the fact that current automotive approaches rarely support fail-operational applications, the current research starts to follow the direction of providing innovative functional safety concepts so fail-operational behaviour at system level for specific functions is achieved. It should be noted that the increase of complexity in automotive application functions requires more sophisticated functional safety concepts. Furthermore, ensuring functional safety in autonomous driving will demand safety monitoring at functional level with redundant algorithms. It is noted that the aerospace domain uses self-adaptation as such in all its highly safety-critical systems successfully already since decades. Thus self-adaptation for the automotive domain has been identified as a viable solution to achieve many upcoming challenges of distributed software systems as well.

In the case of SafeAdapt, fail-silent behaviour is implemented at the core node level whereas SAPC system level safety mechanism decides between different fault tolerant strategies or adaptation scenarios so fail-operational behaviour at system level is obtained. As previously stated, this section points out a refined system level functional safety concept at different levels so that the risks introduced by adaptation hazards are appropriately addressed. Especially, hardware architecture, software and communication perspectives are further explained.

In addition, the functional safety concept at core node level is briefly pointed out. The main mechanisms implemented for fault containment and mitigation of possible hazards are described at different levels. As a consequence for having these safety mechanisms included, the item is maintained in a safe state (with or without degradation) and a fault does not lead directly to the violation of the safety goal(s). The executed automotive function such as ACC (Adaptive Cruise Control) or SBW(Steer-by-wire), plays an important role on the selected adaptation strategy. This includes different redundancy strategies i.e., hot/warm/cold-standby or graceful degradation depending on the nature of the failure and the criticality of the safety function. In other words, to address the critical timing constraints imposed by the highest ASIL functions hot-standby implementations are required.

The following table summarizes in detail the possible adaptations covered by the SAPC.

Form of adaptation	Description
Different core node	An application is instantiated on a different core node. In case the previous core node is defective, the node will also be passivated
Degrade application	An application is provided with fewer re-sources during execution, such as optional input values or longer execution periods. The adaptation is based on different execution paths of the application. SAPC must be aware of these options, to consider them during the adaptation's planning phase.
Passivate application	An application is disabled either as part of a degradation strategy or as part of the passivation of the entire platform. Thus the application is not called by the scheduler anymore.

Table 3 Forms of adaptation

As previously reflected on deliverable D3.1, the architecture consists of two core nodes i.e. TMDP and RACE built on different hardware, platform software and with different safety mechanisms in place.

The functional safety concept at system architectural level is illustrated in the following figure. Fault Tolerant E/E architecture is introduced which includes several of the used fault tolerance mechanisms at system and ECU level together with fault detection, isolation and recovery strategy (FDIR).

These core nodes communicate in a time-triggered manner allowing synchronous communication between them. The required level of fault tolerance is ensured by different strategies. Some of the most important ones are highlighted in green in the figure and will be explained later on.

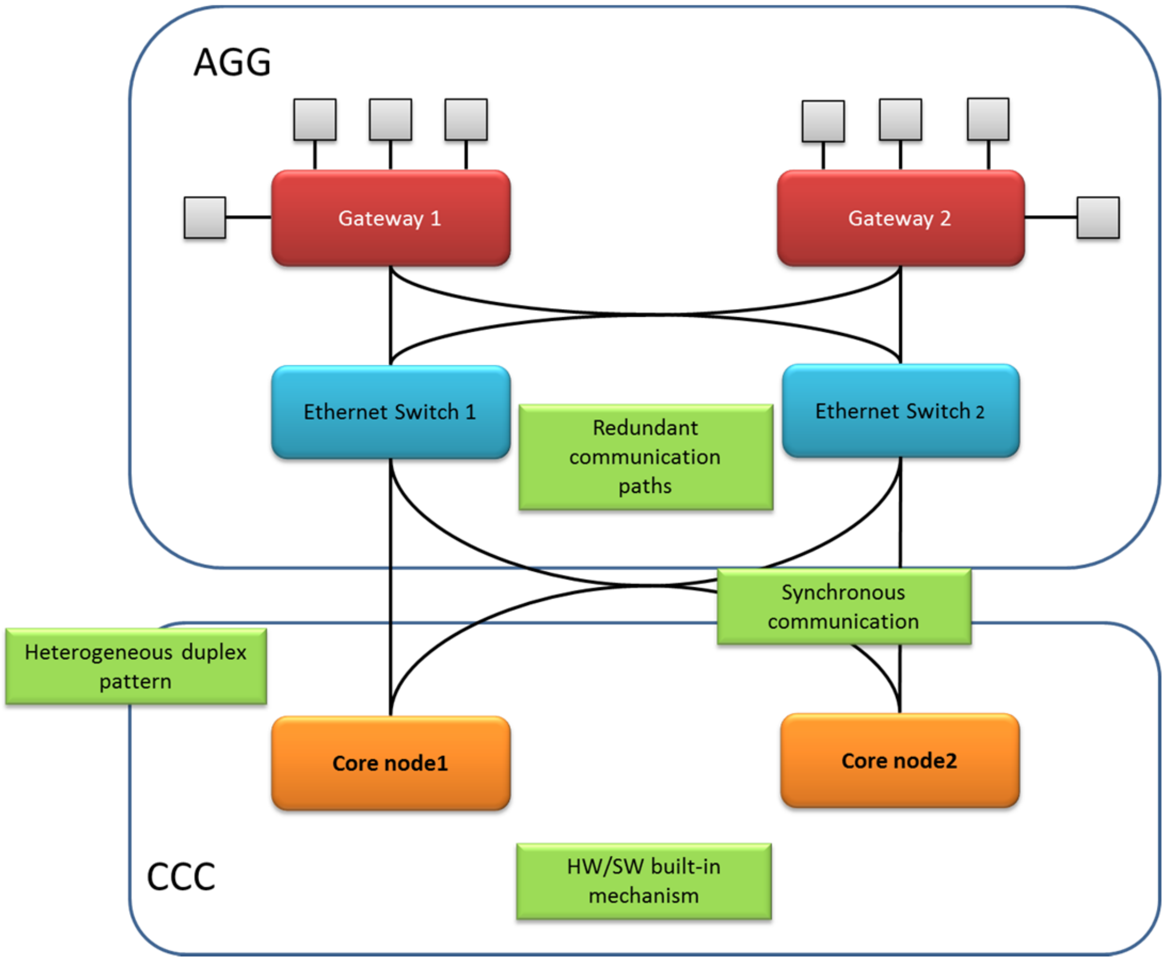


Fig. 15 Safety concept at architectural level

As illustrated in **Fig.** 15, the different elements in the network are distributed as double star architecture, this way possible communication failures are addressed. Furthermore, the required level of fault tolerance is achieved by a replication of the smart sensor and actuators. Sensors use the so called 2-out-of-3(2oo3) fault tolerance pattern where a voter determines which the correct sensing value is. It allows high protection against random hardware faults.

Several publications have appeared in recent years documenting fault-tolerance design patterns [Armoush 2010]. One of the preceding redundancy patterns is known as *Heterogeneous Duplex pattern*, *Heterogeneous Redundancy Pattern* or *Diverse Redundancy Pattern*. It deals with random and systematic faults increasing both reliability and safety of the system. As highlighted in a previous paragraph, it consists of two independent and diverse hardware channels (core node 1 -RACE- and 2 -TMDP-) designed in different technologies and developed by different teams. Especially, the combination of two different technologies such as Microcontroller (TMDP) and FPGA (RACE) provides effective coverage for systematic and common cause failures.

One of the available channels is known as the active module being the second one the so called standby or spare module. Depending on how the spare module is implemented three different modes for redundancy are available:

a) **Hot-standby:** both channels are continuously working. This solution is highly recommended when strict time constraints are a must and for high-safety level applications. Even though, one of the major drawbacks of this solution is the high power consumption.
b) **Warm-standby:** the standby spare channel runs in idle state. In case a fault is detected in the primary module, the standby one takes over its corresponding load.
c) **Cold-standby:** this low-power consumption solution is represented by a spare channel which waits for a failure on the primary channel. If this happens, it begins to operate immediately.

As previously stated, this solution has a high random and systematic failure rate being a very effective solution for applications requiring high safety integrity levels. Therefore, this solution has been applied to SafeAdapt. The backup core node will work in standby mode to take over the functions of the primary core node if the master core node fails.

The highest Automotive Safety Integrity Level (ASIL) functions require from hot-standby solutions to address the critical timing constraints.

5.4.1 Hardware/Software Built-in Safety Mechanisms (Core Node Level)

As a matter of fact, each core node should hold a set of hardware and software safety mechanisms to support ASIL D applications and to guarantee fail-silent behaviour at component level. The Siemens' implemented safety mechanisms are implemented at application level whereas Delphi´s solution combines hardware and software implemented safety mechanisms to either detect or correct certain set of hardware random failures.

More specifically, the implemented Siemens' core node safety mechanisms are based on the assumptions that hardware faults will lead to a failure-effect that can be observed looking on the ethernet frames sent out from bot lanes of its core node. On the contrary, Delphi´s system-on-chip solution has several hardware and software embedded safety mechanisms in order to detect/correct the aforementioned failures.

Regarding transient faults, they should be already covered at ECU level whereas not recoverable permanent faults are reported to the fault filter so the adaptation process begins. The detection of a not recoverable permanent fault at core node level such as a not recoverable memory failure, leads to an adaptation need.

In the same way as redundancy architectural patterns are developed at system level, lockstep architectures [ISO26262 2011] [Mariani et al. 2006] are a guarantee at core node level. As stated, different solutions are provided by the core node developers.

In the case of Siemens, a loosely coupled "SW lockstep" is implemented and the output packages compared every 10 ms in order to detect mistmatches.

Concerning the system-on-chip (SoC) used on TMDP, a hardware lockstep is implemented. Moreover, since the two channels can be diversely built, this fault tolerance pattern can deal with both systematic and random hardware failures of the CPU. Operations are executed on both channels and if a mismatch is manifested in the output of the two processors, a flag is activated. The hardware diversity provides effective coverage for common cause failures as well and systematic failures. The drawback of this approach is that it could be extensively complex to prove the diagnostic coverage. CPU failure modes are also covered by software implemented built-in self-test (BIST) features and the inclusion of a watchdog. This last one is capable of detecting scheduling and timing errors.

Regarding memory protection, all memory is protected by hardware Error Correction Code (ECC) and a memory protection unit (MPU) that stretches over the whole address space to detect any software interference.

On the other hand, software based integrity checks are also implemented at application level.

Monitoring offers a solution to the detection of clock (clock monitors) and voltage errors (undervoltage and overvoltage detection).

Platform implemented safety mechanisms serves as core node guarantees to implement the adequate error handling strategy at system level. When this error cannot be handled in a safe manner at ECU level, the SAPC is notified and the whole system starts the corresponding adaptation strategy to continue operating either degraded or fully operationally.

Following list summarizes the types of fault regions that the SAPC concept handles:

- *Platform failure/Core Node*: a random hardware failure that affects the whole core node. The whole platform is considered as the containment region
- *Memory failure*: permanent memory cell failures or of memory banks, or memory area failures are classified in general as memory failures.
- *Timing failure*: A failure on the internal watchdog, so that the outputs are no longer valid.
- *SoC Bus System failure*: Any problem detected on the SoC which is not recovered by CRC appliance thus the SoC is untrusted
- *Power supply failure*: A permanent fault or a transient power supply faults, like a crank pulse or other short power drop, that will make the core node to reset.
- *Application failure*: there is a failure on the application due to unavailability to the input data or a random hardware failure makes impossible to the application to run on a normal more. Software design failures are not covered.

For further detailed information about the error handling strategy at system level, the following table is introduced. Fault region determines where the failure occurs, detection mechanism points out how it is supposed to be handled at component level whereas fault containment represents where to contain the effect. Finally, system reaction for each case is set.

Fault Region	Detection mechanism	Fault Containment	System reaction
Core Node failure	Loosely coupled lockstep/ HW lockstep	System	Failover
Memory failure	Recoverable by MPU	ECU	No need for failover
	Not recoverable by MPU	System	Failover
Timing failure	SoC internal WD	System	Failover
	ECU WD	System	Failover
SoC Bus System failure	Recoverable by CRC	ECU	Depending on performance level comparison
	Not recoverable	System	Yes
Power supply failure	Fail-silent	System	Failover
Sensor failure	Input loss	ECU	Redundant path
	Input comparison	System	Degrade application
Network failure	Input comparison	ECU/System	Redundant paths

Table 4 Error handling strategy at system level

5.4.2 Communication perspective

Another key element of the architecture is the Health Vector (HV). The HV provides information about the status of the core-nodes, in particular errors, and is exchanged periodically. It contains platform specific status such as the running applications information and it is periodically sent from one core node to the other in a predefined time slot through a network connected by switches. It is absolutely essential that the system does not differ from the temporal behaviour defined at design time.

To simplify the temporal determinism and partition of the system, all computing platforms must communicate over a synchronous communication medium, such as a time-triggered network. In case of a not recoverable core node/ECU level failure such as a power failure, the health vector is not transmitted and the other core node takes the function over according to the predefined adaptation strategy. Yet if the failure source is a permanent fault detected by a safety mechanism such as the hardware lockstep the degraded performance level is set to 1 and the HV transmits this information to the other core node.. The HV also informs about the state of the core node. This provides a map of the status of the core nodes network where SAPC is deployed. A faulty core node could be a fail-silent stay where no HV is be transmitted or could stay in a fail-safe state where it continues transmitting the HV but informing that it is in an isolated mode.

In the same way, the application running mode (master/slave) is transmitted in order to know what the current status of a specific application is. Once again, it is important to underline that adaptation rules are predefined in the database.

HV includes two different mechanisms to ensure information correctness or data integrity:

- Cyclic redundant check
- Rolling counter

Data Validation (Integrity Check) is responsible to provide checks on the input data and the system itself during the execution of the derived algorithm. A cyclic redundancy check (CRC) is included for that purpose. Range checks or correctness checks are carried out by computing parity or CRC check. Likewise, a rolling counter is part of the HV to guarantee that the current message has been updated since the last iteration. This is used to check data omission.

5.4.3 Software Perspective

At software perspective the SAPC algorithm should also take care on avoiding and mitigation of hazards. These mechanisms help to complete a safe adaptation process by ensuring the avoidance of single points of failure leading to safety goal violations mentioned in the previous section.

The previous statement is further specified in the following figure where it can be seen that the adaptation software comprises different safety mechanisms (highlighted in green boxes) to shield the system from the violation of a safety goal.

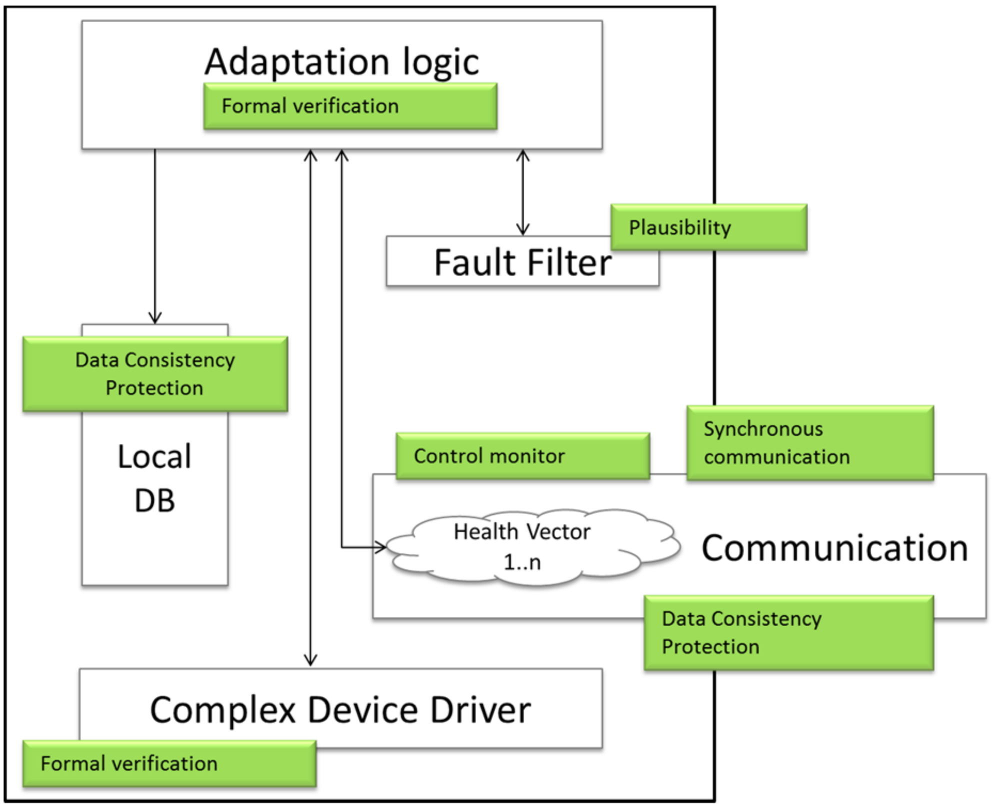


Fig. 16 Safety concept for the SAPC at software perspective

As reported by ISO26262-6, plausibility checks are recommended for ASIL A, B, C and highly recommended as error detection mechanisms at software architectural level for ASIL D. This software implemented safety mechanism checks the integrity of any signal by means of a specified reference model of the desired behaviour, assertion checks or comparing signals from different sources. To put it in another way, some predicates are defined in a set of variables to determine their validity at runtime. This is used to filter the set of failures that SAPC can handle and performance range of the core. Plausibility checks would be part of the fault filtering, in this way the SAPC input will not accept failures that can be handled. Specifically the failures filter should be the ones included on the previous subsection.

The integrity check provides a validation on the input data and the system itself during the execution of the derived algorithm. Range checks or correctness checks by computing parity or CRC check complete the set.

Data consistency protection should be ensured between the local databases of the core nodes (RACE and TMDP). The content of both of them must be equal and neither incoherencies nor inconsistencies will be found between them. Even if this feature is guaranteed during the design time, the local databases include information redundancy mechanisms such as parity bits or CRCs.

On the SafeAdapt core software implementation, we also include different patterns in order to fulfil the safety requirements we have identified before.

Likewise, fault tolerance is achieved through different mechanisms to detect or correct random hardware faults, systematic ones must be either avoided or removed during design time. Jean-Claude Laprie [Laprie 1992] argued that techniques such as formal verification can be applied to ensure fault removal of the design. This could be carried out by performing model checking to find possible design errors of Adaptation Logic and Complex Device Driver modules. Especially, model checking would be performed to verify whether the component model meets a given specification.

5.5 Arguments about safety verification

“A safety case communicates in a comprehensive and structured way set of safety argumentation supported by evidence which demonstrates that the system is safe for a certain context.”

The main concepts handled on a safety case are:

- Arguments: Explanation on how the evidences can be interpreted as indicating safety for the objectives/requirements …
- Evidences: Results from observing the properties of a system. There are different types of evidences.

The OMG has defined a standard SACM (Structured Assurance Case Metamodel) that deals with the safety case modelling. SACM aims to provide a modelling framework to allow users to express and exchange their argument structures. The representation of an argument in SACM does not imply that the argument is complete, valid, or correct. Similarly, the evaluation or acceptance of an argument by a separate party is not covered by the SACM. In the SACM model, structured arguments comprise argument elements (primarily claims) that are being asserted by the author of the argument, together with relationships that are asserted to hold between those nodes. [SACM 2014]

The SACM does not propose a graphical notation but rather reference to existing ones (GSN and CAE) GSN stands for Goal Structuring Notation. GSN explicitly represents the individual elements of any safety argument (claims, evidence and context) and (the relationships that exist between these elements (i.e. how individual requirements are supported by specific claims, how claims are supported by evidence and the assumed context that is defined for the argument). The principal symbols of the notation are shown in the next figure.

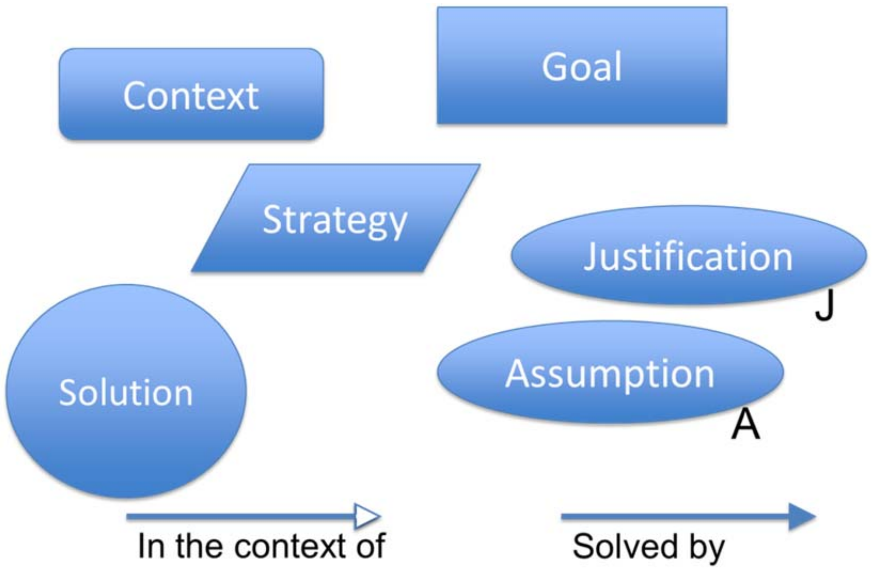


Fig. 17 Main elements for the GSN graphical notation

Prossurance is a safety assurance management system to support a cost-effective compliance assessment and certification of safety-critical products in sectors such as aerospace, railway, maritime and automotive.

Prossurance can work both as a file-based or database-base (Postgress) tool. Prossurance has been developed with the following technologies: Eclipse Kepler with GMF and EMF, XText, Subversion (SVN) Team Provider for artefact versioning if desired, a subversion client such as TortoiseSVN, Java Environment 1.7, Windows Operating System.

Prossurance has a complete functionality to create argumentation diagrams applying the GSN graphical notation. Prossurance includes a safety case editor that internally uses the SACM metamodel and takes advantage of the GSN graphical notation.

For detailed information, the safety case diagrams concerning safe adaptivity have been included in the following pages. To some extent, the main goal of these safety case diagrams is to define the strategy of which the safety goals verification process should be. This helps to justify that the system is acceptably safe and to enumerate which would be all the necessary evidences in order to prove that safety goals are correctly addressed.

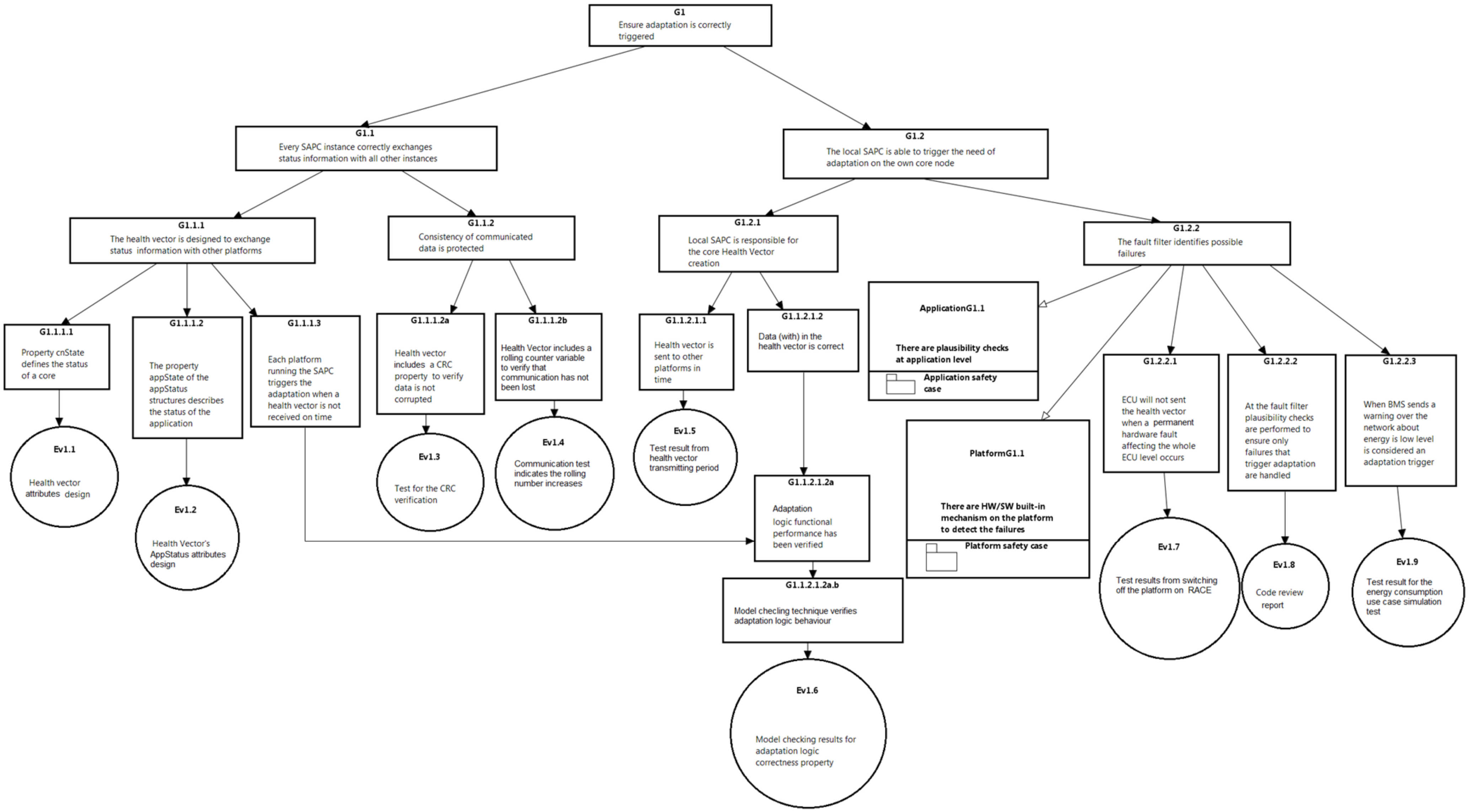

G1
Ensure adaptation is correctly triggered
G1.1
Every SAPC instance correctly exchanges status information with all other instances
G1.2
The local SAPC is able to trigger the need of adaptation on the own core node
G1.1.1
The health vector is designed to exchange status information with other platforms
G1.1.2
Consistency of communicated data is protected
G1.2.1
Local SAPC is responsible for the core Health Vector creation
G1.2.2
The fault filter identifies possible failures
G1.1.1.1
Property cnState defines the status of a core
G1.1.1.2
The property appState of the appStatus structures describes the status of the application
G1.1.1.3
Each platform running the SAPC triggers the adaptation when a health vector is not received on time
G1.1.1.2a
Health vector includes a CRC property to verify data is not corrupted
G1.1.1.2b
Health Vector includes a rolling counter variable to verify that communication has not been lost
G1.1.2.1.1
Health vector is sent to other platforms in time
G1.1.2.1.2
Data (with) in the health vector is correct
ApplicationG1.1
There are plausibility checks at application level
Application safety case
PlatformG1.1
There are HW/SW built-in mechanism on the platform to detect the failures
Platform safety case
G1.2.2.1
ECU will not sent the health vector when a permanent hardware fault affecting the whole ECU level occurs
G1.2.2.2
At the fault filter plausibility checks are performed to ensure only failures that trigger adaptation are handled
G1.2.2.3
When BMS sends a warning over the network about energy is low level is considered an adaptation trigger
Ev1.1
Health vector attributes design
Ev1.2
Health Vector's AppStatus attributes design
Ev1.3
Test for the CRC verification
Ev1.4
Communication test indicates the rolling number increases
Ev1.5
Test result from health vector transmitting period
G1.1.2.1.2a
Adaptation logic functional performance has been verified
G1.1.2.1.2a.b
Model checling technique verifies adaptation logic behaviour
Ev1.6
Model checking results for adaptation logic correctness property
Ev1.7
Test results from switching off the platform on RACE
Ev1.8
Code review report
Ev1.9
Test result for the energy consumption use case simulation test

Fig. 18 Excerpt of the safety case (G1)

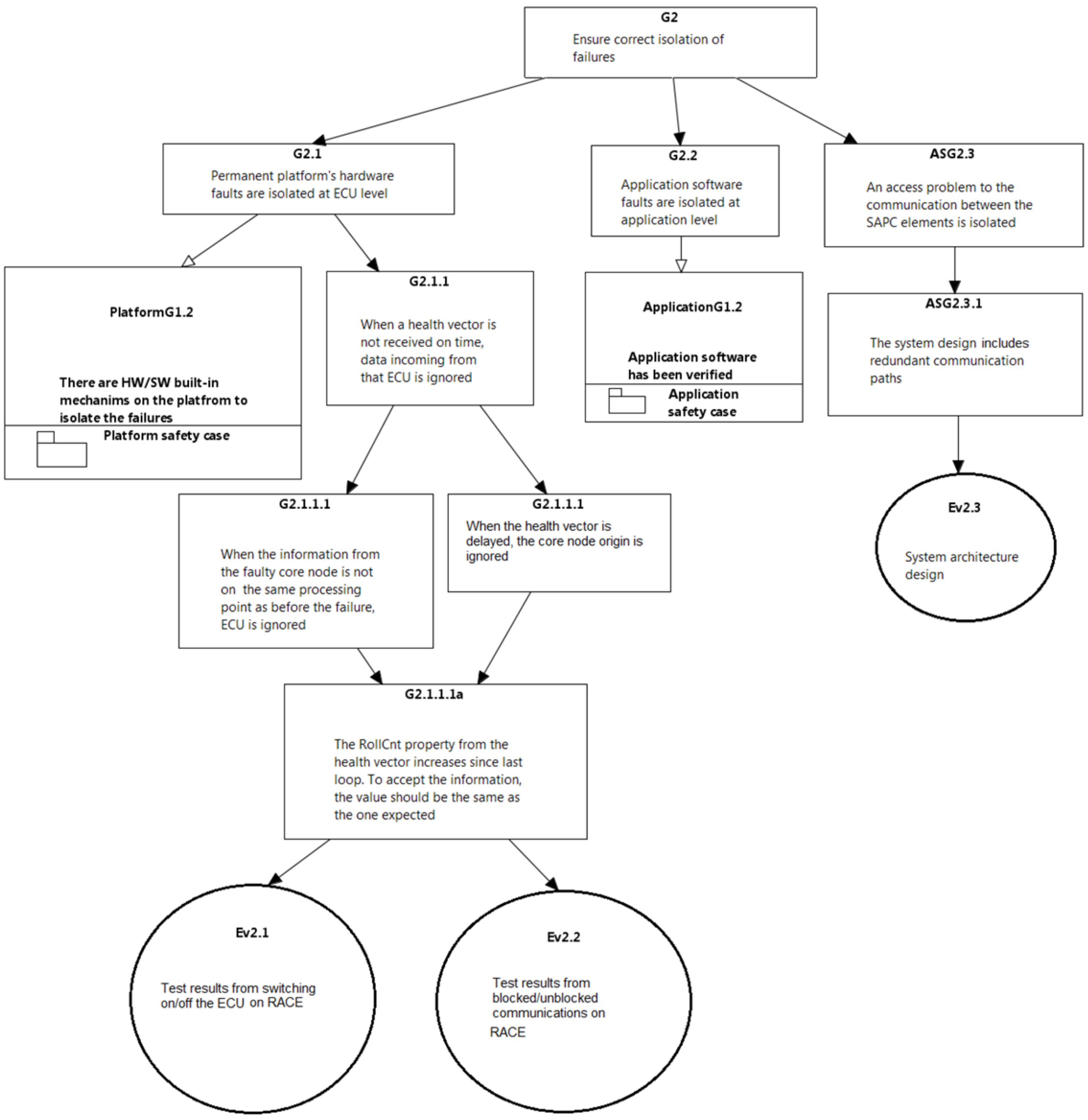

G2
Ensure correct isolation of failures
G2.1
Permanent platform's hardware faults are isolated at ECU level
G2.2
Application software faults are isolated at application level
ASG2.3
An access problem to the communication between the SAPC elements is isolated
PlatformG1.2
There are HW/SW built-in mechanims on the platfrom to isolate the failures
Platform safety case
G2.1.1
When a health vector is not received on time, data incoming from that ECU is ignored
ApplicationG1.2
Application software has been verified
Application safety case
ASG2.3.1
The system design includes redundant communication paths
Ev2.3
System architecture design
G2.1.1.1
When the information from the faulty core node is not on the same processing point as before the failure, ECU is ignored
G2.1.1.1
When the health vector is delayed, the core node origin is ignored
G2.1.1.1a
The RollCnt property from the health vector increases since last loop. To accept the information, the value should be the same as the one expected
Ev2.1
Test results from switching on/off the ECU on RACE
Ev2.2
Test results from blocked/unblocked communications on RACE

Fig. 19 Excerpt of the safety case (G2)

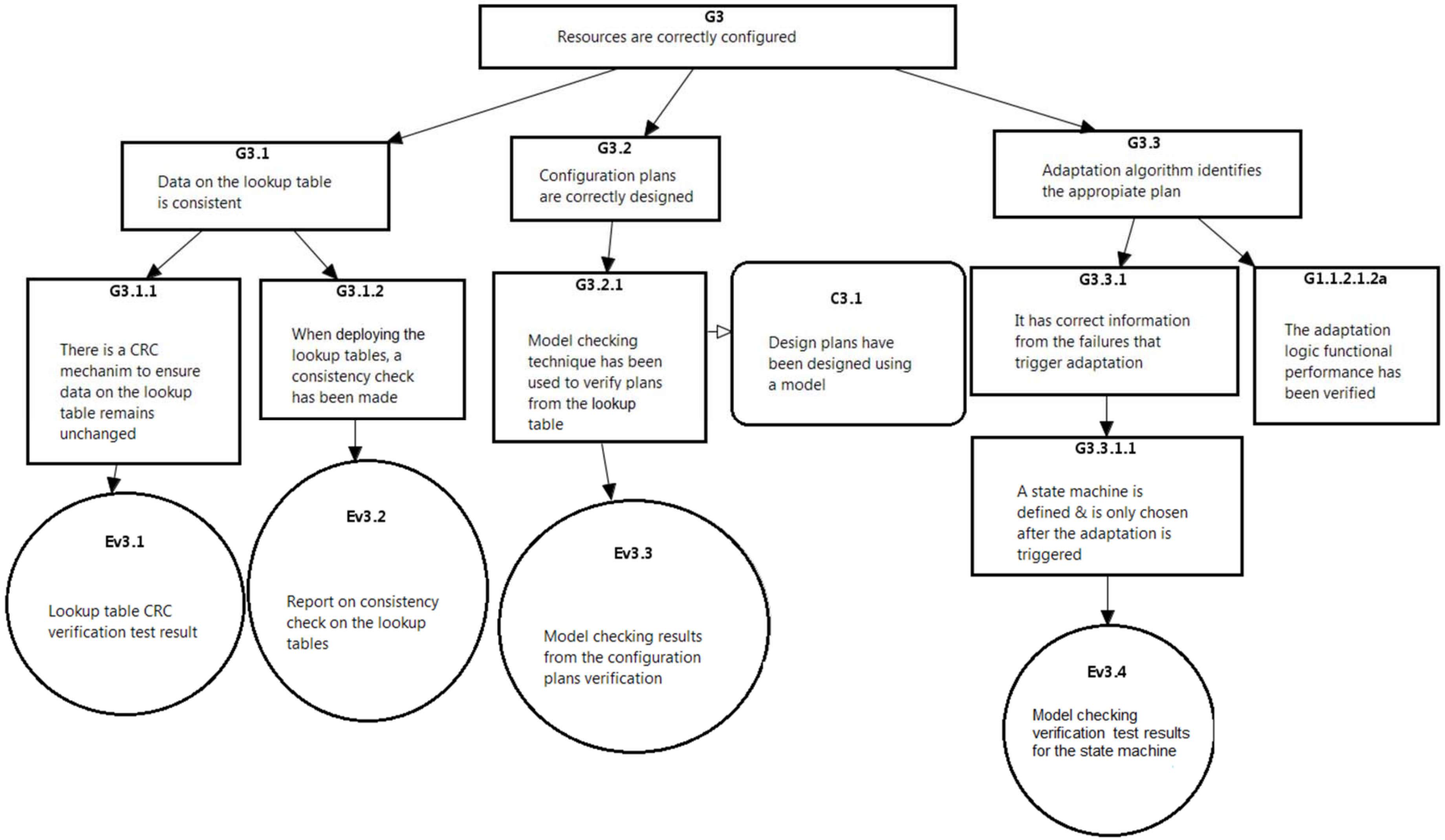

G3
Resources are correctly configured
G3.1
Data on the lookup table is consistent
G3.2
Configuration plans are correctly designed
G3.3
Adaptation algorithm identifies the appropiate plan
G3.1.1
There is a CRC mechanim to ensure data on the lookup table remains unchanged
G3.1.2
When deploying the lookup tables, a consistency check has been made
G3.2.1
Model checking technique has been used to verify plans from the lookup table
C3.1
Design plans have been designed using a model
G3.3.1
It has correct information from the failures that trigger adaptation
G1.1.2.1.2a
The adaptation logic functional performance has been verified
G3.3.1.1
A state machine is defined & is only chosen after the adaptation is triggered
Ev3.1
Lookup table CRC verification test result
Ev3.2
Report on consistency check on the lookup tables
Ev3.3
Model checking results from the configuration plans verification
Ev3.4
Model checking verification test results for the state machine

Fig. 20 Excerpt of the safety case (G3)

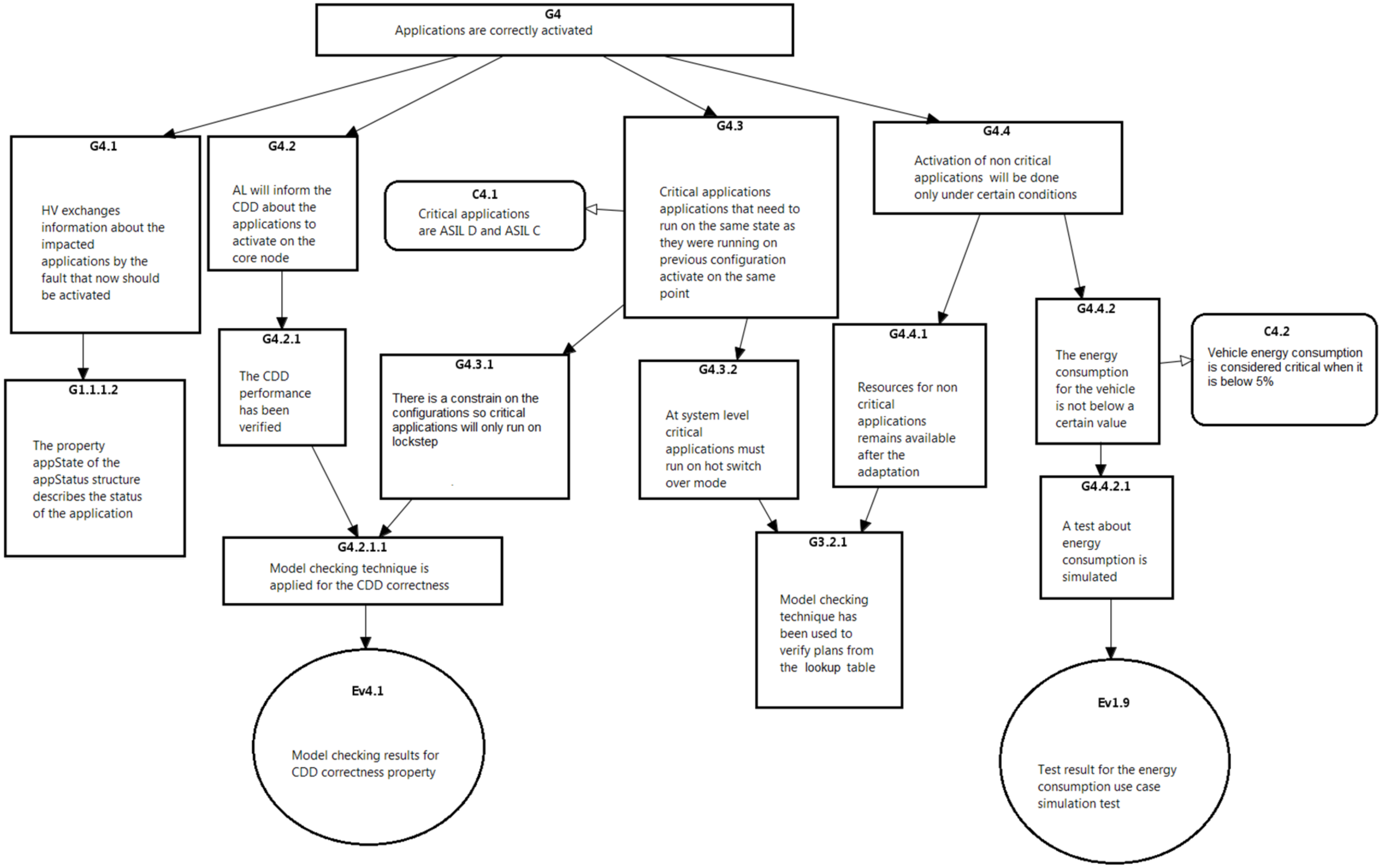

G4
Applications are correctly activated
G4.1
HV exchanges information about the impacted applications by the fault that now should be activated
G4.2
AL will inform the CDD about the applications to activate on the core node
C4.1
Critical applications are ASIL D and ASIL C
G4.3
Critical applications applications that need to run on the same state as they were running on previous configuration activate on the same point
G4.4
Activation of non critical applications will be done only under certain conditions
G1.1.1.2
The property appState of the appStatus structure describes the status of the application
G4.2.1
The CDD performance has been verified
G4.3.1
There is a constrain on the configurations so critical applications will only run on lockstep
G4.3.2
At system level critical applications must run on hot switch over mode
G4.4.1
Resources for non critical applications remains available after the adaptation
G4.4.2
The energy consumption for the vehicle is not below a certain value
C4.2
Vehicle energy consumption is considered critical when it is below 5%
G4.2.1.1
Model checking technique is applied for the CDD correctness
G3.2.1
Model checking technique has been used to verify plans from the lookup table
G4.4.2.1
A test about energy consumption is simulated
Ev4.1
Model checking results for CDD correctness property
Ev1.9
Test result for the energy consumption use case simulation test

Fig. 21 Excerpt of the safety case (G4)

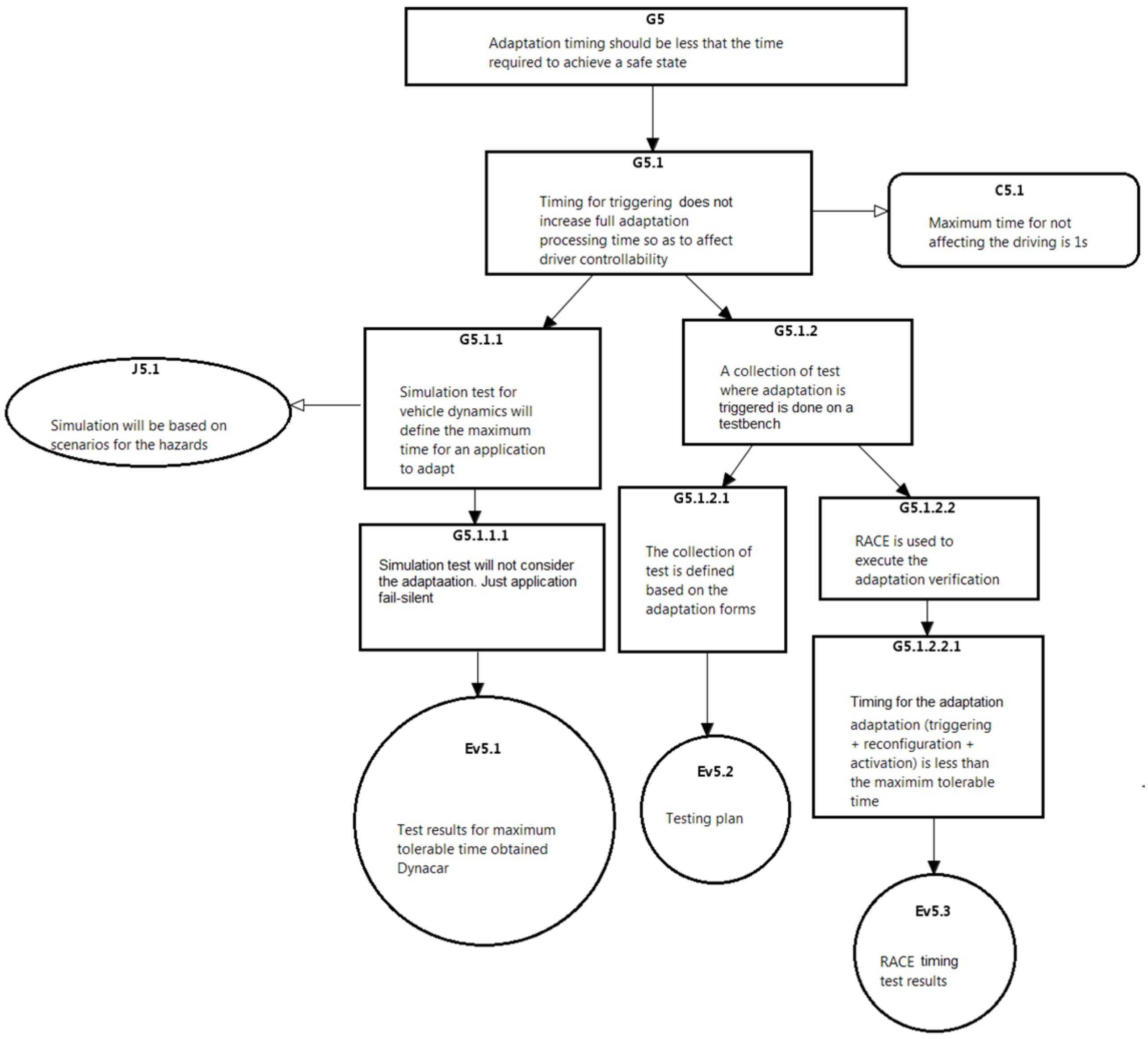

G5
Adaptation timing should be less that the time required to achieve a safe state
G5.1
Timing for triggering does not increase full adaptation processing time so as to affect driver controllability
C5.1
Maximum time for not affecting the driving is 1s
J5.1
Simulation will be based on scenarios for the hazards
G5.1.1
Simulation test for vehicle dynamics will define the maximum time for an application to adapt
G5.1.2
A collection of test where adaptation is triggered is done on a testbench
G5.1.1.1
Simulation test will not consider the adaptaation. Just application fail-silent
G5.1.2.1
The collection of test is defined based on the adaptation forms
G5.1.2.2
RACE is used to execute the adaptation verification
G5.1.2.2.1
Timing for the adaptation adaptation (triggering + reconfiguration + activation) is less than the maximim tolerable time
Ev5.1
Test results for maximum tolerable time obtained Dynacar
Ev5.2
Testing plan
Ev5.3
RACE timing test results

Fig. 22 Excerpt of the safety case (G5)

5.6 Safe Adaption Platform Core safety goals verification

5.6.1 Introduction

Safety critical systems have to fulfil safety requirements in addition to functional requirements. Safety requirements describe the characteristics that a system must have in order to be safe. This involves the identification of all possible hazards that can take place, and that may harm people or the environment. Safety-related issues are often captured in standards describing products and processes to be considered throughout the life-cycle of a safety critical system. In SafeAdapt project, the safety concept of ISO 26262 during the design and product development phase will be taken into consideration. The safety standard ISO 26262 is an implementation of the more general IEC 61508 standard that addresses safety issues in the automotive industry. The standard prescribes that a safety case should be created for every system that has safety-related features, and it says that part of the system documentation should provide evidence for the fulfilment of safety requirements, thus guaranteeing functional safety.

5.6.2 Safety Case

Part of the certification process in the automotive domain is the assessment of a system through an inspection agency. To convince inspectors that a system is safe, a safety case should be created. The safety case communicates a clear, comprehensive and defensible argument that a system is acceptably safe in its operating context. The argument should make clear that it is reasonable to assume the system can be operated safely. It is typically based on engineering judgment rather than strict formal logic, and it provides evidence that the risks (hazards) associated with the operation of the system have been considered carefully, and that steps have been taken to deal with the hazards appropriately.

The safety argument (SA) must identify all matters significant to the safety of the system and demonstrate how these issues have been addressed. A convenient way to define a safety argument is through the goal structuring notation (GSN) devised by Kelly [Kelly 1998] which is based on earlier work by Toulmin on the construction of arguments [Toulmin 1958]. An argument consists of claims whose truth should be proven. The facts used to prove the claims are referred to as data, and the justification for why data prove a claim is described by warrants. If it is possible to dispute a warrant, backing can be used to show why the warrant is valid.

The main elements of the GSN are goals and solutions. Goals correspond to Toulmin's claims whereas solutions relate to Toulmin's data, also termed evidence. For constructing a safety case, we have to determine which evidence is required for a particular safety argument, and why the evidence supports the claim. According to the GSN, the safety case starts with a top-level claim, or a goal, such as "the system is safe" or "safety requirements have been realized."

The top-level claim is then decomposed into sub-ordinate claims down to a level that a sub-claim can be proven by evidence. The concepts of the GSN are displayed in the example in the next figure.

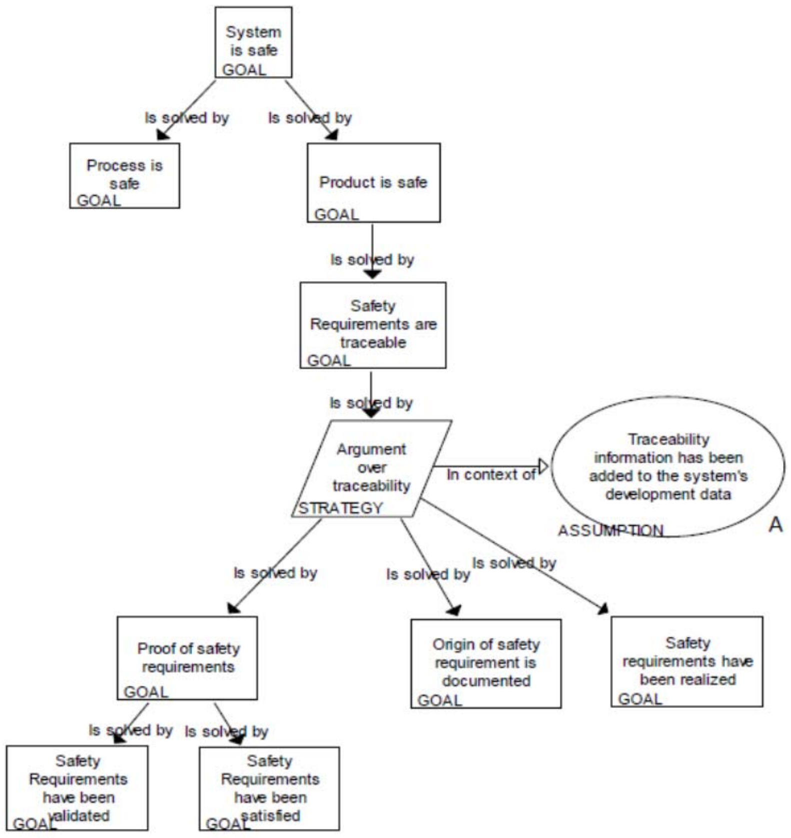

System is safe
GOAL
Is solved by
Is solved by
Process is safe
GOAL
Product is safe
GOAL
Is solved by
Safety Requirements are traceable
GOAL
Is solved by
Argument over traceability
STRATEGY
In context of
Traceability information has been added to the system's development data
ASSUMPTION
A
Is solved by
Is solved by
Is solved by
Proof of safety requirements
GOAL
Origin of safety requirement is documented
GOAL
Safety requirements have been realized
GOAL
Is solved by
Is solved by
Safety Requirements have been validated
GOAL
Safety Requirements have been satisfied
GOAL

Fig. 23 Example of GSN tree, decomposition of the goal “the product is safe”

Claims and sub-claims, or goals, are represented as rectangular boxes and evidence, or solutions, as circles. A strategy-node, represented as rhomboid, contains the explanation why a goal has been decomposed.

The argument can be constructed by going through the following steps [1, 3]:

1. Identification of the goals to be supported.
2. Definition of the basis on which the goals are stated.
3. Identification of the strategy to support the goals.
4. Definition of the basis on which the strategies are stated.
5. Elaboration of the strategy including the identification of new goals and starting from step 1, or moving to step 6.
6. Identification of a basic solution that can be proven.

It is difficult to propose a standard safety case structure that may be valid for most systems. However, some of the argumentation will be the same for many systems, such as “all safety requirements have been realized” or the like. Such argumentation structures, or so-called safety case patterns, may be reused in several safety cases for different systems. In the literature, several theories have been proposed to explain that by using such patterns, safety cases can be devised much faster. Kelly and McMermid [Kelly McDermid 1997] proposed safety case patterns as a means to reuse common structures in safety cases. [Wagner et al 2010] use argument patterns to reusable parts of our safety case and provide them as building blocks for future automotive safety cases. They also mentioned that "it is desirable to establish general and domain-specific patterns as a pattern library for a faster and easier creation of safety arguments."

[Hawkins Kelly 2009] and [Hawkings et al 2011] mentioned "the effectiveness of the software safety argument patterns has been demonstrated through application to a number of case studies".

The drawback of this approach is that potential systematic failures could be introduced and this bears a risk in the direction of the next level which is autonomous driving. Therefore, we would like to emphasize that further work needs to be done in this area to improve the current state of the art.

A particular GSN decomposition proposed by the EU project EASIS [EASIS 2006] organizes the argumentation into a product branch and a process branch, claiming that “a system is safe” if “the process is safe” and “the product is safe”. The safety of the process can be assured through application of certified development standards, such as the IEC 61508 or the V-model. Here, questions should be asked about how the product is developed, such as “did we perform hazard analysis?”, “do we have a hazard checklist?”, “did we perform a preliminary hazard identification?”, “did we implement the results of the preliminary hazard identification?”, and so on [EASIS 2006]. One of the tools considered in SafeAdapt is Prossurance, a product and process assurance management system to support the compliance assessment and certification of safety-critical systems such as automotive products that need to comply with ISO 26262, where all of these questions are addressed.

On the product side, we can decompose the claim “the product is safe” into the sub-goal “the safety requirements are traceable” which turns the satisfaction of our safety case into a traceability problem. We can argue, that if all safety related aspects of our system can be traced to their origin and to their realization, the system is safer, given the process is safe (proof of the process branch). Traceability is a prerequisite for assessment and validation of the safety goals, which we refer to as “proof of safety requirements”. We can then decompose the traceability goal further into “proof of safety requirements”, “origin of safety requirements documented”, and “safety requirements realized”. This extended organization of the safety case is depicted in the previous figure and, through its general nature, it can be used as a pattern for all systems.

It may be envisaged to adapt such approach by the long term proven aerospace development standards such as DO RTCA 178B (or the new version “C”) for software or the DO RTCA 254 for hardware. They define standard development processes that need to be followed and an independent review authority to prove argumentation and design. However, such investigation and suitable adaptation for the automotive industrial domain, exceeds the current means of the project.

5.6.3 Traceability of Safety Requirements

In the previous section, we argued that part of the proof of a safety case can be achieved through tracing all safety requirements to the respective development documents. This is fully in line with ISO 26262 since it demands that “the origin, realization and proof for a requirement are clearly described in the documentation” of a system. A requirement is a condition or an ability which shall be fulfilled by the system. The origin of a requirement is a rationale why this requirement has been elicited for the system. The realization demonstrates how/where the requirement is implemented in the final system. A proof for a requirement means that it should be demonstrated that the requirement has an origin and that it is implemented, in other words, that the requirement is traceable across all development documents in both directions, forwards and backwards. The documents comprise the hazards possible, the safety goals, the safety requirements, design elements, and implementation elements, plus associated review documents.

As shown in the previous figure, the “product is safe”-branch is decomposed into a traceability sub-goal that is split into various traceability claims, i.e., “origin of safety requirements documented”, “safety requirements realized”, and “proof of safety requirements”. This last goal is decomposed into two sub-goals, “safety requirements validated” and “safety requirements satisfied” which can be traced to the respective documents that deal with those issues. The origin of a safety requirement can be demonstrated by backward traceability. Safety requirements are derived from hazards and safety goals. Every safety requirement should be linked to at least one safety goal, and every safety goal should be linked to a hazard. But also forward traceability is important, so that for every hazard, there is a safety goal and for every safety goal, there should be an associated safety requirement.

6 Changes proposed for compliance to ISO 26262

This chapter introduces some example of methods and techniques that would need further analysis to be in conformance with the standard's objectives and, at the same time, induces a critical thinking so that a safer system could be built.

Method/technique	Level of appliance	Rationale
Formal methods	System / hardware / software	Formal methods can be used to verify each of the possible adaptation situations the system could run into.
Safety analysis	System	As a possible safety analysis, the combination of state-space and non-state-space model based techniques.
STPA [Leveson 2012]	System	A New Hazard Analysis Technique has been used for a preliminary HARA analysis on different domains where complex interactions occur. However, its use on adaptive systems needs further research.
Model-based development	System	It enables low cost, iterative investigation and early verification and validation being able to re-design before the real implementation is carried out. In the same way, the adaptation concept and its development could be not only verified but also evaluated and re-designed depending on the obtained results.
Simulation	System / Hardware / Software	For functional behaviour verification For fault injection for safety and dependability validation

Table 5 Summary of methods/tools that need further analysis to be applied on adaptive systems

6.1 Formal Methods

The use of formal methods is widely spread across different safety standards. This term specifies the employment of mathematical techniques during the different phases of a product development (hardware and software) from requirement specification through development and verification. Some of those standards recommend formal methods, however, their use is not highly recommended and the do not appear in all the previous phases.

Concerning ISO 26262, formal methods are used in order to prove the correctness of a system against the formal specification of its required behaviour [Exposito et al. 2011]. Yet they mostly appear always in the background only as recommended and after semi-formal notations (see **Table 6**). The possible reason for that is that as Storey [Storey 1999] stated, their use is not very simple. For this reason, formal methods are rarely applied throughout the whole development lifecycle. This means that while the use of formal specification is quite straightforward, the process of proving the equivalence of different stages of the design is a much specialised task requiring a high degree of mathematical ability.

To be more precise, formal methods are referred as formal notation and formal verification during software and supporting processes parts of ISO 26262-8.

	ISO 26262 Phase		ASIL			
			A	B	C	D
Formal notations	6	Notations for software architectural design	+	+	+	+
		Notations for software unit design	+	+	+	+
	8	Specifying safety requirements	+	+	+	+
Formal verification	6	Methods for the verification of the software architectural design	o	o	+	+
		Methods for the verification of software unit design and implementation	o	o	+	+
	8	Methods for the verification of safety requirements	o	+	+	+

Table 6 Use of formal methods across ISO 26262

As it can be seen in the previous table, no reference regarding hardware specification, design or verification can be found. Actually, only the ones related to hardware safety requirements are briefly described in ISO 26262-8. Especially, design, walk-through, safety analyses and simulation development by hardware prototyping are pointed out as possible hardware design verification techniques. Moreover, their usage is not even highly recommended for none of the exposed phases.

As matter of fact, it would be interesting to consider its application not only to software unit and architectural design, but also to hardware design and verification. In consequence, it could be proved that a specific circuitry correctly meets its intended function avoiding and preventing faults to be introduced into a system.

Even if formal verification is only recommended and not even highly recommended, for software unit and architectural design verification, we believe that techniques such as model checking could help avoiding and removing software systematic faults from blocks like SAPC adaption logic.

To sum up, it would be a potential commitment to complete and promote what ISO 26262 recommends with respect to recommendation levels and applied phases of formal methods. Due to the fact that they can provide multiple benefits during specification, design and verification, it would be a good choice to make them a complementary choice to semi-formal methods. Thus we consider the standard should include a more accurate and less ambiguous definition with respect to the use of formal methods during hardware and software verification phases making its appliance highly recommended for the highest automotive safety integrity levels.

The previous outcome has provided us with the opportunity to apply formal verification to the development, so that fault avoidance can be assured. At design time, we proposed the definition of a state machine to ensure that the algorithm is executing only the necessary logic for the adaptation phases, in which it currently resides.

Moreover, non-functional properties, such as Worst-Case Execution Time (WCET), should be verified formally by a model checking approach. Also, we propose to simulate communications behaviour in early development stages.

6.2 Safety Analysis

Standard methods like FMEA and FTA recommended by ISO 26262 are not directly applicable for adaptive systems. They try to find only cause-consequence relationships between component failures and system failures [Gudemann et al 2006]. Complementary/Extended techniques are required.

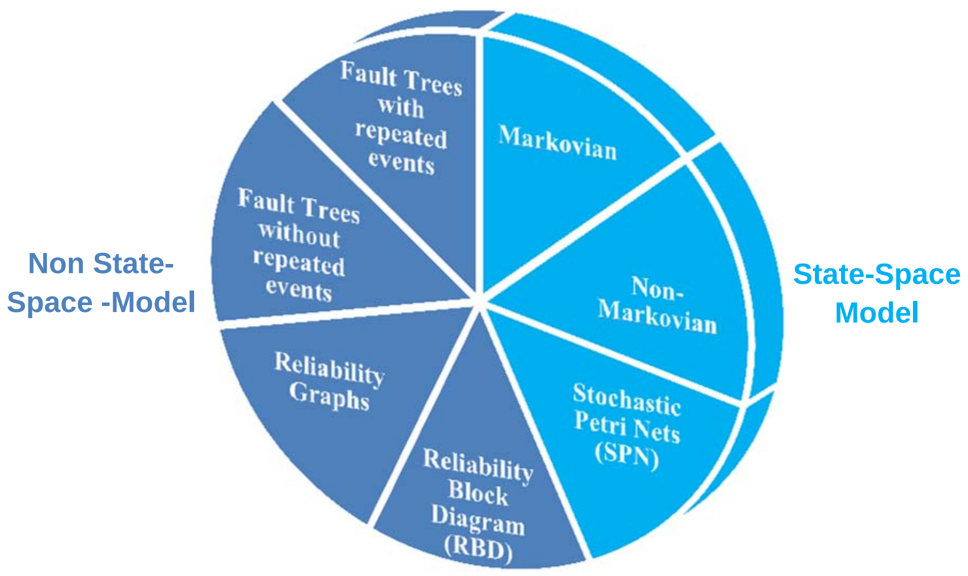

Non State-Space -Model
Fault Trees with repeated events
Fault Trees without repeated events
Reliability Graphs
Reliability Block Diagram (RBD)
Markovian
Non-Markovian
Stochastic Petri Nets (SPN)
State-Space Model

Fig. 24 Classification of the safety analysis techniques

The following table depicts a state of the art of the different safety analysis techniques regarding what is currently required and what it would be needed addressing adaptive systems.

ISO 26262 current state		ISO 26262 addressing adaptive systems	
Qualitative Methods	FMEA, FTA, ETA	Extension of the non-state-space model (state space + non-state space)	FTA + Markov
			Dynamic Fault Tree
			Dynamic Reliability Block Diagrams
			Boolean Driven Markov Processes
Quantitative Methods	FME(D)A, FTA, ETA, Markov Models, Reliability Block Diagrams	Stochastic Petri nets	
		Adaptive Transition Systems	
		State/Event Fault Tree (SEFT)	

Table 7 Safety analysis techniques

As mentioned, new solutions would be required to perform safety analysis in an adequate manner in the context of adaptive systems. Some of the most important techniques are briefly defined below [Manno 2011]:

1. **Markov Chains** [Kaiser et al. 2007]: a state machine whose transitions are labelled with transition rates, i.e. conditional probabilities that the transition to a given successor state occurs in the next small time interval, provided that the system is in the source state.
2. **FTA + Markov Chains:** a popular tried and tested technique to model dynamic systems in the industry. The combination of both techniques could produce more reliable data. This

technique has high degree of appropriateness and easiness as they are easily understandable. [Mouaffo el al. 2013]

3. **DFT (Dynamic Fault Tree):** extend the modelling capabilities of FTs and use a state-space based low level representation for the solution of the model.
4. **SEFT (Single Event Fault Tree)** [Kaiser et al., 2007]: Safety analysis combining elements from FTA and State-charts. Since it provides a clear understanding between state and event, this technique could take in account the adaptive behaviour the system could have. This is a clear difference compared to traditional fault trees or Component Fault Trees (CFT) [Essarel] in which fault trees are built based on failures of the component of a system.

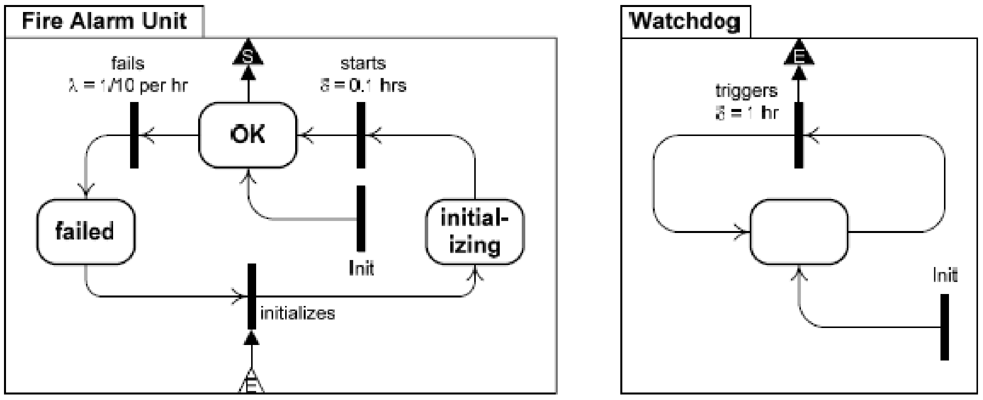


Fig. 25 Fire alarm component and watchdog component

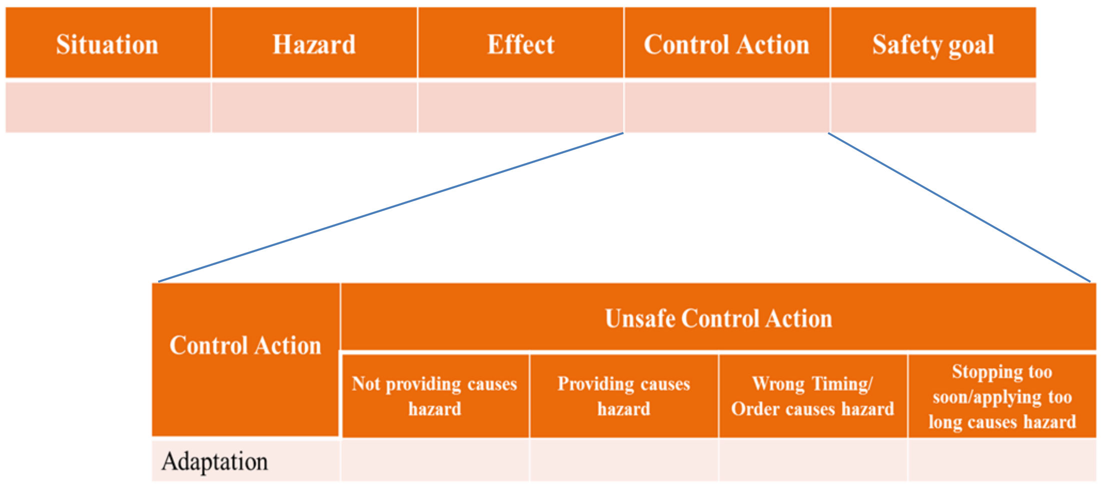
5. **Dynamic Reliability Block Diagrams (DRBDs):** inherit the features of RBD in reliability modelling such as simplicity, versatility and expressive power and extend the formalism allowing taking into account the system dynamics. To this end in DRBD each component can be in three different states: active, i.e., working, failed, i.e., not operational, and stand-by, i.e., reliable but not available. DRBD extend the capabilities of DFTs because of the use of state-space models at the high level model. However, the statespace model is limited to a specific structure (only three states are admitted), thus their application is limited by the formalism itself.
6. **Boolean Driven Markov Process (BDMP**): extends fault trees with two new objects: triggers and Markov chains. Triggers are used to widespread failures of basic events or gates across the tree. To this end triggers carry on a Boolean value that forces BEs to behave differently according to the value of the trigger. In fact, BEs may have two modes. These two modes are represented by two different Markov chains. Only one Markov chain can be active at any time and the one that is selected to be active depends on the value of the trigger related to the BE. Modelling capabilities of BDMP extend the ones of DFT in that is possible to use Petri nets as the leafs of the tree.
7. **Petri Nets**: Petri nets are abstract formal methods, for the description and analysis of flow of information and control in concurrent systems
8. **Stochastic PNs (SPN)** are timed PNs in which all the firing delays are exponentially distributed. The use of exponential distributions for the temporal specifications results in a PN that can be mapped on continuous-time Markov chains

9. **Adaptive Transition Systems (ATS)**: it is an high level modelling formalism that provides a concise and compositional way to describe the behaviour of interdependent reactive systems with general time distributions. Non determinism can be included in the model as well. Different kind of synchronization procedures can be defined, too. In ATS different parts of the system are modelled by different transition systems that are said adaptive because they adapt to each other according to their relative evolution as time progresses.

6.3 STPA: A New Hazard Analysis Technique

Leveson [Leveson 2012] indicates that using the new causality model called Systems-Theoretic Accident Model and Processes (STAMP), changes the emphasis in system safety from preventing failures to enforcing behavioural safety constraints. Moreover, she suggests a new hazard analysis technique called STPA. This method has been used for a preliminary HARA analysis on different domains where complex interactions occur. The main reason to include that is the fact that current hazard analysis techniques such as FTA, Event Tree Analysis and HAZOP do not take in account accident scenarios that encompass the entire accident process, not just the electromechanical components.

As a result, it must be pointed out that the current way HARA is carried is not sufficient for adaptive systems; therefore ideas from what STPA would be a good solution. **Fig.** 26 depicts how control actions are included to the traditional analysis.



Situation	Hazard	Effect	Control Action	Safety goal

Control Action	Unsafe Control Action			
	Not providing causes hazard	Providing causes hazard	Wrong Timing/ Order causes hazard	Stopping too soon/applying too long causes hazard
Adaptation				

Fig. 26 STPA inputs to HARA

Even though, its use on adaptive systems is still quite ambiguous and it needs further research.

6.4 Model-based development

It enables low cost, iterative investigation and early verification and validation being able to re-design before the real implementation is carried out. In the same way, the adaptation concept and its development could be not only verified but also evaluated and re-designed depending on the obtained results.

Model-based development is only considered at software level within ISO 26262 Part 6 Annex B. This means that no references are defined neither at system or hardware level. Some guidelines on how to apply model based design at different development phases would be very helpful. In case of the SAPC adaptation mechanism, the system level needs to be modelled as well. This is the only way to confirm that adaptation works as it is supposed to do.

6.5 Simulation

Implementing safety verification by simulation could be challenging for adaptive systems even if it poses as a really promising solution to achieve early safety verification. An expansion to traditional functional verification, the new fault injection and safety verification technologies are highly required during different phases of the ISO 26262 compliant product development. Fault injection experiments have demonstrated as an effective approach for dependability validation and evaluation.

Taking in account that the design holds software, it could be challenging to validate the safety/dependability of a system where many configurations are possible at a time. To get accurate results by simulation based fault injection, all the possible situations would need to be considered. Some of the most interesting approaches to this issue have been proposed by Benso and Di Carlo [Benso DiCarlo 2011] or Eghbal et al. [Eghbal et al. 2009]. They stated how running different workloads on the same microprocessor changes the observed results. SafeAdapt proposes to adapt running software depending on a set of configurations and operational situations. One of the major drawbacks of implementing this technique in this context is that all possible run time scenarios should be tested at design time to validate safety of the system by means of simulation-based fault injection. This could deal to a time explosion problem.

In the same way, the use of a virtual test driving or simulated vehicle would be beneficial in order to test the perceived vehicle dynamics; this will define the maximum time for SAPC to execute the adaptation, before the driver perceives the risk and the car is out of control. Thus it helps to evaluate and validate the expected fault tolerant time interval at vehicle level. This is especially important for defining the controllability of the vehicle during the adaptation. Furthermore, by defining the appropriate fault models and testing experiments, it would be possible to validate both the Functional Safety Concept and the HARA of the system.

Additionally, Hardware-in-the-Loop (HiL) tests should be executed on a test bench to verify the SAPC functional performance of the adaptation, which ensures that the adaptation behaviour after the software deployment in the real system is the same as expected.

6.6 Outlook

It is planned to conduct a specific review of the requirements at the end of the project after fulfilling development and design work and after the implementation in the demonstrator is completed during the evaluation period. The idea is to check for safety related requirements and their relation to ISO 26262. The approach in SafeAdapt is that in a first instance the safety related requirements will be identified by a review of the requirements document D2.2.

In a second step, these requirements shall be brought into connection with ISO 26262. In case this works for a specific requirement, ISO 26262 is considered sufficient w.r.t. such requirements and it is assumed that the other way round, namely looking at ISO 26262 and base the definition of requirements directly on the ISO would also work sufficiently. If not, a defect or deficit will be identified. In SafeAdapt not necessarily all requirements were defined according to such approach since the work content is also a continuation of work and results conducted and achieved in other projects and even for different domains than the automotive domain targeted in SafeAdapt. The good side of allowing this approach is that it is thus possible to identify potential “holes“ and deficits in ISO 26262, that would not have been visible when strictly following the ISO 26262 standard, only during defining requirements. Within the conduct of the SafeCer and the Safe EC Projects, one of the achievements was that several deficits were detected within ISO 26262 in case one looks at the stringent domain of highly automated and autonomous driving sector and its very strict safety mechanisms needed. Thus SafeAdapt borrowed ideas and concepts from other industrial domains, such as the aerospace domain, using parts of their approaches and basic philosophies. Some of the communication and reconfiguration mechanisms and philosophies behind the ideas used in the SAPC were derived from concepts used in the aerospace domain such as the reconfiguration approach selected.

The idea to make suggestions for the ISO 26262 enhancement can then be based on a final evaluation of the requirements and if they have been implemented successfully within the SafeAdapt project. By checking if ISO 26262 also has foreseen their definition and if so, if the approaches defined in ISO 26262 are sufficient and exhausting to cover the issue under consideration for the specific requirement such study can lead to the desired result. Thus the SafeAdapt team expects to be able to derive valuable examples from the SafeAdapt work at the end of the project and under consideration of the implementations made in order to point at specific areas of ISO 26262 that might need adaptation or enhancement to further guarantee sufficient measures being built-into this standard referring to safety relevant application.

7 Conclusions

As defined at the beginning of the document, different aims have been achieved along this deliverable.

On the one hand, the HARA of some vehicle functions has been presented together with the HARA of the adaptation process. This second step has been used to derive the corresponding adaptation safety goals. At the same time, a sophisticated functional safety concept has been highlighted so all the hazards can be overcome. Thereby, the SAPC is able to handle different adaptation scenarios in a safe way. This follows the direction of creating functional safety concepts that provide fail-operational behaviour at system level for individual functions and in terms of autonomous driving. Even though, this last point is not the main goal in SafeAdapt project.

Moreover, we have introduced how a safety case can be constructed based on the goal structuring notation that proposes to create a defensible argument of goals that are decomposed recursively into subgoals until a subgoal can be proven by evidence. The evidence is provided by documents addressing the safety issues. The standard prescribes that part of the safety case should demonstrate that the origin, realization and proof for every requirement is clearly documented. In other words, all requirements should be traceable to their respective implementations (forward and backward). It is a method to show how to proof the safety requirements have been realized and validated.

On the other hand, in order to promote ISO 26262 compliant adaptive systems, the main challenges have been identified. Due to the immaturity of some of the aspects and the no agreement of the responses, it has not been possible to come up with a clear and common vision/outcome of all of them. However, some of them have been further discussed proposing several contributions to the current status of the standard. It has to be pointed out that some of them have been more deeply analysed, whereas others, such as STPA, have been introduced but not been specified in detail. We believe this work is a good starting point for further research. This future work should consist of trying to close the gap in those aspects. This means that much more effort from both industry and research community should be expended when addressing ISO 26262 compliant adaptive systems.

Bibliography

1. [ISO26262 2011]. International Organization for Standardization (ISO), «ISO/DIS 26262: Road vehicles - functional safety», 2011.
2. [Whittle et al 2009] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel, «Relax: Incorporating uncertainty into the specification of self-adaptive systems», in Requirements Engineering Conference, 2009. RE'09. 17th IEEE International, 2009, pp. 79–88.
3. [Gudemann et al 2006] M. Gudemann, F. Ortmeier, y W. Reif, «Safety and Dependability Analysis of Self-Adaptive Systems», in Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, 2006. ISoLA 2006, 2006, pp. 177-184.
4. [Pop et al 2013] Pop, P., Tsiopoulos, L., Voss, S., Slotosch, O., Ficek, C., Nyman, U., & Ruiz, A. (2013), «Methods and tools for reducing certification costs of mixed-criticality applications on multi-core platforms: the RECOMP approach», In WICERT 2013 Conference Proceedings
5. [Eghbal et al 2009] A. Eghbal, H. Zarandi, P.Yaghini, «Fault tolerance assessment of PIC microcontroller based on fault injection», 2009, 10th Latin American Test Workshop
6. [Alexandersson and Karlsson 2011] R. Alexandersson, J.Karlsson, «Fault injection-based assessment of aspect-oriented implementation of fault tolerance», 2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN), 2011
7. [Arlat et al 1993] Arlat J., Alain Costes, Yves Crouzet, Jean-Claude Laprie, and David Powell, Member, IEEE, « Fault Injection and Dependability Evaluation of Fault-Tolerant Systems», 1993
8. [Mariani et al 2006] R. Mariani, P. Fuhrmann, y B. Vittorelli, «Fault-robust microcontrollers for automotive applications», On-Line Testing Symposium, 2006. IOLTS 2006. 12th IEEE International, 2006, p. 6–pp.
9. [Frtunikj 2014] J. Frtunikj, V. R. (2014). «A safety aware run-time environment for adaptive automotive control systems», Embedded Real-Time Software and Systems, ERTS2.
10. [Ruiz et al 2013] A. Ruiz, H. Espinoza, S. Torchiaro, F. Tagliablò, A Melzi, «A Preliminary Study towards a Quantitative Approach for Compositional Safety Assurance», Assuring the Safety of Systems Proceedings of the Twenty-first Safety-Critical Systems Symposium, Bristol, UK., 2013.
11. [Autosar] AUTOSAR, AUTomotive Open System Architecture, http://www.autosar.org/index.php?p=3&up=0&uup=0&uuup=0 [Accessed: 11-may-2014].
12. [RACE] RACE project, http://www.projekt-race.de/
13. [Butler 2008] R. W. Butler, «A Primer on Architectural Level Fault Tolerance»; Langley Research Center, Hampton, Virginia, 2008
14. [Armbruster 2009] M. Armbruster, «Eine fahrzeugübergreifende X-by-wire Plattform zur Ausführung umfassender Fahr-und Assistenzfunktionen», PhD Thesis, University of Stuttgart, Stuttgart, Germany, 2009
15. [SAFE project D3.3.2] P. Cuenot, «Deliverable D3.3.2 – Requirement specification for a multi-class ECU concept», SAFE project, 2014, http://www.safe-project.eu/SAFE-Publications/SAFE_D3.3.2.pdf
16. [Armoush 2010]A. Armoush, «Design Patterns for Safety-Critical Embedded Systems», PhD thesis, Aachen University, 2010
17. [Laprie 1992] J.C. Laprie, «Dependability: Basic Concepts and Terminology» ,Springer-Verlag, 1992. ISBN 0-387-82296-8
18. [SACM 2014] «SACM Structured Assurance Case Metamodel», OMG Standard, 2014
19. [Kelly 1998]. T.P. Kelly., «Arguing Safety: A Systematic Approach to Managing Safety Cases», PhD Thesis, University of York, UK, September 1998.
20. [Toulmin 1958]. S.E. Toulmin., «The Uses of Argument», Cambridge University Press, 1958.[Kelly McDermid 1997] T. P. Kelly and J. A. McDermid, «Safety case construction and reuse using patterns» in Proc. 16th International Conference on Computer Safety, Reliability and Security (SAFECOMP'97). Springer-Verlag, 1997.
21. [Wagner et al 2010] S.Wagner, B.Schätzy, S. Puchnerz, and Peter Kockx, «A Case Study on Safety Cases in the Automotive Domain: Modules, Patterns, and Models», 2010 IEEE 21st International Symposium on Software Reliability Engineering, 2010.
22. [Hawkins Kelly 2009] R.Hawkins and T.Kelly, «A software safety argument Pattern Catalogue», Technical Report, Department of Computer Science, The University of York, YCS-2013-482, 2009.
23. [Hawkings et al 2011] R.Hawkins, K. Clegg, R. Alexander, and T. Kelly «Using a Software Safety Argument Pattern Catalogue: Two Case Studies», Safecomp '11, Naples, 2011.
24. [EASIS 2006]. O. Bridal and others. Deliverable D3.1 Part 1 Appendix E: Safety Case, Version1.1. Technical Report, EASIS Consortium (www.easis-online.org), February 2006.
25. [Mouaffo el al. 2013] Adrien Mouaffo, Kavyashree Jamboti, Davide Taibi, «Empirical Evaluation of State Event Fault Tree and Fault Tree combined with Markov Chains for the Safety Analysis of Dynamic Embedded Systems», In , pp. 36, 2013.

26. [Kaiser el al 2007] Bernhard Kaiser, Catharina Gramlich, Marc Förster, «State-Event Fault Trees - A Safety Analysis Model for Software Controlled Systems», Journal: Reliability Engineering & System Safety - RELIAB ENG SYST SAFETY , vol. 92, no. 11, pp. 1521-1537, 2007
27. [Leveson 2012] N. G. Leveson, «Engineering a Safer World: Systems Thinking Applied to Safety», Cambridge, Mass.: The MIT Press, 2012.
28. [Esposito et al. 2011] C Esposito, D Cotroneo, N Silva, «Investigation on safety-related standards for critical systems», Software Certification (WoSoCER), 2011 First International Workshop on, 49-54
29. [Storey 1999] N. Storey, «Design for Safety. Towards System Safety», Proc. 7th Safety-Critical Systems Symposium, Huntington, UK. Feb. 1999, 1-25, 1999
30. [Manno 2011] G. Manno, «Reliability modelling of complex systems: an Adaptive Transition System approach to match accuracy and efficiency», PhD Thesis (2011).
31. [Essarel] ESSaRel project, http://www.essarel.de
32. [Benso DiCarlo 2011] A. Benso, S. DiCarlo, «The art of fault injection», in J. of Control Engineering and Applied Informatics, vol. 13, 2011, pp. 9–18.

List of Abbreviations

Abbreviation	Definition
AGG	Aggregate
ASIL	Automotive Safety Integrity Level
CCC	central ICT computing core
CDD	Complex Device Driver
CSCC (or Core Node)	central ICT sub computing cores
DC	Diagnostic Coverage
ECU	Electronic Control Unit
E/E	Electrical or/and Electronics
FTTI	Fault Tolerant Time Interval
GSN	Global Structuring Notation
GW	Gateway
HARA	Hazard and Risk Analysis
HiL	Hardware-in-the-Loop
HSI	Hardware-Software Interface
HW	Hardware
HV	Health Vector
FEV	Fully Electric Vehicles
FTA	Fault Tree Analysis
FMEA	Failure Mode and Effects Analysis
GW	Gateway
HARA	Hazard and Risk Analysis
ICT	Information and Communication Technology We consider a vehicle‘s ICT, which contains [1..2] low-voltage power distribution networks [1] high voltage power distribution networks [1] central ICT computing core (CCC) [1..3] central ICT sub computing cores (Core Nodes or CSCC) that are delivered from DELPHI, SIE [1..N] aggregates (AGG) / network nodes besides the ICT computing core (CCC) which (a) provide data and/ or (b) receive commands from the CCC. [0..R] gateways (GW) to interlink different communication links which are design using different communication technologies (CAN, FlexRay, Eth, …) [1] central safe adaptation platform core (SAPC) [1..2] safe adaptation sub platform cores (SASPC) which are implemented within each central ICT sub computing cores that are delivered from DELPHI, SIE [1..M] automotive functions that are realized via the ICT (and considered within the HARA) [1..K] applications, that are implemented on the CCC [1..L] applications, that are implemented on an aggregate AGG [1..X] communication links
re-allocation	Re-allocation contains the installation of binaries on a vehicle control-computer including the configuration of the resource-management within the OS and the middleware.
re-activation	Re-Activation does not include the reassignment of resources This aspect is considered within the re-allocation. Re-Activation just describes the activation and passivation of applications within the schedule (means switching on and off of applications).
SAPC	Safe Adaptation Platform Core
SASPC	Safe Adaptation Sub Platform Cores

SEooC	Safety Element Out Of Context
STAMP	Systems-Theoretic Accident Model and Processes
SW	Software
VCC/ vehicle control-computer	Electronic control-computer within domain-model
WD	Watchdog