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 5.3

Evaluation Results of the specified Use Cases and Scenarios

Due date of deliverable:	30.06.2016
Actual submission Date:	30.06.2016
Lead beneficiary for this deliverable:	Duracar

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	Evaluation Results of the Specified Use Cases and Scenarios
Creator	Duracar: Ken Lam
Description	The document contains a detailed description of the evaluation of the SafeAdapt results of the specified use cases and scenarios according to the evaluation methodologies described in D5.1. The improvements and advantages/disadvantages gained by SafeAdapt are shown in this deliverable.
Publisher	Members of the SafeAdapt Consortium
Contributors	Fraunhofer: Philipp Schleiss, Christian Drabek, Gereon Weiss TTTech: Andreas Eckel Ficosa: Andrea Saccagno Tecnalia: Garazi Juez, Alejandra Ruiz, Mª Carmen Palacios, Maite Alvarez, Maite Alvarez, Josu Albizu CEA: Ansgar Rademacher, Mahmoud Hussein Siemens: Cornel Klein, Jan Sawallisch, Andre Marek, Marc Zeller Pininfarina: Sandro Morero Duracar: Ken Lam Delphi: Thorsten Rosenthal
Language	en-GB
Creation date	27.01.2016
Version number	1.0
Version date	30.06.2016
Audience	☐ internal ☒ public ☐ restricted

Table of Contents

List of Figures

List of Tables

Executive Summary

The focus of this document is to evaluate the results of the SafeAdapt project with respect to advantages and disadvantages of the proposed methods. For this, the results of the project are first evaluated for general feasibility and then further assessed based on measurable indicators. These measurable results were initially described in the Description of Work (DoW) and then further outlined in Deliverable 5.1, which explained the methods used to attain these figures.

The evaluation in this document clearly shows that SafeAdapt's concept for an adaptive E/E-architecture is a viable option for future vehicles in general, and is moreover applicable for highly safety-critical functions. In addition, the measured results highlight significant performance improvements with respect to weight, cost, complexity, redundancy, amount of handled failures, certification effort, and energy efficiency.

In sum, SafeAdapt's adaptive and fail-operation E/E-architecture outperforms current state-of-the-art technology in all categories of interest, and thus, provides a sound foundation for the future development of safe and efficient vehicles, especially with regard to fully electric and autonomous cars.

1 Introduction

At the early stage of this project, the use cases and requirements and their specific scenarios had been defined for the safe adaptation of safety-relevant systems in Fully Electric Vehicles (FEVs). Subsequently, the development, the design, the support tool chain, and the implementation of the Safe Adaptation Platform Core (SAPC), which enforces the safe adaptation of networked embedded systems in FEVs, were realised. In order to examine the SafeAdapt approach in an objective way, the evaluation methods were defined in detail before implementing the prototypes in Deliverable D5.1 [1]. These methods in turn, were also mapped onto use cases and scenarios defined during the requirement engineering phase of the project, previously documented in Deliverable D2.1 [2].

Based on these use cases, different prototypes, such as a full-scale electric prototype vehicle, scale-model vehicle, and driver-in-the-loop simulators, were chosen to demonstrate the main aspect each use case intends to address. In sum, these aspects reach from proving runtime-efficient fail-operational E/E-architectures up to energy optimisation topics.

The main subject of this deliverable is the evaluation of the SafeAdapt results. In order to exploit these results, the evaluation of both, the viability and the efficiency, of the SafeAdapt approach have to be taken into account. As a starting point, the overall goals of the project are summarized in Chapter 2. In Chapter 3, the viability of SafeAdapt is regarded. This involves an overview of the prototypes developed during the project and the corresponding use cases. Chapter 4 evaluates the supporting tools and development processes. The content of Chapter 5 is the evaluation of the efficiency of SafeAdapt. Therein, the measurable results such as energy consumption, cost reduction, or reduced complexity are examined. Finally, the last chapter contains the summary and conclusion of this document.

2 Goals

The goal of this document is to provide an evaluation of the SafeAdapt project's results. The evaluation is an integral part of the project plan of SafeAdapt. Therefore the evaluation methods have been well defined in D5.1 [1]. As the project goals are the guideline for the evaluation, they are summarised here in short.

The project is driven by both, use cases and predefined targets, with respect to enhanced safety and reductions in complexity as well as unit and development cost. Consequently, the goals are also two-fold:

The first part of the documents shows the SafeAdapt approach is viable and can be used to fulfil the functional aspects of the project's objectives, i.e., the approach is effective:

- Objective #1: Provide novel architecture concepts to enhance robustness, availability, and efficiency of safety-relevant systems while preserving the functional safety in FEVs
- Objective #2: Increase safety and availability through the ability to handle complex failures, especially failures where current systems do not degrade gracefully
- Objective #3: Reduced bill of material by reducing the number of ECUs by providing a generic failure management based on the SafeAdapt Platform Core
- Objective #4: Reduced development costs (time-to-market & testing costs) in future FEVs by providing a generic failure management and software update mechanism (dealer retrofit) based on a SafeAdapt Platform Core
- Objective #5: Increased energy efficiency in automotive E/E-architectures

The second part evaluates if the effort required to use SafeAdapt's approach is manageable and can even reduce the effort compared to other approaches, i.e., the approach is efficient. This will be evaluated by comparison of the following measurable results (MR):

- MR1: Optimise energy consumption of safety-relevant features by up to 30%
- MR2: Handle 20-30% of failures in safety-relevant systems by adaptation or reconfiguration
- MR3: Reduce development and testing costs by up to 20%
- MR4: Reduce certification cost by up to 20%
- MR5: Reduce complexity and hardware cost of safety-relevant systems by up to 20%
- MR6: Improve current redundancy concepts (duplication of ECUs) by 50%
  (require less extra ECUs while meeting the redundancy concept requirements)

With this two-fold evaluation it is targeted to show that SafeAdapt's approach for safe adaptation is both viable and efficient.

3 Evaluation of Technical Feasibility

3.1 Demonstrator Platforms

As described in Deliverable D5.1 [1], SafeAdapt utilised multiple demonstrators to showcase the viability of handling different types of use cases (see D2.1 [2]) in a safe and efficient manner. The following Table 1 provides a brief mapping between these use cases and the demonstrator prototypes. In addition, the key characteristics of each use case are carved out, thereby, describing a set of use cases that can be handled by the same SafeAdapt mechanism.

Use Case	Demonstrator	Key Characteristics
UC_110_01: Reconfiguration of Failed Cruise Control	AUTOSAR Demonstrator	Backup instances is provided in **cold-standby** manner, allowing high resource efficiency
UC_111_01: Steer-by-Wire Adaptation after ECU-Failure	RACE & AUTOSAR Demonstrator	**Failure of a single Core Node** does not affect the safety of the system
UC_114_01: Adaptation after Brake-by-Wire Malfunction	RACE & Frozen-Standby Demonstrator & AUTOSAR Demonstrator	As resources are limited after a failure, the system uses **graceful degradation** to keep all functionalities with fail-operational requirements active
UC_116_01: Communication Failure with External Aggregate	RACE	A single **failure of a communication link** does not affect the safety of the vehicle
UC_211_01: Installation of New Component	RACE (SafeCar)	The system is able to revaluate its dependability whenever its **hardware architecture changes**
UC_311_01: Update of Function	RACE (SafeCar)	The system can revaluate if its safety requirements are still met when **software is exchanged**
UC_411_01, Degradation of Steer-by-Wire Application	Dynacar	The **failover time** of a functionality must remain within the limits given by physical properties of the respective functionality (e.g., the maximal time span the steering is allowed to not be under control without affecting the performance of driving)
UC_511_01, Adaptation for Range Extension	Dynacar	The system is capable of adapting functionality to improve **energy efficiency**

Table 1: Mapping of use cases to demonstrators

3.2 Full-Scale E-Vehicle Prototype

3.2.1 Platform Description

For evaluation of the SafeAdapt results, the RACE car (i.e., the SafeAdapt full-scale e-vehicle prototype) was used as already described in Deliverable D5.1 [1], in Chapter 3.2. The goal of this demonstrator is to prove the feasibility of SafeAdapt's safety concepts in a full-scale electric vehicle, with special focus on guaranteeing the availability of a highly critical driving functionality.

Figure 1: SafeAdapt car with components TMDP (black), RACE DCC (silver), and switches (blue)

Figure 1 shows the car with corresponding adaptations. For demonstrating the fail-operational capabilities of the advanced SafeAdapt architecture, the Steer-by-Wire functionality was chosen, as it requires fail-operational behaviour. For this, the e-vehicle is hoisted with a hydraulic lift in the laboratory garage to allow the wheels to move freely. Moreover, the system also hosts a fail-operational Brake-by-Wire and a non-critical SomnoAlert application, with the application deployment and IDs of each instance depicted below in Table 2:

TMDP	RACE DCC
1st application with ID "1": ***Steer-by-Wire***	1st application with ID "4": ***Steer-by-Wire***
2nd application with ID "2": ***Brake-by-Wire***	2nd application with ID "5": ***Brake-by-Wire***
3rd application with ID "3": ***SomnoAlert***	---

Table 2: Instance ID of applications

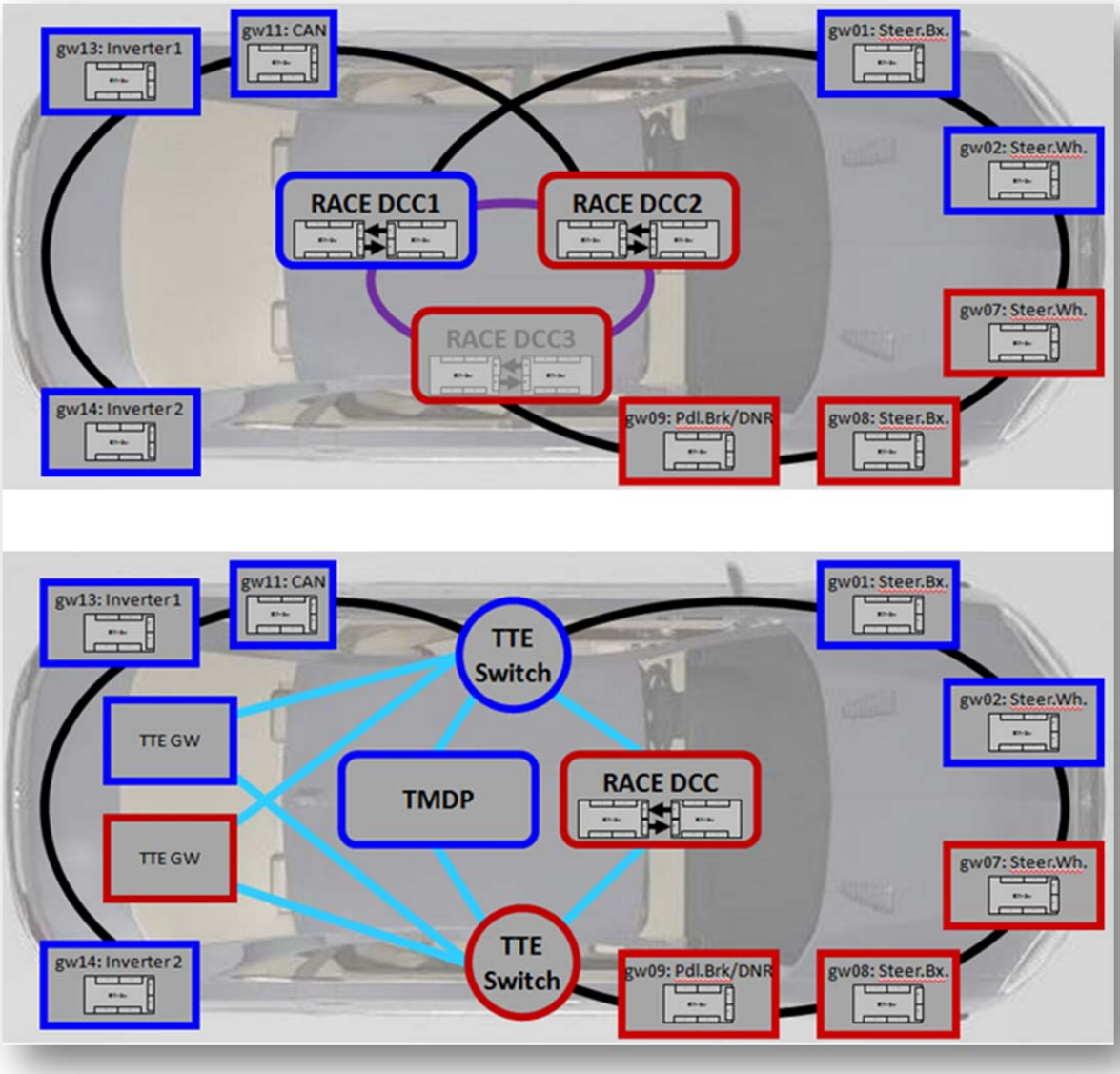


Figure 2: Ring topology (top: original layout; bottom: improved SafeAdapt architecture)

Figure 2 depicts the topology of the main aggregates. The outer ring with gateways (sensors and actuators) was not further modified. The inner ring was originally designed with three RACE DCCs for the former RACE demonstrator car. This part was upgraded with a combination of a TMDP and a RACE DCC, with additional Time-Triggered Ethernet switches.

To monitor and manipulate all components in the architecture separately, a test system was used (see Figure 3). Here, especially the continuous steering wheel motion („SteeringWheel - Istwert“) and the resulting correction of wheel angle at the steering box („SteeringBox - Sollwert“) proved the complete data flow from sensor via application and the Core Nodes (CCC) to an actor (see bottom part of Figure 3). Moreover, a systematic set of fault injection test cases were performed with this tooling. The impact of application failure, Core Node failure, and power failure tests are described in the following.

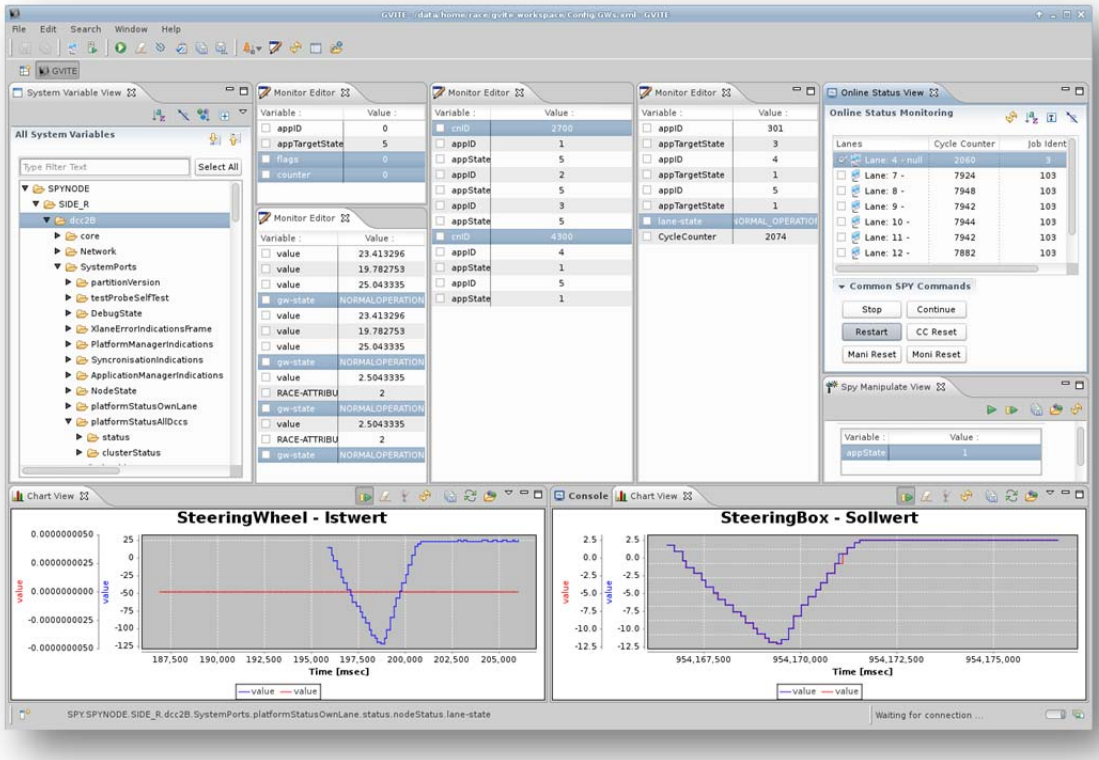


Figure 3: Test system for monitoring all components

In addition, the Health Vector exchanged between the Core Nodes can also be monitored, to trace each ECU state and verify that the required adaptations were performed correctly after a failure has been injected. Figure 4 shows this monitoring solution, which allows the identification of the Core Node, the state of adaptation, and each application's individual mode of operation.

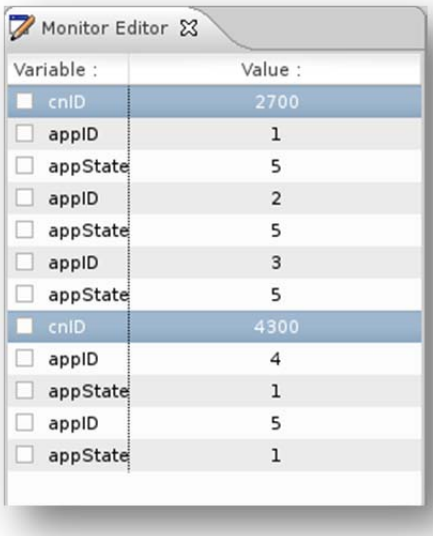


Figure 4: Health Vector monitoring system

During normal mode of operation, both ECUs are operating in an adaptation mode with ID 50 (“adaptID”). In this system mode, the TMDP is configured to supply the SomnoAlert functionality whereas the RACE DCC provides Steer- and Brake-by-Wire functionality, as shown below:

TMDP	RACE DCC
appID “1”: **Steer-by-Wire** *HOTSTANDBY*	appID “4”: **Steer-by-Wire** *ACTIVE*
appID “2”: **Brake-by-Wire** *COLDSTANDBY*	appID “5”: **Brake-by-Wire** *ACTIVE*
appID “3”: **SomnoAlert** *ACTIVE*	---
Adaptation ID “50”	

Table 3: Normal system mode

3.2.2 Failure of Single Application

To simulate the failure of a single application, the test system was used to manipulate the operating state of the Steer-by-Wire (SbW) application through the RTE (Runtime Environment) of the RACE DCC, thus, passivating the application. This simulated failure of the application is then automatically communicated through the Health Vector to the TMDP platform and to the monitoring system. As soon as the SAPC algorithm is executed again, both platforms simultaneously transition into a new mode, forcing the TMDP to take over the control of the SbW application and thus, compensating for the application’s failure. The new operating mode is displayed below in Table 4:

TMDP	RACE DCC
appID “1”: **Steer-by-Wire** *ACTIVE*	appID “4”: **Steer-by-Wire** *DEACTIVATED*
appID “2”: **Brake-by-Wire** *COLDSTANDBY*	appID “5”: **Brake-by-Wire** *ACTIVE*
appID “3”: **SomnoAlert** *ACTIVE*	---
Adaptation ID “201”	

Table 4: Failure of Steer-by-Wire on RACE DCC

The practical test showed the uninterruptible steering by the continuously lateral motion of the steering wheel. The test system displayed the reactivation of the SbW application on the TMDP platform. Similar to deactivating the SbW application on the RACE DCC (described above), it was also possible to deactivate the *Brake-by-Wire application* on the RACE DCC.

3.2.3 Failure of Core Node

To simulate the failure of an entire Core Node, electric switches were integrated into both platforms to allow an easy interruption of the power supply for a specific ECU. Based on this, it was possible to simulate a complete failure of a Core Node, which consequently led to the suppression of Health Vector transmission. In this test case, the power of the RACE DCC was interrupted, so the TMDP assumed that all RACE DCC applications are in a DEACTIVATED state. The SAPC consequently reacted with the corresponding adaptation (adaptID “300”) allowing the TMDP to take over the critical steering and braking functionality, while deactivating the non-critical SomnoAlert application.

TMDP	RACE DCC
appID “1”: **Steer-by-Wire** *ACTIVE*	appID “4”: **Steer-by-Wire** *DEACTIVATED*
appID “2”: **Brake-by-Wire** *ACTIVE*	appID “5”: **Brake-by-Wire** *DEACTIVATED*
appID “3”: **SomnoAlert** *DEACTIVATED*	---
Adaptation ID “300”	

Table 5: Failure of ECU

The practical test once again showed the uninterruptible steering by the continuously lateral motion of the steering wheel. Analogously, a failure of the TDMP also led to a correct adaptation of the RACE DCC. To protect against undefined system states through ECUs re-joining the SafeAdapt network after a failure or ECU reset, additional tests were performed by repowering a failed ECU after an adaptation occurred. As expected, the newly powered ECU was prevented from joining the network, thereby, ensuring a consistent system state.

3.2.4 Failure of Power Source

The RACE e-vehicle prototype is equipped with two independent power sources. For an easier understanding the two integrated electric circuits were called “blue” and “red” (see Figure 2). A separate power switch for each circuit is implemented in the vehicle. Therefore, it was possible to shut down the whole “blue” or “red” power supply. In this case, not only the relevant Core Node failed but also the half of all redundant sensors and actors. Figure 5 displays both failure scenarios.

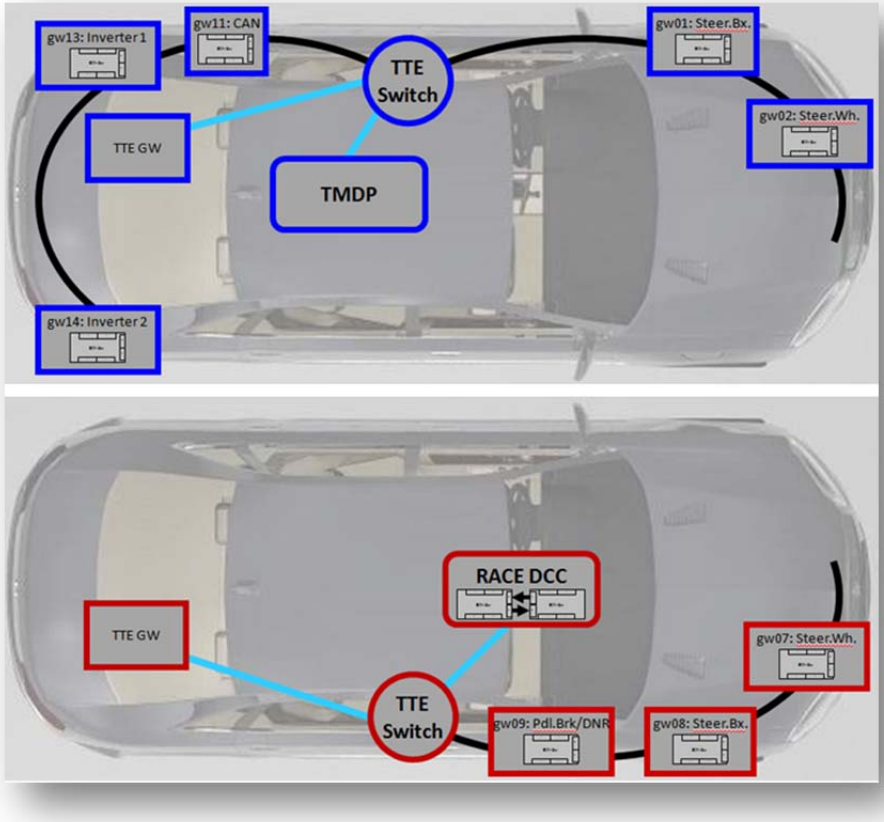


Figure 5: Failure of “red” and “blue” power source

Despite this severe failure, the steering functionality could still be supplied without any drawbacks with respect to availability.

In sum, the practical tests showed the uninterruptible steering with continuously lateral motion of the steering wheel during all three failure test cases. Thereby, the soundness and safety of SafeAdapt's adaption method could be underlined.

3.2.5 Use Case concerned with Plug'n'Play and HW/SW-Updates

Installation of New Component

The goal of the SafeCar prototype is to verify that new components (SW or HW) can be integrated into the vehicle, while the safety requirements of the overall system are preserved (UC_211_01). Therefore, a compositional, deductive safety analysis in form of an FTA (as recommended by ISO 26262) is performed to evaluate that the safety requirements are still met after the installation of the new component. The SafeCar, as a conceptual representation of the RACE car, is a radio controlled demonstrator vehicle as depicted in Figure 6. For further details, please refer to Deliverable D5.1 [1].

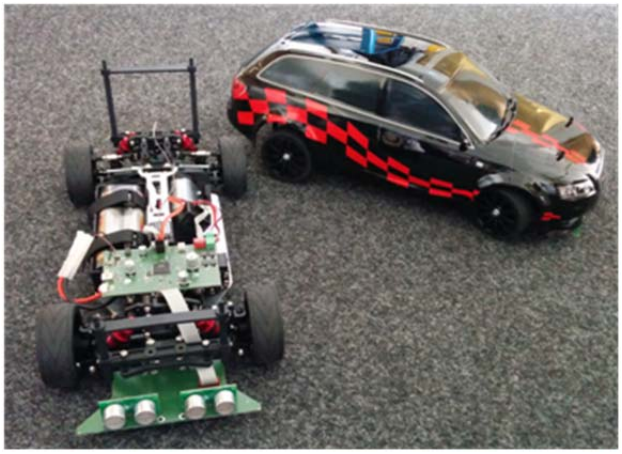


Figure 6: SafeCar demonstrator platform (see Deliverable D5.1)

The case study UC_211_01 "Installation of New Component (New)" is implemented by providing a system architecture model in form of a SysML internal block diagram (IBD) to describe the system architecture of the emergency braking function.

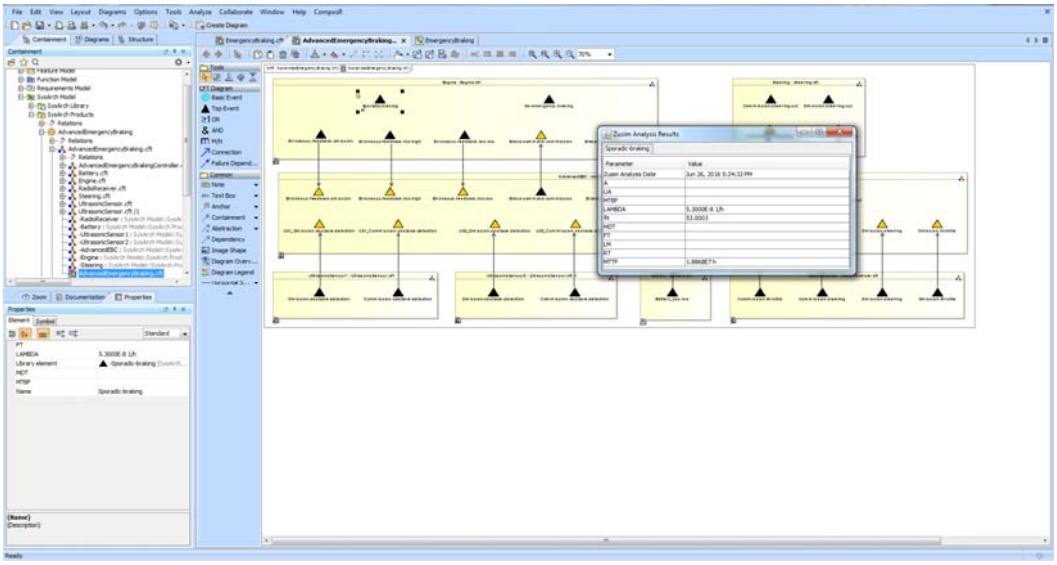


Figure 7: Component Fault Tree Analysis tool composR

In order to evaluate whether the systems' safety requirements are fulfilled, a compositional safety analysis model of the emergency braking function is provided in form of a Component Fault Tree (CFT) using the composR tool. composR is a tool for building CFTs and enabling qualitative and quantitative FTA based on the CFT models (see screenshot in Figure 7). Moreover, composR allows the integration and synchronisation of any system design modelling approach (e.g., SysML, EAST-ADL, UML, etc.) with CFT methodology. Hence, such a compositional and model-based development strategy enables deductive safety analyses of the system in a qualitative as well as a quantitative manner (cf. Figure 8).

The integration of a new component is demonstrated by adding it to the system design model. The safety analysis model is adjusted accordingly in an automated manner by adding the CFT element of the new component (e.g., from a repository) and integrating it into the existing CFT model.

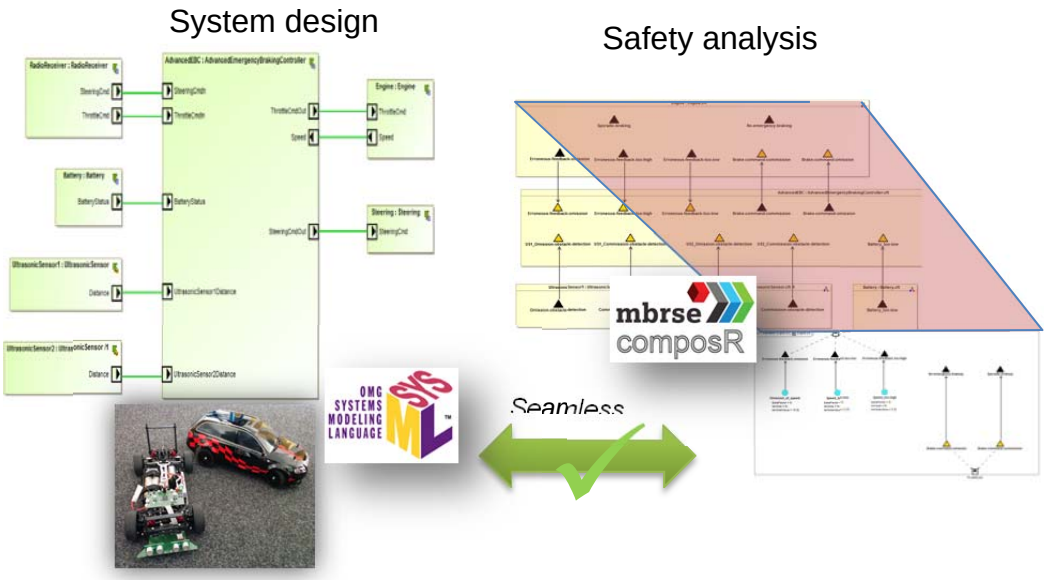


Figure 8: Compositional, model-based development strategy (UC_211_01)

For instance, it is assumed that the emergence braking functionality of the SafeCar demonstrator is realised using one single ultrasonic sensor to detect obstacles in front of the vehicle (variant with 1 sensor). The result of the CFT-based FTA for this architectural realisation of the functionality is shown in Table 14. composR allows the designer to add a second ultrasonic sensor (homogenous redundancy) to the system design model. Now, the CFT is adjusted to the new system design to automatically validate the system in terms of safety (see results in Table 9). Finally, the results of the safety analysis before and after a new component is integrated are compared to check whether the system still meets its predefined safety requirements. By adding a second redundant sensor as input for the emergency braking functionality, the failure probability for the safety relevant hazard (the emergency braking fails although an obstacle is detected) decreases. The reliability relevant failure (the emergency braking is triggered sporadically) is increased by the introduction of a second sensor to the system design. In this evaluation scenario, it was assumed that stopping the vehicle is the safe state.

Top Event	Probability
Omission of Braking (variant with 1 sensor)	$1.0\ 10^{-8}$ 1/h
Sporadic Braking (variant with 1 sensor)	$4.5\ 10^{-8}$ 1/h
Omission of Braking (variant with 2 sensors)	$6.92\ 10^{-9}$ 1/h
Sporadic Braking (variant with 2 sensors)	$5.3\ 10^{-8}$ 1/h

Table 9: Results of the quantitative CFT-based FTA for the SafeCar (UC_211_01)

Update of Function

The use case UC_311_01, “Update of Function (Update)” has been selected to demonstrate how new software components can replace existing ones, while the vehicle is in the field and the safety requirements of the overall system are preserved.

In this use case, the driver wants to upgrade his vehicle by installing new software at an official maintenance service provider. Once in the garage, the maintenance service provider proceeds to perform all required overhaul operations. The goal is to demonstrate that a new version of a software component can replace an existing one, while the safety requirements are preserved. Therefore, a safety analysis is performed to evaluate that the requirements are still met after the update of a software component.

The case study UC_311_01, “Update of Function (Update)” is implemented in the same way as use case UC_211_01 “Installation of New Component (New)”, with the difference that an existing component is exchanged by a different version. Thereby, the composR tool allows the automatic adjustment of the CFT model, when a component within the system model is replaced by a different one. Moreover, since deployment relations of software to hardware can be represented within the CFT methodology using so-called failure dependencies [3], changes of the deployment of software functions to ECUs can also be evaluated automatically in terms of safety, using composR.

For evaluation, it is assumed that the emergence braking controller (EBC) of the SafeCar demonstrator is available in two different variants. In variant 1, the EBC stops the vehicle if an obstacle is detected within a fixed, predefined distance in front of the car. In variant 2, this distance is depending on the current speed of the vehicle. With increasing speed of the car, the distance is also increasing at which the vehicle starts emergency braking, in case an obstacle is detected.

The results of the CFT-based FTA for the realisation of the emergency braking functionality with both variants of the EBC are shown in Table 10. Moreover the results of the safety analyses before and after the update of an existing component are compared, to check whether the system still meets its predefined safety requirements. The results clearly show that the failure rate of the hazard (the emergency braking fails) is higher in variant 2 (the more complex one) than in variant 1. The failure rate for the hazard (emergency braking is triggered sporadically) is the same in both variants of the EBC.

Top Event	Probability
Omission of Braking (variant 1)	$5.33\ 10^{-9}$ 1/h
Sporadic Braking (variant 1)	$5.3\ 10^{-8}$ 1/h
Omission of Braking (variant 2)	$6.92\ 10^{-9}$ 1/h
Sporadic Braking (variant 2)	$5.3\ 10^{-8}$ 1/h

Table 10: Quantitative CFT-based FTA for the SafeCar (UC_311_01)

3.3 Dynacar Demonstrator

In the context of SafeAdapt, Dynacar was used to analyse two main scenarios to show that the SafeAdapt approach is viable. The first use case is related to “Degradation of Steer-by-Wire Application” (see UC_411_01) and estimates the failover time for a Steer-by-Wire application. The failover time is defined as the maximum time the vehicle can operate safely without control. This use case was analysed with a Driver-in-the-Loop (DiL) approach, since the driving performance effect is mainly dependent on the user characteristics (age, skills, experience, etc.).

The second scenario is related to the “Adaptation for Range Extension” (as defined in the UC_511_01). It analyses the possible range extension by increasing the energy efficiency through adaptation.

3.3.1 Platform Description

The Dynacar platform is a real-time vehicle dynamics simulation software solution based on:

- Real-time testing platform software (NI Veristand® real-time framework)
- Graphic visualisation system and vehicle control for real test driving in a virtual environment
- Full vehicle dynamics model running on real-time equipment (PXI hardware)
- Full Hardware-in-the Loop and Model-in-the Loop configurable capabilities

Dynacar gives the possibility to run hardware or software against a vehicle dynamic model to test the vehicle dynamics behaviour, so it allows the mixing of virtual or real ECUs, vehicle sensors and vehicle control variables are used to change the behaviour of the vehicle in real-time. A detailed description of Dynacar’s HW and SW architecture is given in Deliverable D5.1 [1]. In Figure 11, a schematic of the Driver-in-the-Loop simulator is presented.

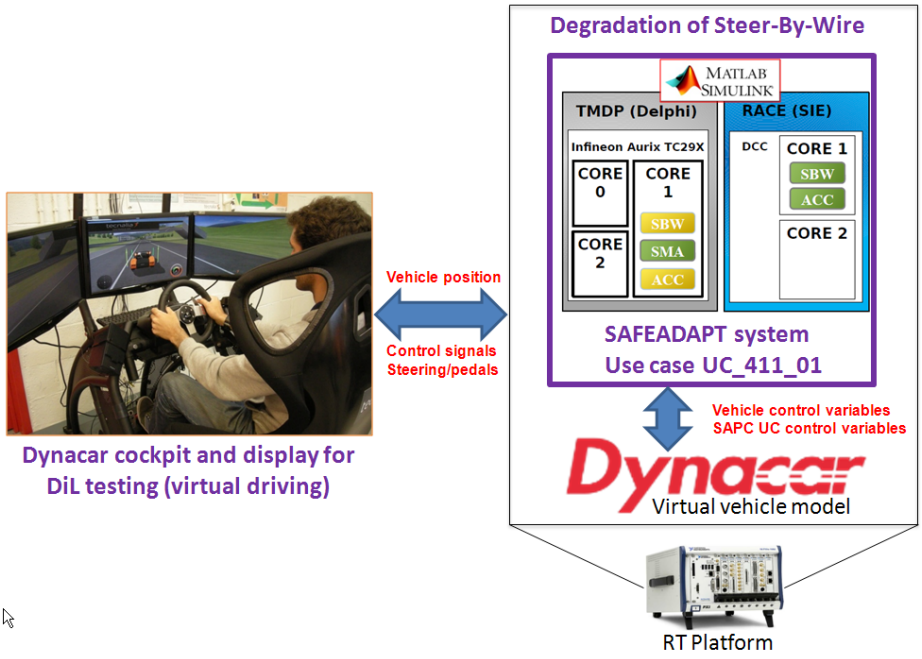


Figure 11: Driver-in-the-Loop Simulator

Moreover, optimisation with respect to energy efficiency could be safely evaluated in this driver-in-the-loop simulation. Therefore, the drowsiness detection SomnoAlert has been integrated as an exemplary non safety-critical application, which can be shutdown to save energy consumption and thus, leading to an increased range of an e-vehicle.

The screen shown in Figure 12 gives the possibility to control the state of different use cases. It activates the fault trigger for the Steer-by-Wire performance degradation analysis, and it also activates the energy efficiency cases. Finally, the Dynacar GUI allows the possibility to configure the autonomous driving mode to compare the energy efficiency between different configurations, without the need of manual driving control. This facilitates the comparison of energy consumptions.

Figure 12: SafeAdapt use case monitor interface

3.3.2 Use Case concerned with Maximal Failover Times

The use case UC_411_01, “Degradation of Steer-by-Wire Application” was selected as representative of a critical use case, where a critical system for the vehicle safety is recovered by adaptation. This use case is utilised to estimate the admissible failover time. The Dynacar demonstrator is used to measure the maximum response time of the system without any safety deviation perceived by the driver. Figure 13 shows the steps for the test case scenario definition.

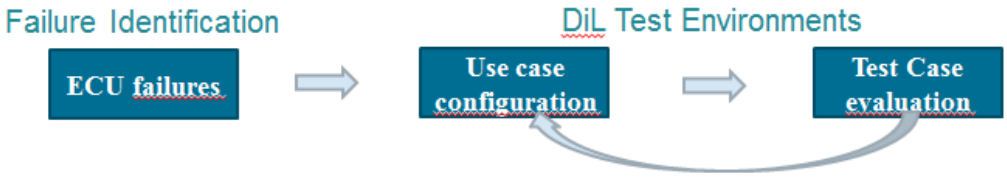


Figure 13: Steps followed for test case scenario definition

The test case definition was started by analysing possible ECU failure propagation and studying the hypothetical impact in the safety performance of the steering system. This implies that the possible HW/SW faults could lead to specific errors that can (or not) propagate to the system e.g., cyclic redundancy check (CRC) validation to avoid incorrect functional set point selections, like the steering angle. After these steps, 2 fault propagation cases have been identified:

- *F3*: Core Node failure or SAPC adaptation time being too long for safe system recovery, causing a continuously cached steering set point (“frozen” SbW). As a consequence, the main aspect when evaluating its impact on SbW is the SAPC adaptation time.
- *F4*: Power failure, e.g., due to inverter malfunction, in the active steering mechanism causing that vehicle to lose steering angle due to self-centring effect on the wheels.

The test environments (TE) must evaluate the impact on the steering performance. Therefore, the selected scenarios are those where SbW performance degradation has the biggest impact:

- *TE1*: Urban environment (40/50 km/h). These cases comprise closed curves where the driver must change the direction of the vehicle and so the driving operation has a high steering demand. This only occurs for urban like roads and the selected speeds are medium/high for these environments.
- *TE2*: Highway driving (150/175 km/h). These cases comprise high speed scenarios during highway driving with steering demand. The main reason for their selection is that at high speeds any small degradation in SbW response could be safety-critical. In this scenario, the high steering demand is caused by a reverse curve and a constant radius curve.
- *TE3*: Sporty driving with high steering demand and driving as fast as possible.

For each test case that evaluates the impact of a SbW degradation, the following parameters have been defined:

- Target vehicle speed. Target speed for the driver should try to maintain along the test.
- Fault duration time. Time interval for which the steering system is performing in one of the 2 defined degraded states (50, 150, 250 and 400 ms).
- Fault type. The degradation in the SbW (Steer-by-Wire) is simulated by taking 2 possible fault cases: frozen steering state (*F3*) for a predefined amount of time and torque-less steering state (*F4*) where the momentum generated by the road centres the steering.

It is important to note that the test case scenarios itself have been completely specified with the previous steps, but there has been an additional “tuning” step to select valuable fault configurations to avoid the execution of test cases that do not yield any extra information. In particular, a prior

non-accurate failover time (near 200 ms) has been estimated with only 3 drivers and then the test case failure times have been selected by taking this preliminary result into account.

These fault injection experiments are configured based on the test evaluator in the Dynacar HMI, defined in Veristand (cf. 2.4 in D4.3). This is carried out by triggering a fault when the vehicle passes a specific position in the driving scenario, e.g., the two cones that represent the line of the trigger shown in Figure 14.

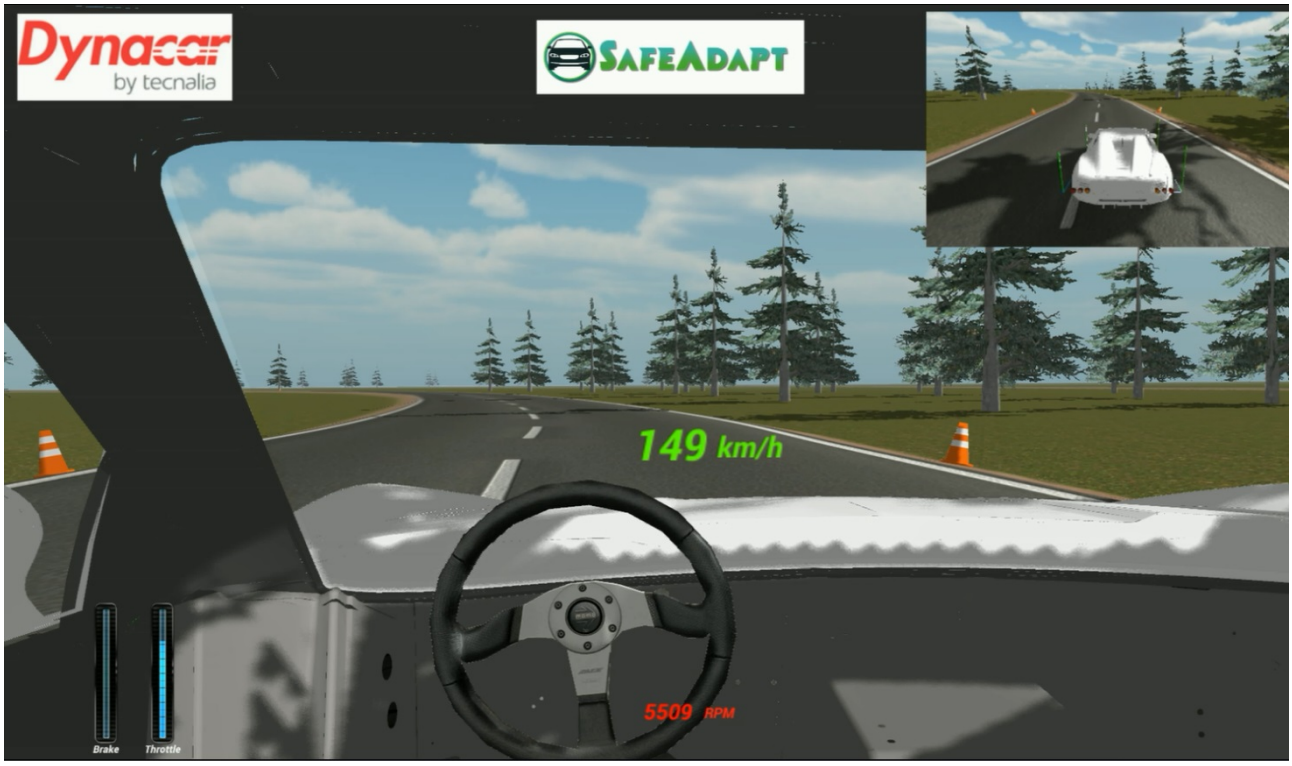


Figure 14: Vehicle approaching cones that represent the road intersection for the fault trigger

For a correct driver sample that reflects the drivers' responses well, a sample of 25 volunteers with the following distribution has been used to perform the tests. The selection followed a realistic driver distribution (shown in Figure 15) so that they can be generalised for a European driver distribution context:

- 5 drivers with < 6 years of experience
- 4 drivers with experience between 6-10
- 3 drivers with each group between 11-15, 16-20 and 21-25 years of experience
- 2 drivers with each group between 26-30, 31-35 and 36-40 years of experience

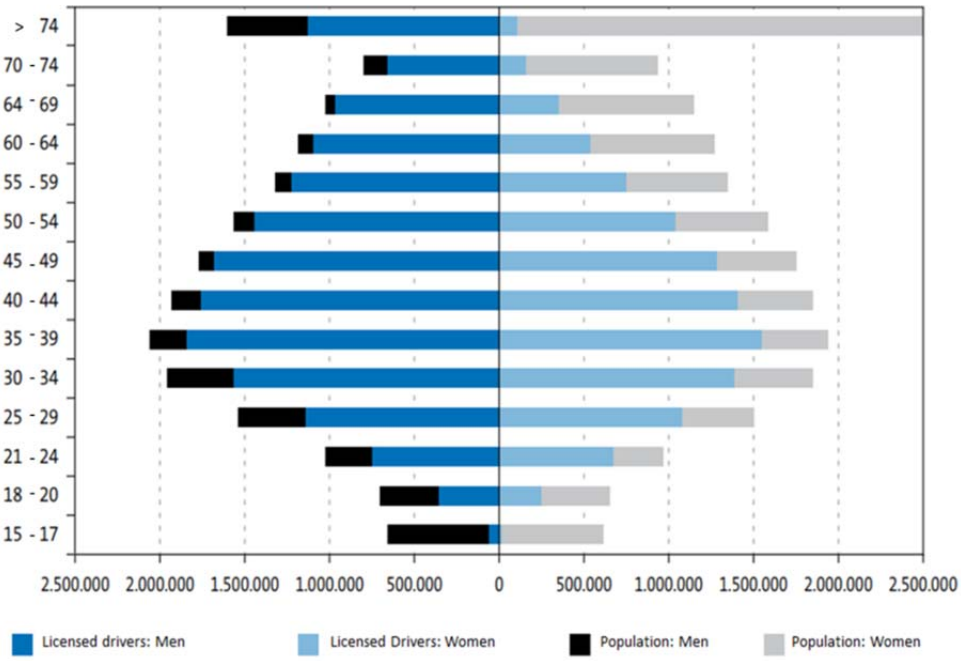


Figure 15: Driver distribution by gender and age in Spain (Source: Dirección General de Tráfico)

For the evaluation of each test case, the following outputs were captured:

- Change of the yaw rate (defined as the ratio between the yaw angle or vehicle orientation in the lane with the time interval duration, what is equivalent to linear speed divided with steering radius). It is an objective parameter to evaluate how the faulty behaviour influences the driving or the controllability aspect.
- Deviation from the lane centre, as another parameter for an objective test evaluation.
- Driver subjective perception. After each test, the drivers were asked to answer if they detected performance degradation. The answers were then processed to calculate the mean perception for each test case given in Table 6.

Environment, Fault Type, Driving Speed	**Simulated Fault Duration**			
	50 ms	**150 ms**	**250 ms**	**400 ms**
TE1, F3, 40 Km/h	1.0	1.1	1.1	1.4
TE1, F3, 50 Km/h	1.0	1.1	1.3	1.6
TE1, F4, 40 Km/h	1.0	1.1	1.5	2.4
TE1, F4, 50 Km/h	1.1	1.2	1.6	3.4
TE2, F3+F4, 150 Km/h	1.0	1.2	1.2	2.2
TE2, F3+F4, 175 Km/h	1.1	1.0	2.3	3.3
TE3, F3+F4, unlimited	1.0	1.1	1.7	3.0

Table 6: Mean driver safety perception

Scale used: 1 – Not perceivable, 2 – Slightly perceivable, 3 – Perceivable but not dangerous and 4 – Perceivable and dangerous

After the analysis of the results, there is no perceivable effect in the vehicle handling in the driving simulations if the steering performance is degraded for only 50 milliseconds (ms). There were only isolated cases in which the driver perceived “something”. However, it is more reasonable to consider these cases as being false positives, as drivers also imaging faulty events for simulations without any error injection.

To confirm the subjective perception of the drivers in an objective way, the recorded driving parameters were evaluated. The plots of the test cases show a clear “flat” or “hole” event at the moment at which the error is occurring, corresponding to the performance loss cases of “frozen” steering and “torque-less” steering, respectively (cf. Figure 16). The more visible the “flat” or “hole” event is, the clearer is the effect on the performance (cf. Figure 17).

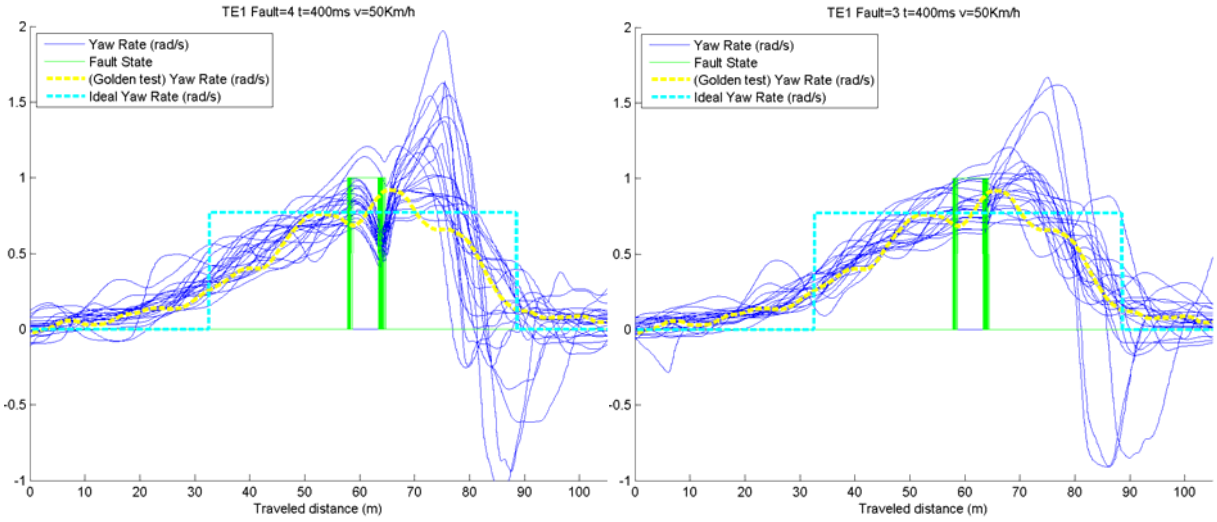


Figure 16: Dynacar TE1, faults are clearly visible as “hole” (left) or “flat” (right)

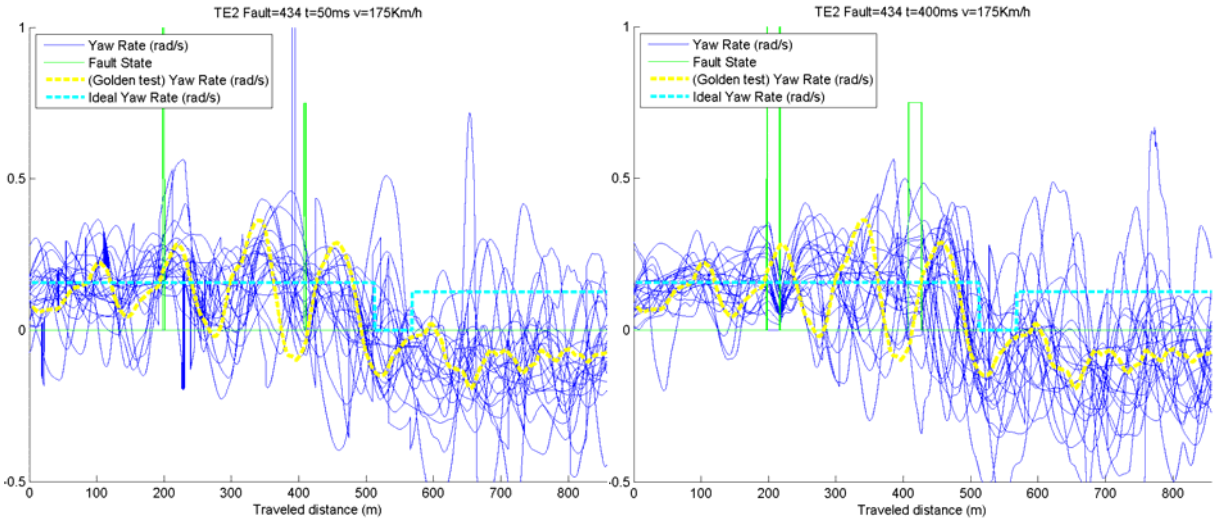


Figure 17: Dynacar TE2, with multiple faults, more visible faults on right plot

After analysing the coincidence in the plots and the subjective perception annotations, the conclusion is that with a fault duration greater than 250 ms the driving could be clearly unsafe, and that the driver starts perceiving the performance degradation when the fault duration is about 150 ms. The data shows that an adaption time of 50 ms in the SAPC could not degrade the safety performance because the estimation of the driver reaction time is close to 150 ms, 3 times the presently selected SAPC adaptation time.

These results validate the requirements defined in the Deliverable D2.2 [4] for the overall time response for safety-critical functionality. In D2.2 it was defined that failover times should be less than 50 ms, with 10 ms for fault detection and 10 ms for passivation, so that the fault-recovery takes less than 50 ms for safety-critical functionality with required fail operational behaviour.

3.3.3 Use Case concerned with Energy Management

The use case related to energy efficiency (UC_511_01, “Adaptation for Range Extension”), analyses the range extension of the vehicle attainable with various energy efficiency adaptations. This chapter describes how the use case has been realised. For detailed calculations of efficiency improvements see Section 5.3.

In this use case the vehicle is driving on the road and the Battery Management System (BMS) detects a SOC (State Of Charge) of less than 35%. In this situation, the SAPC can be used to load different application profiles, where the use of energy is prioritised to increase the remaining range. The following adaptations were performed:

- The BMS cuts off the power supply for auxiliary services (radio, audio system, navigation, and air conditioning). The services’ energy consumption will be defined from literature.
- The In-Wheel-Motor (IWM) performance (torque and power) is reduced by 50%.
- The BbW system activates rules to maximise the regenerative braking while taking into account that the vehicle dynamics must remain safe.

In order to evaluate the range extension criteria, the Dynacar platform has been used to simulate the New European Driving Cycles (NEDC) shown in Figure 18, as it is considered a reference standard in fuel consumption analysis. The reference NEDC is a speed-time profile that lasts 20 minutes and implies that the vehicle travels a 10 km interval. The platform was configured to autonomous driving mode in order to eliminate influences from a manual driver.

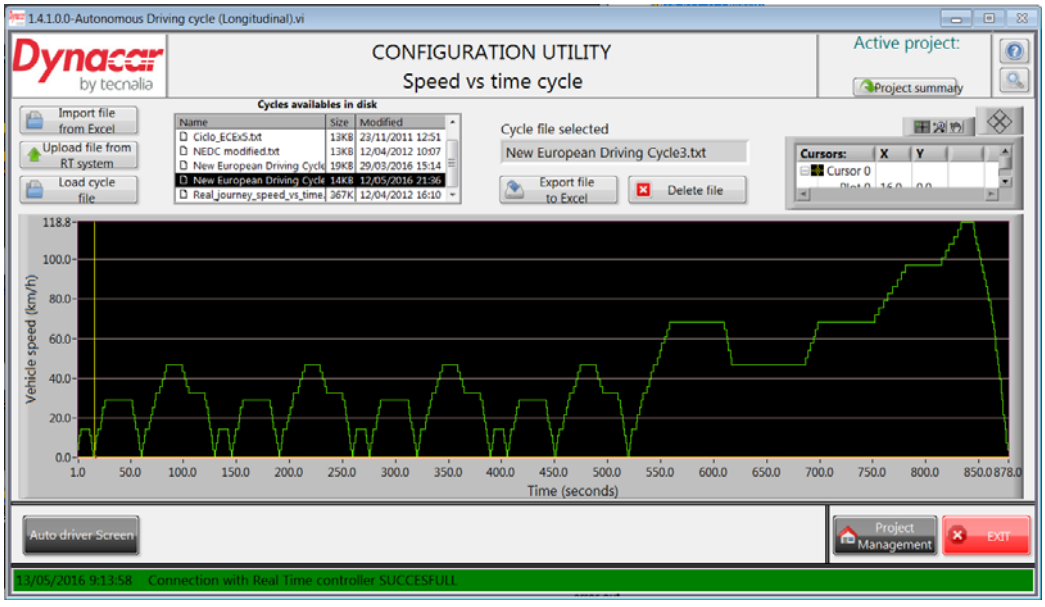


Figure 18: Energy consumption test for range extension evaluation (UC_511_01)

Using the measurements from this setup, Section 5.3 shows that if the SAPC triggers the switchover when the battery SOC drops below 35%, the vehicle's range can be extended by 25%.

Next to energy saving potential from driving dynamics, SafeAdapt's adaptive technology was also used to deactivate the non-critical SomnoAlert® drowsiness detection system in the event of a low remaining battery charge.

Current drowsiness detection systems are based on the analysis of many parameters available on the CAN network of the vehicle in order to deduce the state of the driver. In the simplest form, they only make use of signals commonly available, like the action on the pedals, the steering wheels, the blinkers state (on/off), the hour of the day, and so on. More sophisticated ones combine this information with the analysis of the trajectory of the vehicle in the lane using the output from a Lane Departure Warning system, comparing the driver performance against a reference standard measured at the beginning of the ride. In the SafeAdapt project the most recent version of SomnoAlert is used, which is based on the direct measure of a biologic parameter of the driver, namely the measure of a respiratory effort to infer drowsiness. Thoracic effort is related to autonomic nervous system, so it can provide direct information of the driver physiological state. The algorithm performing this job, called TEDD (Thoracic Effort Drowsiness Detection), was developed using a direct measure done using a thoracic band worn by the testers, and connected to a PC via Bluetooth. Along the project it was further carried on the development of an algorithm able to extract the thoracic effort from the images of a camera looking at the driver thorax, in order to perform a contactless measure of this parameter to be elaborated by the TEDD algorithm to detect the drowsiness condition and issue an appropriate warning.

Within SafeAdapt, the SomnoAlert is used as an exemplary ADAS application which can be deactivated in an energy saving scenario. SomnoAlert, which is based on a software analysing images taken by a camera, is sufficiently representative for a typical ADAS based on camera sensors, especially in terms of energy consumption and with respect to the upcoming new generation of semi- and fully-automated vehicles. Many of these vehicles are expected to use electrical propulsion, and thus, could make use of SafeAdapt architecture. So a first essay about how to handle the need of such systems with the need of using adaptive network appeared quite interesting. Therefore, SomnoAlert has been implemented on the Dynacar simulator. As explained, the SomnoAlert application was used as an example ADAS application, and the use case implemented is the following: if the battery charge level is above a certain threshold, the application works normally, but once the battery charge level is below that threshold, all the auxiliary systems are switched off, thus, demonstrating the versatility of the adaption concept. The following Table 7 shows how the SomnoAlert system is adapted based on the vehicle's state of charge.

	NON-DROWSY	DROWSY
BATTERY	Nothing to do	SomnoAlert alarm (either acoustic or visual)
NO BATTERY	Switch off auxiliary systems	Switch off auxiliary systems

Table 7: States of SomnoAlert

3.4 Frozen-Standby Demonstrator

3.4.1 Platform Description

To evaluate the full spectrum of adaptivity with respect to fault tolerance, the ability to perform autonomous recovery in a flexible manner is of specific interest. As the RACE, TMDP, and AUTOSAR platforms do not support the concept of dynamically loading individual applications at runtime, an additional demonstrator is used to evaluate the feasibility of this dynamic scenario. For this, the concept of an “application database” is introduced, which stores the binary images of all applications, allowing them to be transferred on-demand to the node requiring the application, as depicted in Figure 19. In case of a failure of the “main node”, an application originally running on this will be transferred from the “application database” to a “back-up” node and continue working there. This will ensure the safe operation of the whole system without noticeable interruptions. Further, a time-triggered switch is used to fulfil the strict time requirements. With this setup, it is now of interest to determine the time needed for detecting the failure, fetching the application from database, and running it on the back-up node. Based on this measurement the suitability of this adaptation method with respect to fail-operational scenarios can be judged.

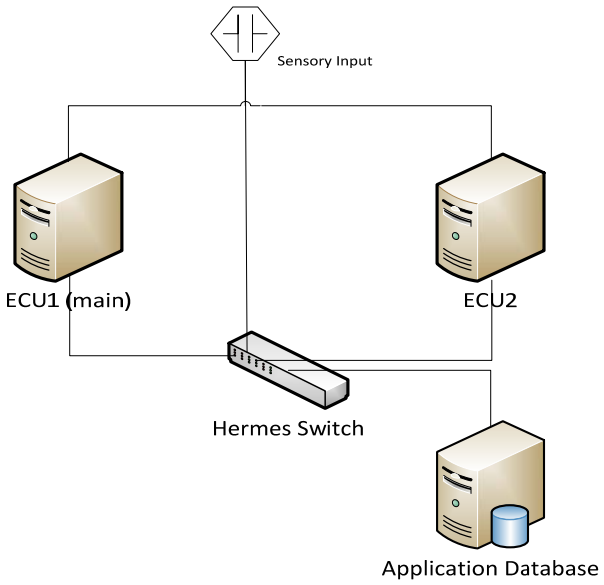


Figure 19: Frozen-Standby demonstrator

The demonstrator is implemented with two nodes and a deterministic Ethernet network. The whole system setup consists of two computation nodes, a TTTech Hermes gigabit switch and two additional supportive devices: one to supply these two nodes with a sensory input and a second one as an application database to deliver failed application to the back-up node. Beaglebones were chosen as supportive devices for demonstrator setup (see Figure 20).

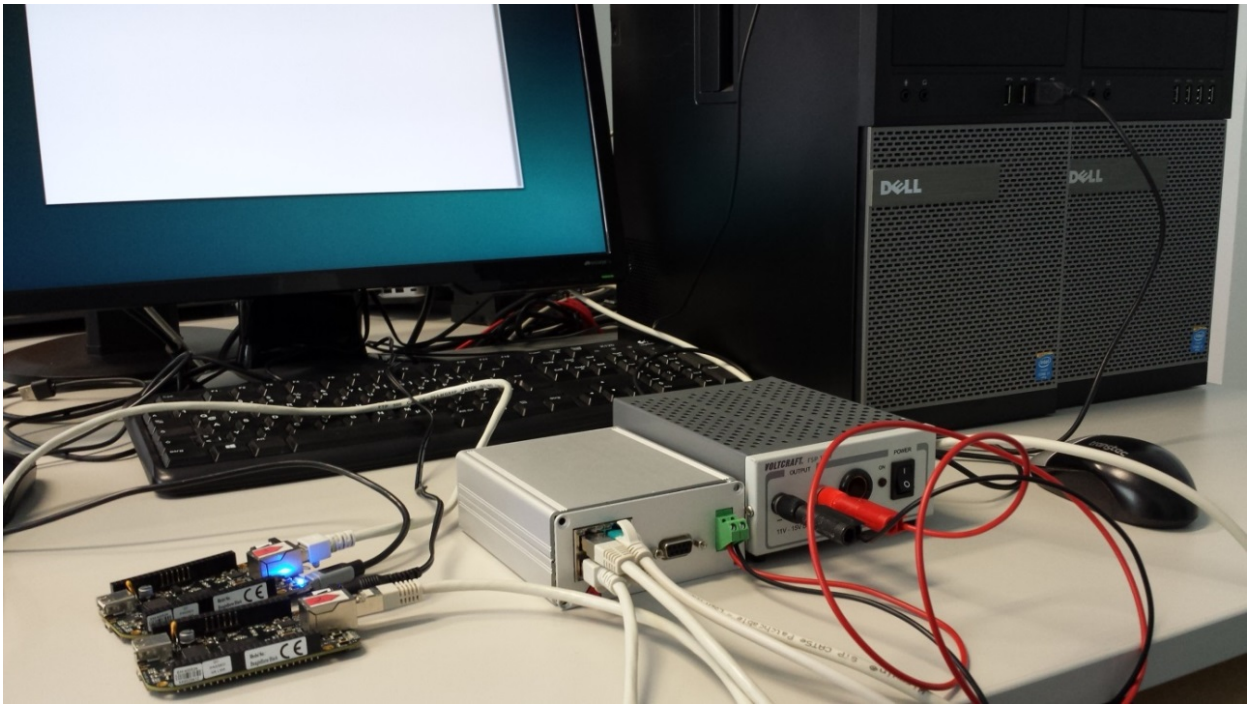


Figure 20: Frozen-Standby demonstrator: Two ECUs, Hermes Switch and two beaglebones

3.4.2 Evaluation of Failover Times

For the evaluation of frozen-standby failover times, the time required by the following reconfiguration steps is considered:

a) Detection of faulty function (hardware and/or software fault)
b) Disconnect the faulty function / unit from the network
c) Lookup in reconfiguration table to determine new host for failed functionality
d) Preparation of new host for taking over the function (i.e., stop an non-critical function)
e) Load at least a gracefully degraded version of functionality onto new host
f) Connect the actuators and sensors to the new unit
g) Start the function on the new unit

Based on this, the measurements for the failover time was conducted by analysing every step and measuring the required time, as further described in the following and also depicted in a trace output of the demonstrator in Table 8:

1) T0 is the start moment which is denoted when the first heartbeat is sent from ECU1 to ECU2

2) Operation step 1: artificial insertion of a failure in ECU1 triggers the reconfiguration process and initialises the time counting for the reconfiguration

3) Operation step 2: Detection time between ECU 1 crashing and ECU 2 recognising that a failure occurred. There is a waiting interval integrated in t2 in order to differentiate between a missed packet and a crash. This time could be a multiple of the packet sending frequency of ECU1. Below it is set to 6 ms, which equals the time duration for 3 packets (Time needed: **t1 = 6 ms**)

4) Operation step 3: process time of the switch **t2 = 8.172 µs** and upload of degraded function via the switch from application database to ECU2
5) Operation step 4: initialise and start programme on ECU2 (note: the sensor simulation is connected to both ECUs permanently since there was no significant influence on the time to establish a new connection. The time was in the lower µs range and is thereby considered negligible)

In sum, it can be assumed that the Frozen-Standby reconfiguration will be possible in **6 ms to 10 ms**. Deviations will depend on the sending frequency and size of the packets.

ID	Message
[Backup random number generator]	Initialized
[Backup random number generator]	Going to wait for signal from heartbeat watcher... Node1 started sending at: 2016-06-28 15:05:09
[Heartbeat watcher]	The application failed to receive the last 3 heartbeats in the last 6000000ns
[Backup random number generator]	Crash detection time: 2016-06-28 15:05:40.399640194
[Backup random number generator]	Time from crash to backup started: 0.006008172
[Backup random number generator]	Backup generator started: 2016-06-28 15:05:40.399648366

Table 8: Trace of Frozen-Standby demonstrator

3.5 Fail-Operational AUTOSAR Demonstrator

In order to demonstrate the compatibility between the Safe Adaptation Platform Core (SAPC) and then current AUTOSAR standard, a specific demonstrator was developed. It is moreover used to showcase that the adaptiveness of the SafeAdapt project is independent from the used technology, thus, allowing the use of more dynamic and flexible operating systems, as seen in the RACE demonstrator and more static operating systems, such as OSEK and AUTOSAR. In addition, the demonstrator is utilised to determine how adaptivity can be integrated into the AUTOSAR standard in a non-invasive manner and rely on existing concept as far as possible.

The demonstrator is a scale model of an electric vehicle with electronic steering and braking capabilities (Figure 21). At its core, it consists of two control units responsible for providing critical steering and braking functionalities as well as additional non-critical functionalities. Consequently, the two control units are both equipped with an instance of the SAPC in order to ensure fail-operational behaviour for the critical functionalities. Moreover, both ECUs are running the Arctic Core AUTOSAR implementation. In detail, both ECUs share a global time base to allow the SAPC to be executed in a synchronous manner, thus, enabling short failover times within a single execution period of the SAPC. Next to this, each ECU is preconfigured with a set of schedule tables for every anticipated failure mode of the system. Thereby, the foundation for system-wide runtime adaptation is given while still adhering to the static design principles of the AUTOSAR standard. The mapping between these schedule tables and the system-wide failure modes is stored within the local database of the SAPC, including additional information, such as the required PDU groups in each failure mode (cf. Deliverable 3.3 [5]). At runtime, the SAPC can now perform a lookup in its database with the received Health Vector information, and in the case of a failure,

instruct the AUTOSAR OS to activate a different set of PDU groups, cancel the currently active schedule table, and start the new schedule table at a predetermined offset relative to the global time base. Through this, all ECUs transition into a new mode of operation at the same point in time, thereby, eliminating the potential for undefined system states in the case of a failure. In addition, the system supports backup instances to either be in a hot- or cold-standby mode, thereby allowing resource-efficient implementations of fail-operational behaviour.

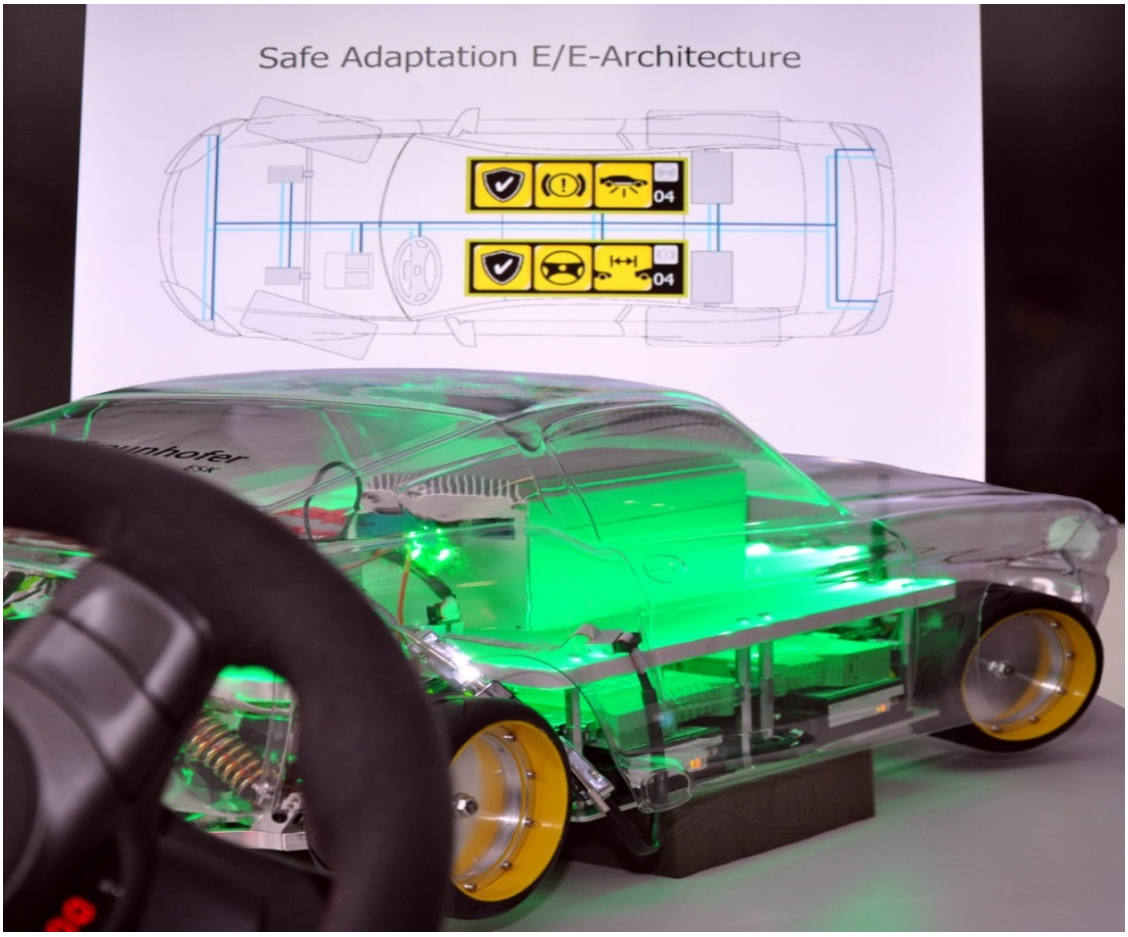


Figure 21: Fail-Operation AUTOSAR demonstrator

Next to the runtime perspective, this demonstrator further focuses on an integrated tool workflow, thereby, interlinking the system modelling results of Work Package 4 with the runtime adaptation. As such, the following workflow depicted in Figure 22 only presents the excerpt concerned with AUTOSAR-related aspects of the entire system design tool flow (see Figure 23).

More specifically, the developed workflow aims at automating AUTOSAR-related design and configuration aspects. Here, system requirements, such as the application-specific maximal acceptable failover times, and further properties, such as worst case execution times (WCET) or HW-specific failure modes are exported from the higher order EAST-ADL models of Work Package 4 in form of AUTOSAR ARXML exchange format files. Thereafter, a tool-based analysis of the data flows between applications, the hardware architecture and its failure modes is performed to derive a mathematical problem formulation. This formulation aims at synthesising a valid system configuration for each anticipated failure mode by solving the generated equations.

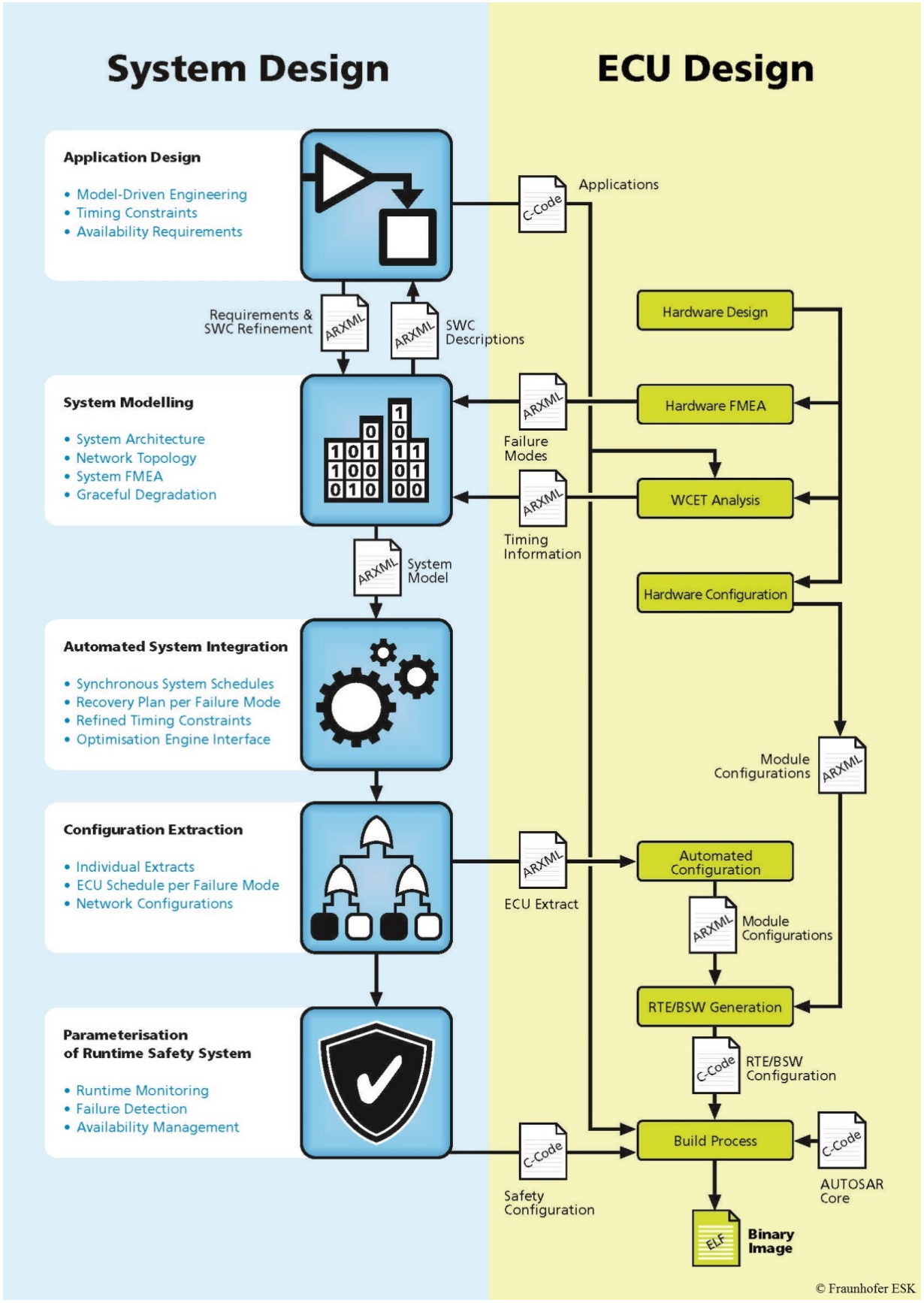


Figure 22: AUTOSAR workflow

More specifically, each system configuration consists of a mapping between a certain system state, which is communicated by the Health Vector, and a set of ECU-specific runtime configurations. These ECU-specific configurations in turn contain detailed AUTOSAR timing requirements for every hosted application and every network communication interaction between applications. Based on this information, the local database is generated for each SAPC instance. In addition, the previously calculated timing requirements are detailed enough to enable a direct transformation into AUTOSAR ECU configuration artefacts, which are in turn directly used for generating RTE and module source code, such as the OS schedule tables, PDU and signal routings, communication timings, OS tasks, and application to task mappings.

In sum, the developed workflow and tooling provides the foundation for efficiently applying the concept of adaptation in the context of AUTOSAR. This is achieved by eliminating a multitude of human sources of error through tool automation and significantly reducing the design effort for fail-operational systems. Moreover, the process ensures the typical safety benefits of static system design while still allowing a reduction in complexity through generic runtime adaption, thus showcasing the most advanced form of safety-critical runtime adaption possible within the current AUTOSAR standard.

4 Evaluation of Development Process

4.1 Tool-Chain

A number of tools have been developed to support the design, validation, and implementation of adaptive vehicle software systems. This includes a modelling tool for designing the software system using EAST-ADL, and a tool for validating the system's adaptive behaviour using simulation and fault injection, the automatic generation of the AUTOSAR model based on the EAST-ADL model, and the generation of the configuration file for the SAPC database. In the following, these tools are described in detail. Figure 23 presents an overview view of the SafeAdapt tools regarding the design, implementation and V&V flow.

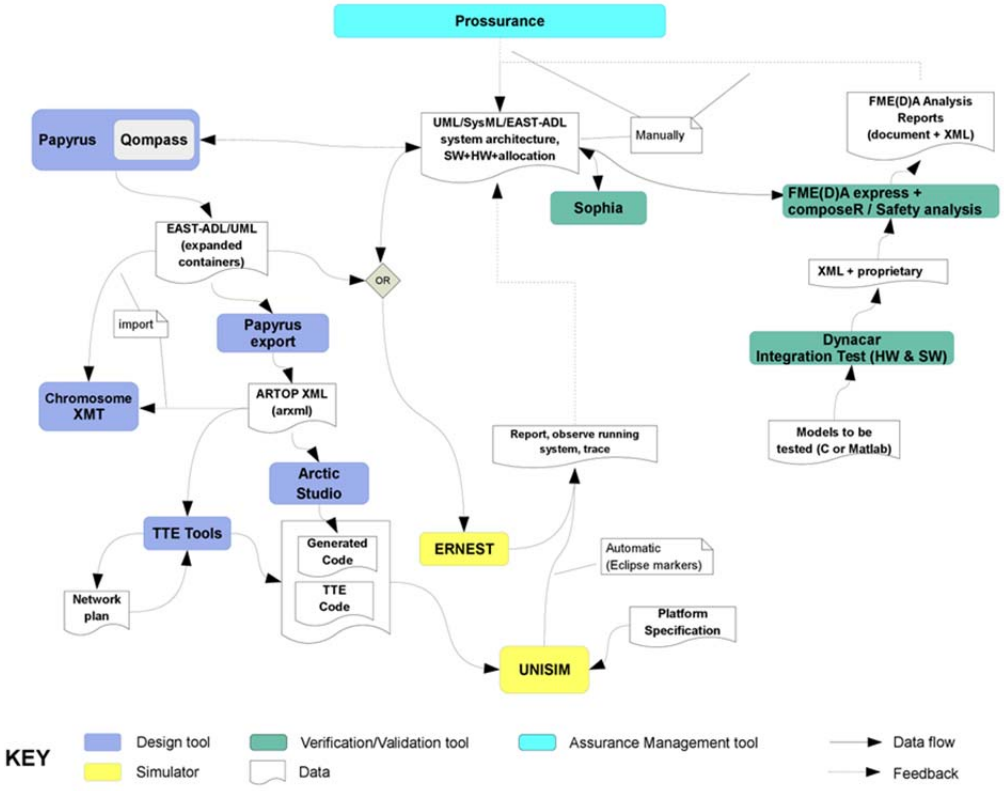


Figure 23: SafeAdapt tools overview

4.1.1 Modelling Adaptive Vehicle Software Systems

The system used in the demonstrator is composed of a set of applications (e.g., Steer-by-Wire, Break-by-Wire, etc.) that interact with each other to meet the needs of the user. In addition, while the system is in operation it needs to adapt itself in response to context changes such as application failure. In the following, an approach to model the RACE car demonstrator is presented.

Modelling the System's Functional Architecture

The system's functionality is modelled as a composite structure that consists of a set of functional components that interact with each other through functional ports. In addition, to specify the system's variability, the cardinality of the system components is utilised. A component with cardinality {0 or 1} is an optional component which can or cannot exist at runtime, while the

cardinality {2} means that the component has two instances and the system can switch between them at runtime. A component with cardinality {1} specifies that the component is mandatory and should always exist while the system is in operation.

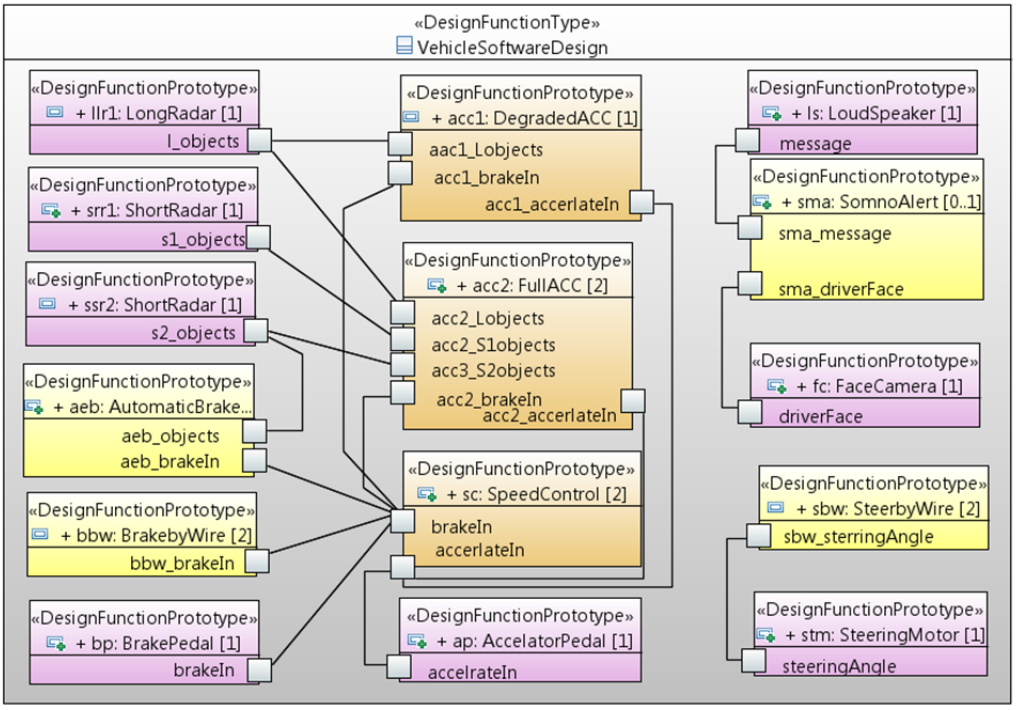


Figure 24: Design model for the vehicle system's functionality

The design model for the adaptive vehicle software is shown in Figure 24. In this figure, the fully adaptive cruise control (ACC) has the cardinality {2} which means that it has two instances at runtime. The two instances can replace each other in the case on of them fails. The SomnoAlert is an optional component, i.e., it can exist or not while the system in operation.

Modelling the System's Hardware Architecture

As discussed above, a system component may have two instances (e.g., the ACC). These two replicas need to be allocated to different electronic control units (ECU) to increase the reliability of the system. Therefore, in case of a failure of the ECU that hosts the active replica, the other replica can be activated to achieve the system functionality. To specify such an allocation, the system's hardware platform needs to first be specified and then the allocation of the software components can be defined (see below).

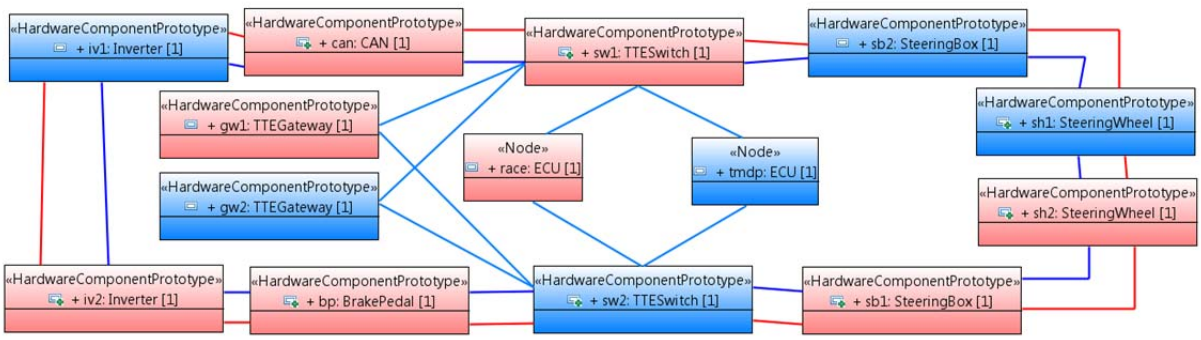


Figure 25: Design model for the system's hardware architecture

The hardware model is designed as a composite structure that contains the hardware elements of the system (e.g., ECUs, sensors, actuators, etc.). A hardware platform model is shown in Figure 25, which corresponds to the RACE vehicle architecture. It consists of the two ECUs. These RACE and TMDP ECUs are connected with each other via a hardware connector. It also includes a number of sensors and actuators such as brake pedal, steering box, steering wheel, etc.

Modelling the Timing Requirements

To model the system's timing requirements, the technique proposed in the project TIMMO-2-USE (TIMing MOdel - TOols, algorithms, languages, methodology, and USE cases) was adopted. To specify a timing requirement, events that are associated with software components or one of their ports are defined. These events are then used to define a timing constraint such as execution time constraint, periodic constraint, reaction constraint, etc.

For the adaptive vehicle system a number of constraints have been defined. Two constraints are shown in Figure 26. First, to specify a periodic constraint, an event associated with the Brake-by-Wire application is defined (BBWEvent). Then, a periodic constraint is specified that references this event. The BbW application has a minimum inter-arrival time and period of 60 milliseconds. Secondly, the execution time (10 milliseconds) for one of the functions of the Brake-by-Wire is defined based on the event BBW_T1Event that references the BBW function.

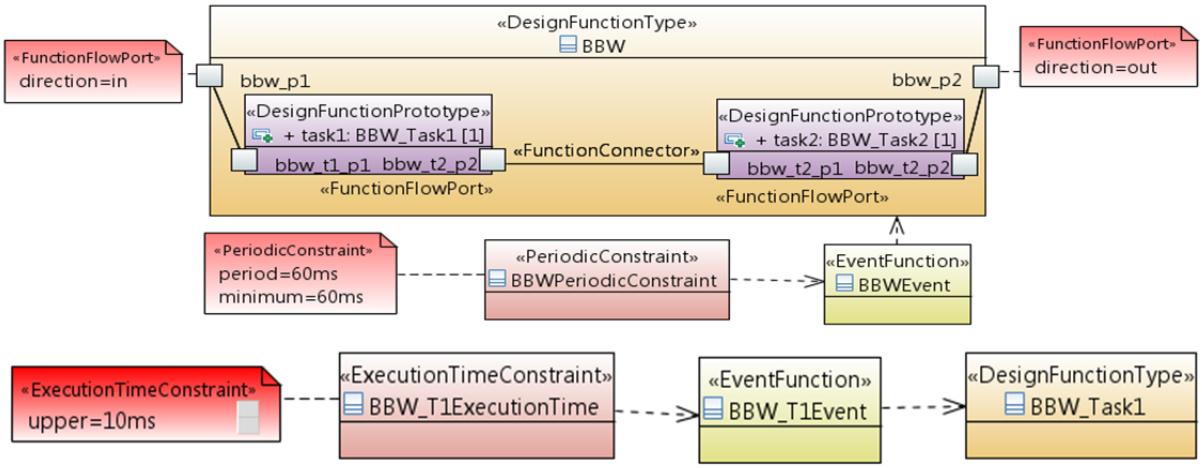


Figure 26: Example timing requirements in an adaptive vehicle system

Modelling the System's Adaptive Behaviour

To adapt the system in response to context changes, there is a need for a mechanism to decide when and what to adapt and then apply the defined adaptation actions. This is performed by the system management component. The system management switches from one system configuration (a certain state of the system) to another configuration in response to an adaptation trigger (e.g., failure of the ACC). Thus, the adaptation triggers and the different runtime configurations (states) of the system must be modelled. Both, the adaptation triggers and the system configuration are a runtime system state. To model the system states, the concept of instance specification was adopted, where instances of the system's design (functionality and hardware platform) are created and configured to specify the runtime configurations of the system and the adaptation triggers.

An example adaptation trigger is shown in Figure 27 (see the left part). For each application a runtime state is defined which can be one of the following: Active, Inactive, Hot, Cold, or Failed. In this example SBW0, BBW0, SMA, and AEB are in the state "Active", SBW1 is in the state "Hot", BBW1 and ACC1 are in the state "Cold", and ACC0 is in the state "Failed".

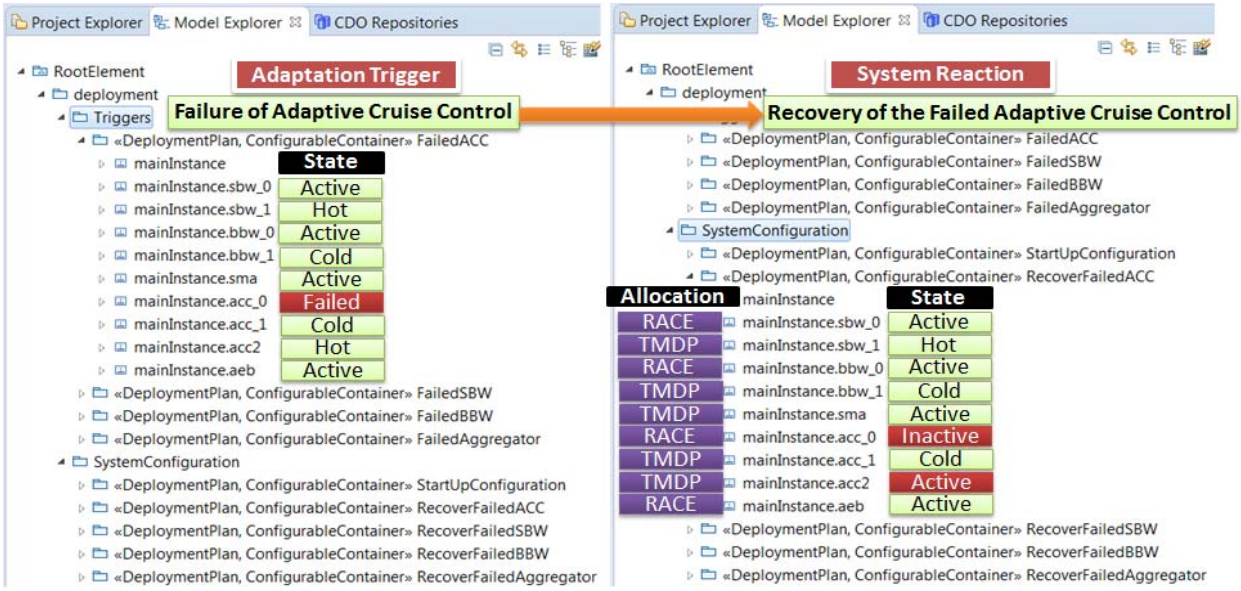


Figure 27: Specifying the adaptation triggers and the system response to that trigger

An example system configuration to recover from the failure of ACC0 is shown in Figure 27 (see the right part). The instance specification of each application is defined as <Application name, ECU allocation, State> where this configuration is defined as follows:

{<SBW0, RACE, Active>, <SBW1, TMDP, Hot>, <BBW0, RACE, Active>, <BBW1, TMDP, Cold>, <SMA, TMDP, Active>, <ACC0, RACE, Active>, <ACC1, TMDP, Cold>, <ACC2, TMDP, Hot>, <AEB, RACE, Active>}

To model the adaptive behaviour of the system, a state machine approach was utilised. In the state machine in Figure 28, the states are the different configurations of the system while the transitions are the adaptation triggers that move the system from one configuration to another. For example, in response to a failure of the ACC (the adaptation trigger specified on the left part of Figure 27), the system moves from its initial configuration to a configuration that recovers this failure (i.e., the system configuration shown in the right part of Figure 27). This switching of the system is specified using the first transition “ACCFailure” in Figure 28. In this transition, the state of the first instance of the ACC is changed from “Failed” to “Inactive” while the state of the second instance of the ACC is changed from “Hot” to “Active”.

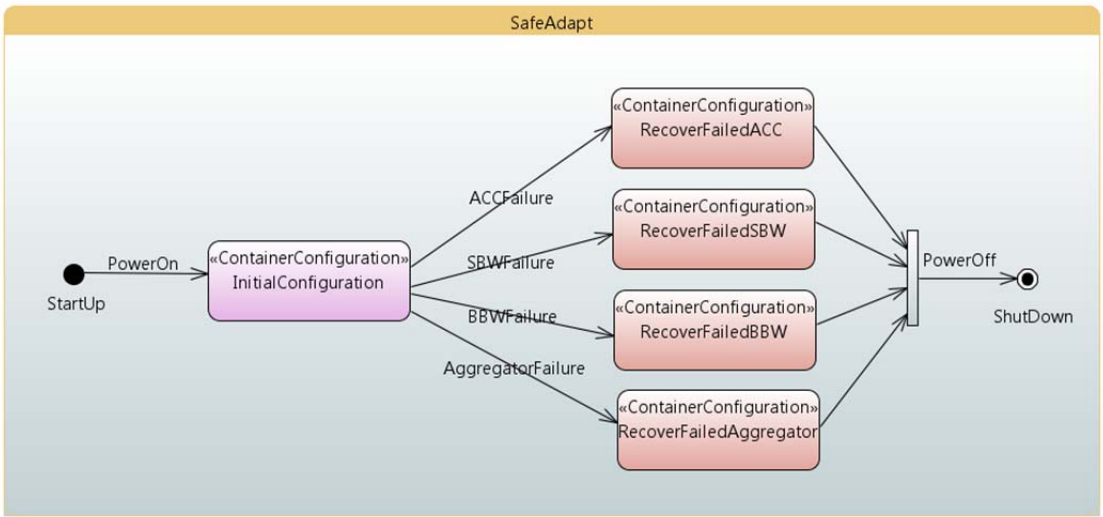


Figure 28: Adaptive behaviour for the vehicle software system

4.1.2 Simulation

In order to ensure that the vehicle software system model in the previous sections is going to execute correctly and improves the reliability of the system by reacting to the expected failures, the system can be *simulated*. As shown in Figure 23, there are two different ways to simulate an application. Simulation can either use the binary code for a target processor or more high level code. The former is done with UniSim-VP, the latter with the ERNEST simulator, which are described in the following.

Code Generation for Validation

At first "C" code needs to be generated that will run on the UniSim simulator. For this, an Eclipse plugin was created that transforms the design models into "C" code. Figure 29 shows how the plugin can be used to generate "C" code. The software engineer can use the modelled state machine for the adaptive behaviour of the system to generate and execute a UniSim "C" project automatically.

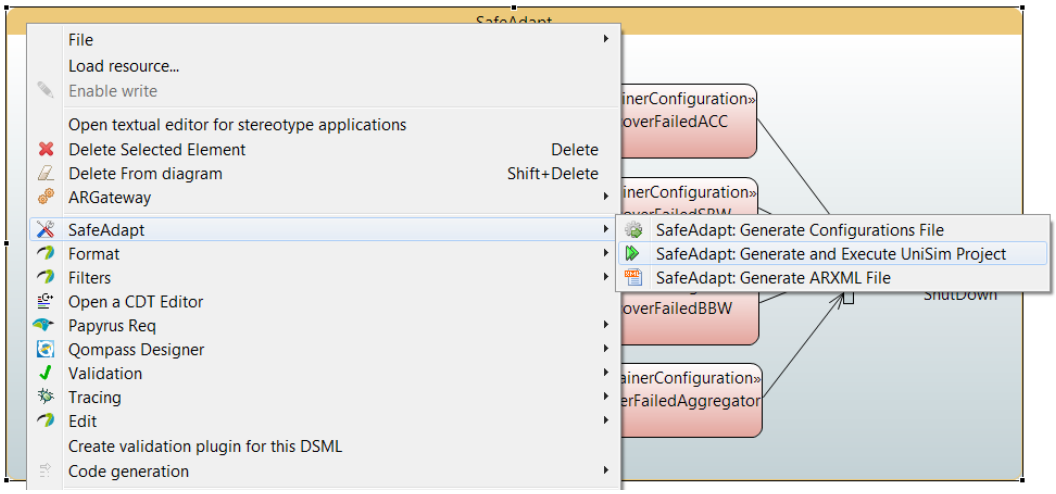


Figure 29: SafeAdapt plugin to generate "C" code from the design models

The generated project as shown in Figure 30 consists of simulator files, an adaptation manager, system applications, scheduler, and telnet connection.

Simulator files: The simulator files include the UniSim simulator and its configuration file. The configuration file specifies, for example, the binaries to run on the simulator (e.g., main.c.elf) and the starting address for executing the binary based on the simulator memory map (see Figure 30)

Adaptation Manager: The adaptation manager is responsible for executing the adaptation scenarios. It has three main tasks: "trigger adaptation", "decide adaptation actions", and "execute the adaptation actions".

System Applications: For each software component defined in the design model, the corresponding "C" code for the application (i.e., an empty skeleton) is generated.

Scheduler: To coordinate the execution of the system's functionality and the adaptation manager, a system scheduler that manages the execution of the applications and the adaptation scenarios is generated.

Telnet Connection: To monitor the execution of the adaptation scenarios, the telnet connection outputs some printouts that show the current system state, the trigger of the system adaptation, and the execution of the adaptation plans.

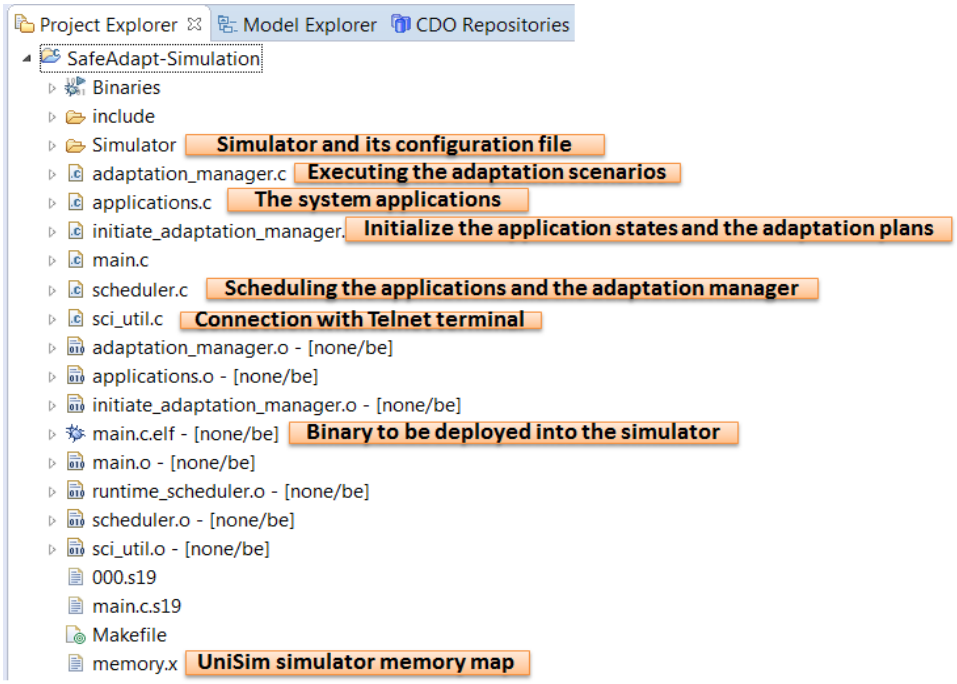


Figure 30: The generated C project that can run on the UniSim simulator

Validating Adaptive Behaviour

The software engineer can execute the generated C project (see Figure 31). The simulator is launched using the configuration file (Figure 31-3) and a terminal is also launched and connected to the simulator (see Figure 31-2).

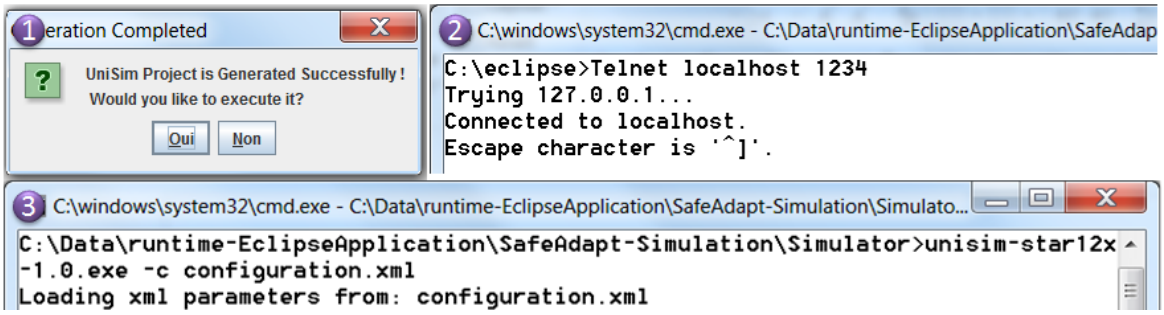


Figure 31: Executing the generated C project on the UniSim simulator

When the simulator starts, it moves the system from the non-set state to its initial configuration as shown in Figure 32. Then, it starts to execute the pre-defined adaptation scenarios (i.e., the adaptation scenarios 1-4 in Figure 32). An example execution of a scenario is shown in Figure 32, where in response to a failure of the first instance of the ACC, two actions are identified: change the first instance state from “Failed” to “Inactive”, and change the second instance state from “Hot” to “Active”. Then its outputs are taken into account afterwards.

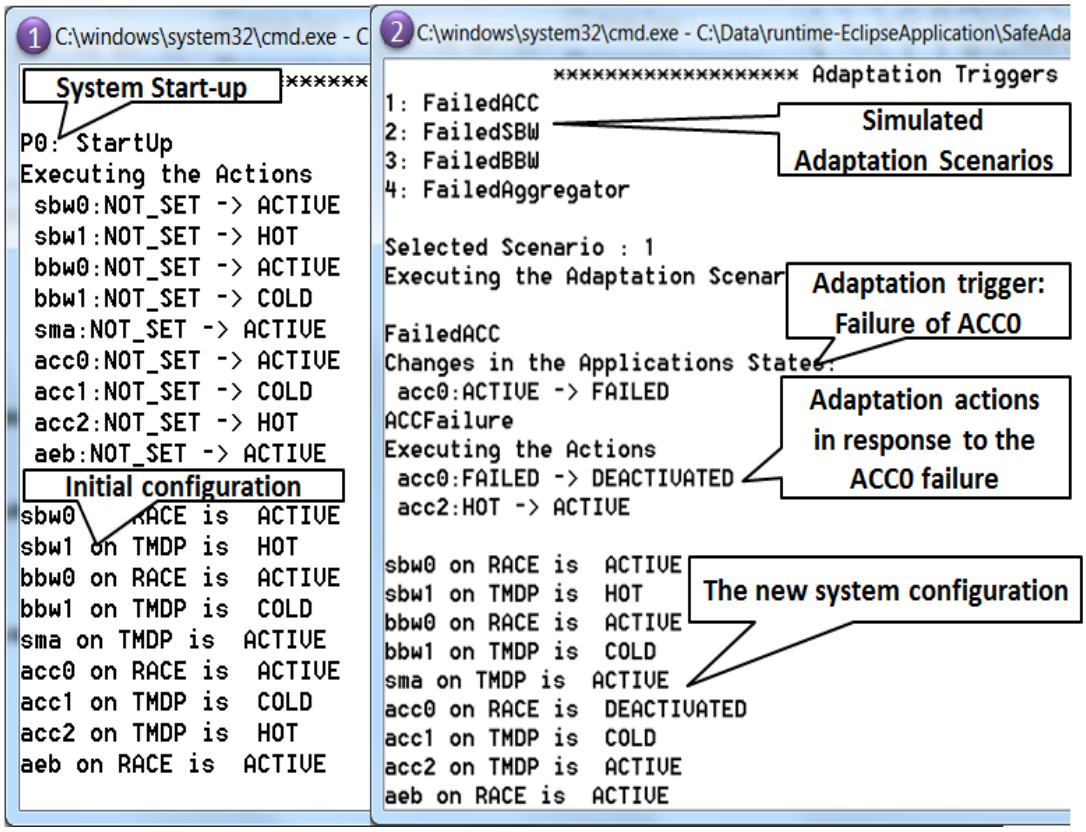


Figure 32: Executing the specified adaptation scenarios

To inject faults, UNISIM-VP allows instrumenting the variables of the embedded software. An example of an injected fault is shown in Figure 33. The state of the instance “SBW0” is set to “Active” while the state of the instance “SBW1” is set to “Failed” (see Figure 33-1). When the adaptation manager detects events that trigger an adaptation, it selects an adaptation plan which is then executed in response to this trigger. In Figure 33-2, the injected fault cannot be handled by the simulated software (i.e., there is no adaptation plan to cope with this fault). The advantage of the simulation is to detect such a situation early in the development and either define a specific adaptation scenario or rely on an automatic reconfiguration at runtime, as shown later in Section 4.2.

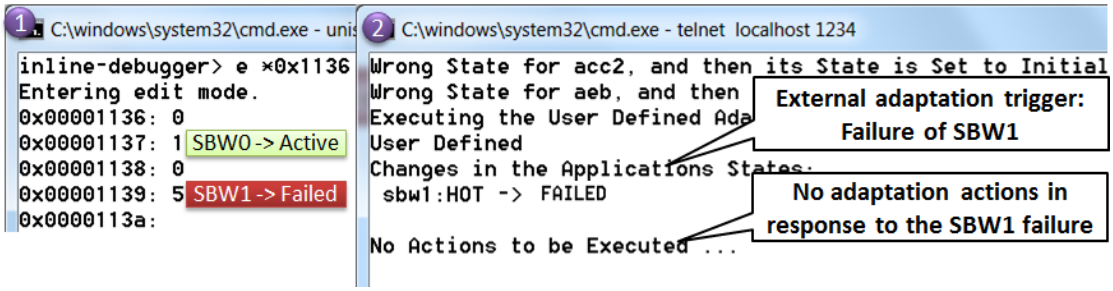


Figure 33: Injecting faults to the running software

Simulation with ERNEST

The ERNEST simulation framework enables the analysis of non-functional properties as well as adaptive behaviour on a system-wide level in early design stages. It is integrated into SafeAdapt's tool flow as shown in Figure 23. In order to validate non-functional requirements during the system modelling and design phases, ERNEST links a proprietary simulation framework to the Eclipse development environment. The heart of the ERNEST platform is a simulation-based analysis programme based on a simulation framework developed under the SystemC description language. The framework aids in simulating the behaviour of the modelled embedded system.

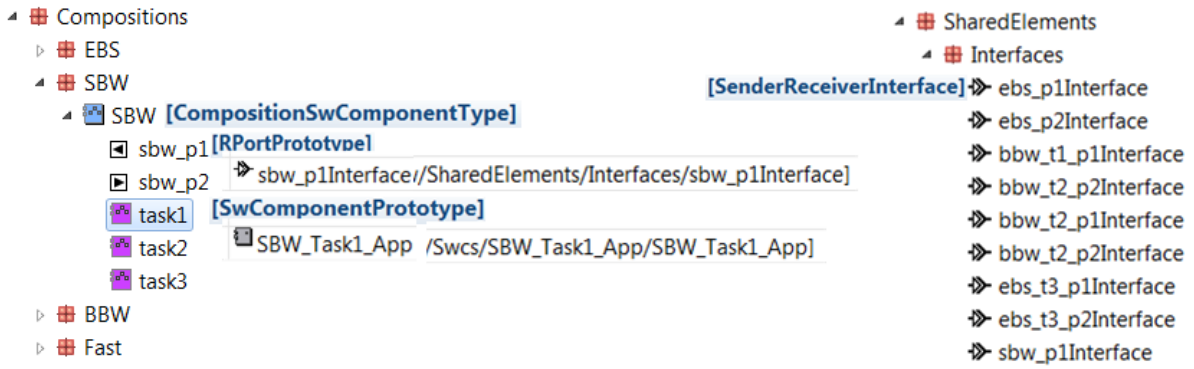
When using ERNEST, first of all a model of the system has to be created (in EAST-ADL for example). This model can be transformed automatically into an analysis model according to the ERNEST meta-model. In the analysis model the non-functional requirements of the system can be defined as constraints. Now the simulation code is generated in SystemC, and then executed using the simulation framework. During the simulation a trace is created which logs the communication behaviour. To analyse the resulting data, the information is fed back into the analysis model. Thereby the trace is analysed in order to check if all constraints are fulfilled. The results are visualised and markers in the analysis model show which components violated a constraint. Thereby the simulation results can be linked with the modelled system requirements.

Moreover the ERNEST simulation framework has been enhanced in order to allow the simulation of the needed degree of adaptation on system and component level. The SAPC has been added to the simulation framework, which allows the simulation of the SAPC to generate traces for the different adaptation plans and switchovers. The traces can be analysed with respect to the defined constraints to verify the integrity of the design.

4.1.3 Generating AUTOSAR Models form EAST-ADL Model

Code generation for the target is based on the AUTOSAR standard, whereas the design model is specified using EAST-ADL. In order to assure consistency between these two models, it is necessary to synchronise them by an automatic process. For this, a transformation that maps each concept at the design level to its corresponding implementation concept is proposed. An existing Eclipse plugin (AR Gateway) that exports an AUTOSAR models has been extended to cope with timing information required for reconfigurations. In the following, the generated AUTOSAR model of the demonstrator and its correspondence to the EAST-ADL model is described.

Each system application is transformed into a package (e.g., steer-by-wire application is transformed to SbW package) that has a composite component. The composite includes the input and output ports (e.g., sbw_p1 and sbw_p2), and a number of parts as shown in Figure 34. In addition, the timing requirements for each component are defined using "virtual function bus timing" elements. For example, SbW has a periodic constraint with period of 120 milliseconds and minimum inter-arrival time of 120 milliseconds (see Figure 34).



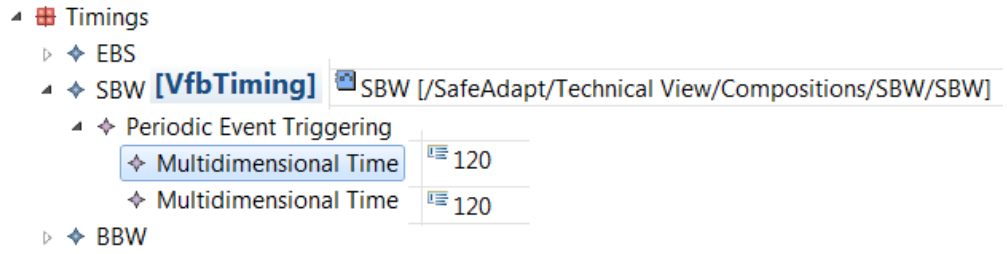


Figure 34: The system application in the AUTOSAR model

Second, the internal functions of an application are transformed into application components which include in/out ports and the internal behaviour. Timing information about the implementation is added. For example, the execution time of task1 of steer-by-wire is 10 milliseconds on the TMDP ECU as shown in Figure 35.

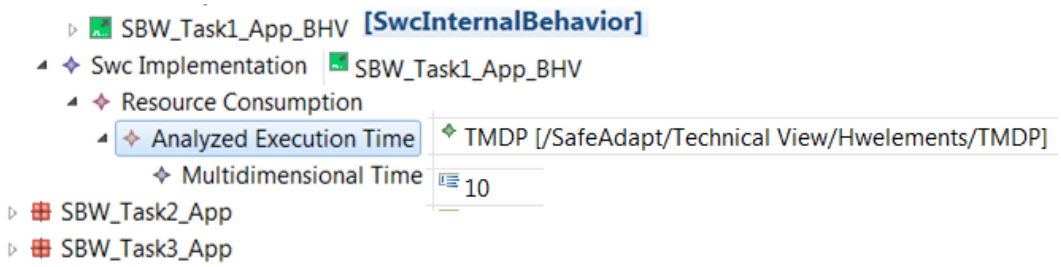


Figure 35: The internal functions of an application in the AUTOSAR model

Third, the hardware platform is transformed into three packages: ECUs, HWelements, and Communication. Each processing element (node) in the EAST-ADL model is transformed into an ECUInstance with communication ports to interact with each other. In addition, each node is transformed into a hardware element. A communication package is generated that captures the properties of the communication between the ECU instances as shown in Figure 36.

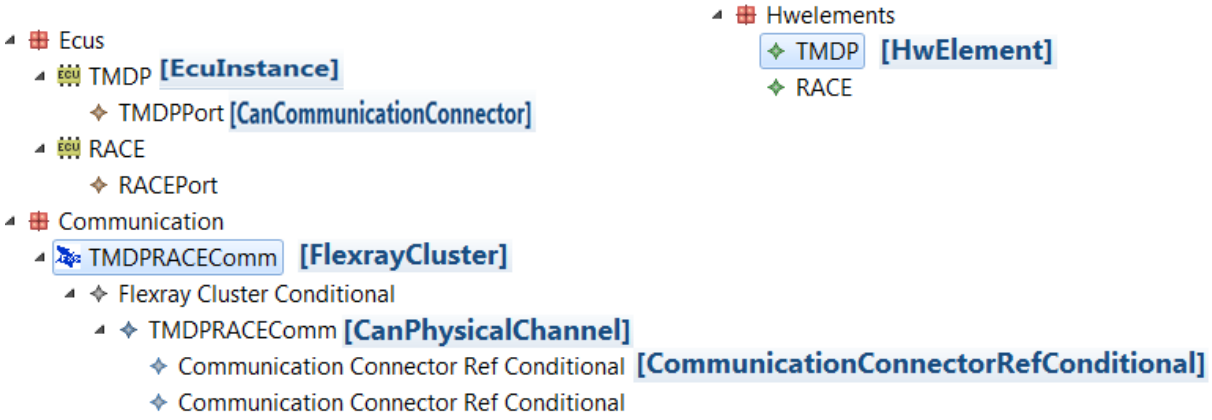


Figure 36: The hardware platform in the AUTOSAR model

Finally, the allocation of software components to hardware elements is transformed into a system mapping element as shown in Figure 37. This mapping specifies to which ECU the software elements are allocated (e.g., task1 of the emergency braking system (EBS) is allocated to RACE ECU (see Figure 37).

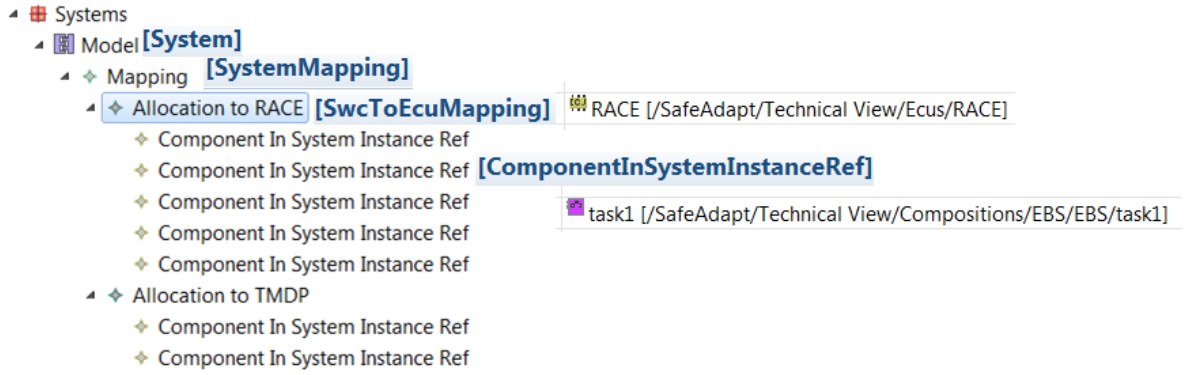


Figure 37: The mapping of the software components to the hardware elements

4.1.4 Generation of Configuration File for the Adaptation Core

Based on the adaptation state machine, a configuration file for the SAPC adaptation core is generated. This is split in two steps: at first a Java project is generated that contains the code that represents the adaptation machine (see the top part of Figure 38). When the generated Java project is compiled and executed, the configuration "C" file is generated as shown at the bottom of Figure 38. This split was motivated by existing tooling and the ability to manually setup a configuration file via Java code.

```
ConfigurationManager configs = new ConfigurationManager();
// Applications
ApplicationInstance acc1_RACE = configs.createApplicationInstance("acc1","RACE");
ApplicationInstance acc2_TMDP = configs.createApplicationInstance("acc2","TMDP");
…
ApplicationInstance sma_RACE = configs.createApplicationInstance("sma","RACE");
// ################ Adaptation Plan "ECUFailure" ################
AdaptationPlan ECUFailure = new AdaptationPlan("ECU2Failure");
ECUFailure.setAdaptationID("43");
// States of the Applications on the Source Configuration:"S1_IntialConfiguration"
ECUFailure.addCurrentApplicationState(acc1_RACE,AppState.HOT);
ECUFailure.addCurrentApplicationState(acc2_TMDP,AppState.ACTIVE);
…
ECUFailure.addCurrentApplicationState(sma_RACE,AppState.ACTIVE);
// States of the Applications on the Target Configuration:"S2_ECU2Failure"
ECUFailure.addTargetApplicationState(acc1_RACE,AppState.NOT_SET);
…
ECUFailure.addTargetApplicationState(sma_RACE,AppState.NOT_SET);
configs.addConfiguration(ECUFailure);
…
new SAPCGenerator().plan(configs);
```

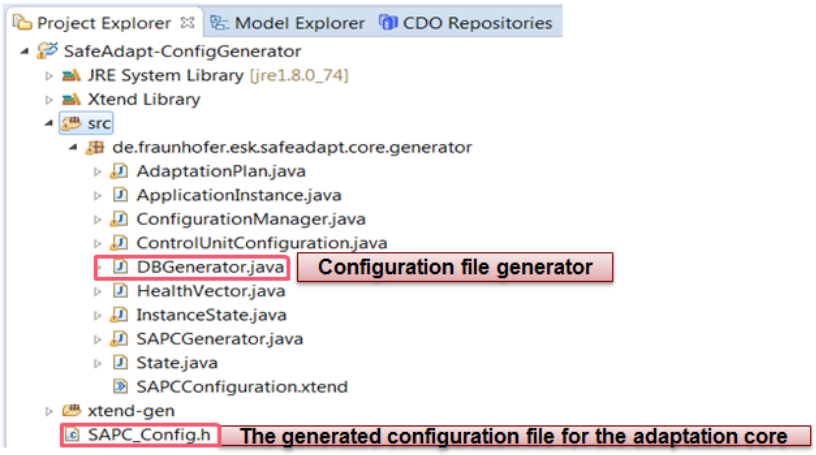


Figure 38: Generation of configuration file for SAPC adaptation core

4.2 Unanticipated Behaviour

Up to now, system configurations (set of software components along with their allocation to hardware components) have been calculated at design time and stored in a database of the SAPC. However, this approach cannot cover all scenarios: the combination of different adaptation reasons would lead to an exponentially growing number of configurations (and in turn a big database). It is also possible that a certain adaptation case has not been anticipated. Therefore, only a limited number of configurations are created and validated at design time and the system has to be able to calculate a new configuration at runtime (or utilise an internet-based service for this). It must also validate a configuration, i.e., assure that resource and timing requirements are met. This calculation requires a model@runtime, a subset of the original design model with information about requirements of software components and the capabilities of the hardware components. In the following, the focus lies on runtime support for ensuring the schedulability of a new system configuration at runtime.

The considerations that follow are done in the context of a specific scheduling policy based on EDF (earliest deadline first) for critical tasks and CBS (constant bandwidth server). The main reason for this choice is that EDF/CBS has been proven to work for mixed-critical systems and the calculation of a new schedule is much faster compared to the calculation of thread priorities. In particular, a heuristic and iterative algorithm is used. The iterative nature makes it suitable to calculate new configurations efficiently from existing ones. Figure 39 shows the output of a calculation tool for finding an allocation for a specific system configuration (the screenshots are taken from a desktop PC, but the calculation could also run on the target). It starts with an allocation where the most important tasks get a budget equal to the worst case execution time, and the less important tasks are assigned a budget equal to their average execution time. The initial allocation as shown in the top of Figure 39 shows a schedulable allocation. The probability of meeting the deadline for important tasks is 100%, while it is 11% for the unimportant tasks. Starting from this allocation, the possible allocations are searched to find better allocations with an increased probability of meeting the deadline of unimportant tasks. The bottom part of Figure 39 shows found allocations which guarantee that the deadline for the unimportant tasks (QoS = 1.0) will be met. The new allocation is found either through increasing the budget of the non-critical tasks or through moving tasks from one processing unit to another.

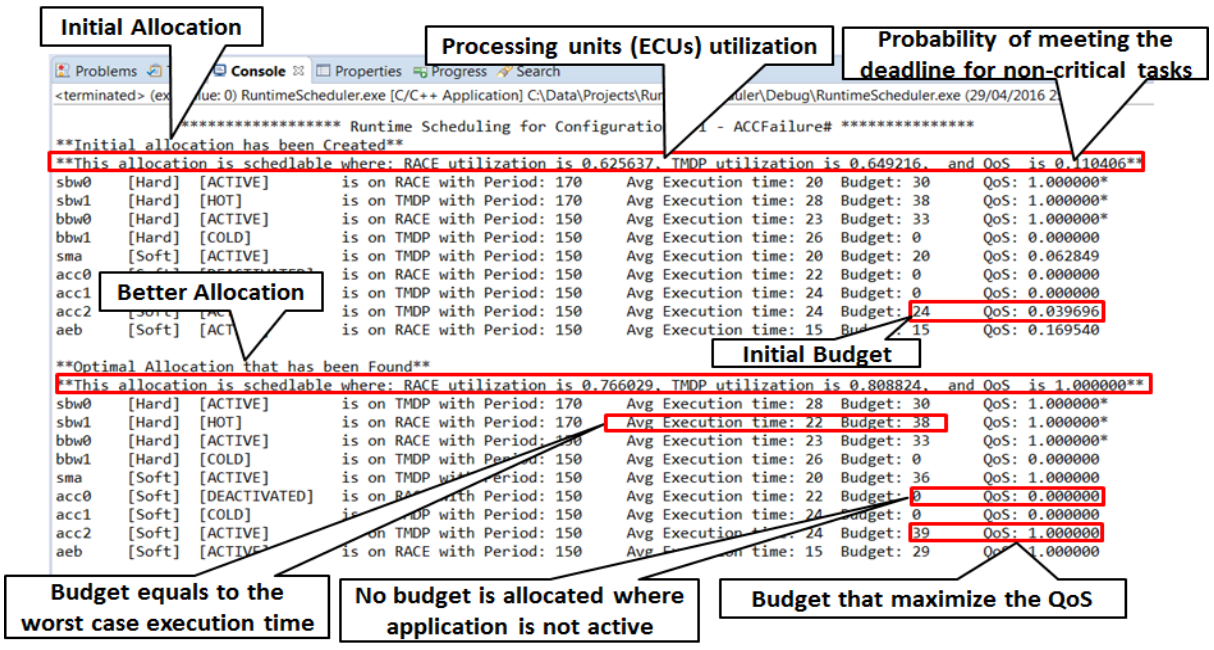


Figure 39: Finding an allocation for an anticipated system configuration

During runtime, tasks can be added, removed, or their timing information is changed. To check the schedulability of a new system configuration at runtime, the current system configuration is used as basis. Then, the difference between the current configuration and the new one is identified. Afterwards, a schedulability analysis is performed to check whether the new allocation is schedulable or not.

In sum, the results indicate that it is acceptable to apply methods of unanticipated reconfiguration for non-safety-critical applications, whereas the benefits of static system design and predetermined adaptation are best suited for all safety-relevant reconfigurations, as uncertainty and risk are eliminated through rigorous offline validation and verification.

4.3 SAPC development according to ISO 26262

One of the main challenges for the SAPC development is to carry it out in such a way that it can be reused in different platforms. To do so, a Safety Element out of Context (SEooC) approach which is defined in ISO 26262-10, has been pursued and a general error handling defined. Figure 40 illustrates the steps followed when establishing the SAPC component. In this section, the focus lies on the left part i.e., the SAPC component development steps.

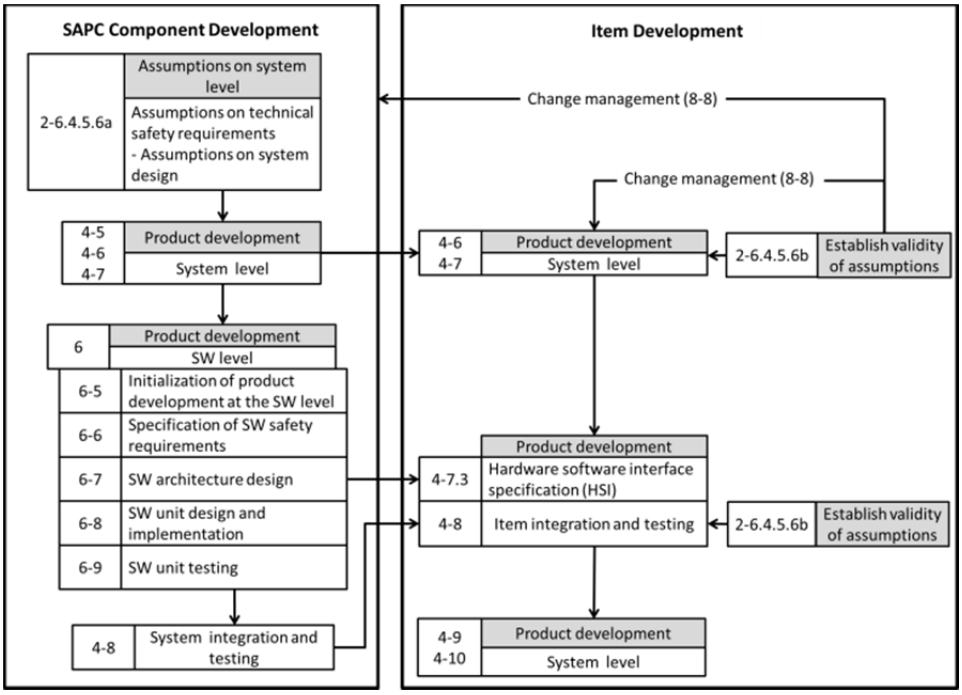


Figure 40: SAPC component development from ISO 26262

In order to share the same understanding of the standard, both the SEooC development team and the item development team need to be aligned. For this, the guidelines shown in [6] were modelled and then tailored to the SEooC needs and the requirements with respect to adaptation.

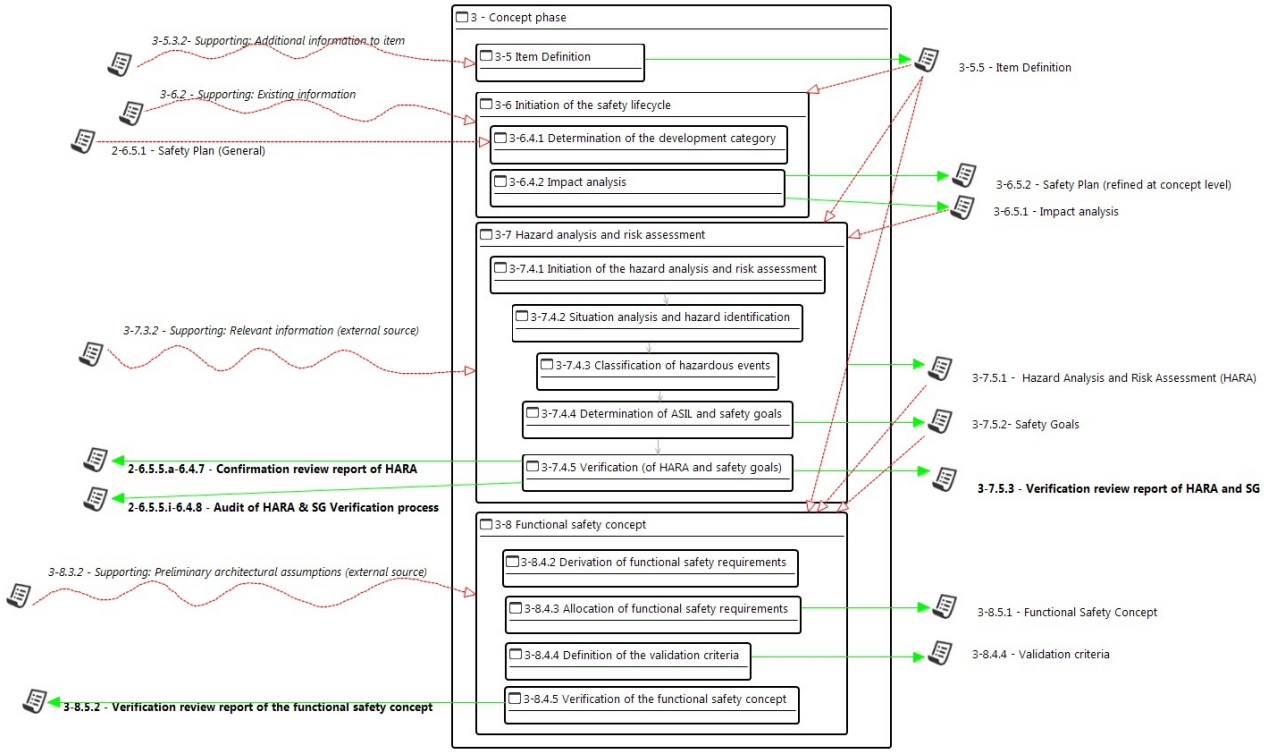


Figure 41: Excerpt of the ISO 26262 standard model

In the deliverable D3.3 [5] challenging parts in terms of ISO 26262 compliant fully self-adaptive systems have been introduced. Some of the recommendations made in D3.3 and D4.2 to tailor the ISO 26262 model have been taken to the specific requirements for the SAPC development. One of the first changes made were the inclusion of two new requirements associated with the Hazard Analysis and Risk Assessment (HARA) phase:

- Consideration of adaptation scenario during the hazard identification task
- ASIL assigned for the SAPC should be based on the highest ASIL associated with the application affected by the adaptation

Further, a model based approach has been followed, where the phases “software safety requirements”, “software architectural design” and “software unit design and implementation” have a stronger coalescence according to ISO 26262. With this in mind, the EAST-ADL language in combination with the UML MARTE profile has been selected. In this sense, an early validation of the design model has been applied especially for non-functional properties related to adaptation such as timing constraints.

Another important tailoring to the requirements for the standard compliance is related to safety validation. A system level Failure Modes and Effects Analysis (FMEA) has been performed and used as a reference fault model in order to verify the SAPC implementation and its corresponding safety/dependability requirements with respect to permanent faults at system level.

As previously stated, sometimes it is necessary to develop safety-related systems when all the needed data is not yet frozen. Following this approach the SAPC development requires an ISO 26262 compliant SEooC process which is based on assumptions on an intended functionality, context and use, including the assumptions of safety requirements. Later on, once the SEooC is integrated, all the corresponding assumptions will be verified on the actual item.

The SAPC includes a safety case as required by the standard which summarises the safety argument supported by adequate evidence. For this, a modular argumentation technique was

applied. The safety case architecture shown in Figure 42 illustrates the boundaries of the SAPC and the information requested by other elements from the final item. The green modules in the figure correspond to the SAPC argumentation and it is decomposed into M1.1 and M1.2. These modules require the validity of certain claims that have been assumed related to the hardware platform and the applications that are susceptible to be adapted. In this case, the assumed claims are planned to be included in M2 in terms of the target platform in which the SAPC software is going to be deployed. In addition to this, M3 is composed by the ones related to the applications susceptible to be adapted.

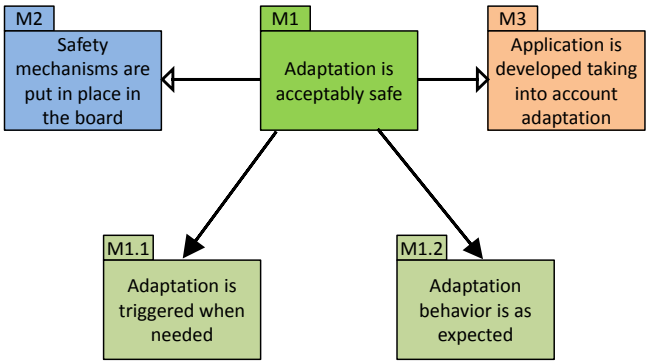


Figure 42: SAPC Safety Case Architecture

Validating the concept of the SAPC developed as SEooC

For validating the concept of the SAPC developed as SEooC, it was integrated within RACE and TMDP platform. For this integration, the “item development” activities shown on the right part of Figure 40 must be performed. The integration should take into account requirements from the standard as well as technical integration. Here, the focus is on integrating the SAPC into this target platform and validating the assumptions. With respect to the steps shown in Figure 41, the target further lies on connecting arrows between the SEooC development and the item development. This includes the task of establishing validity of the assumptions in order to comply with the requirement clause 2-6.4.5.6b of the ISO 26262 standard.

Previously, an assurance project with respect to the SAPC development has been specified. In this assurance project, all the evidences are traced to ensure the fulfilment of the requirements to comply with the standard as well as a safety case for the SAPC. Afterwards, a second assurance project has been created, which focuses on compliance between requirements and the ISO 26262 standard in relation with the SEooC integration. As mentioned before, the applications development and the platform development are considered black boxes and for the SEooC integration only the assumptions made on the SAPC (functional safety requirements and safe operational behaviour) and the guarantees offered (fault detection mechanisms and notifications through APIs to the SAPC) are visible.

In addition, some activities for the correct integration have been performed. Here special attention was paid to the safety validation where a new activity has been included for adaptation behaviour simulation on a target environment. In order to ensure the adaptation timing for critical applications, a vehicle dynamics simulation based analysis was also performed with the Dynacar RT tooling.

As seen in Figure 40, the validity of the assumptions must be established during the item development phase illustrated on the right side. For explanatory purposes Table 9 shows the validation done for the TMDP. In case the assumptions are not valid, either the SAPC or the platform needs to be changed. During the vehicle development, part of the TMDP behaviour had to be modified in order to apply and develop the interface to communicate the faults of the platform to the fault filter block in the SAPC.

Assumptions made by the SAPC	Validation within TMDP and critical applications
A Core Node failure shall be detected by a lockstep mechanism	TMDP uses a hardware lockstep to detect a failure on the platform
A memory failure shall be detected	TMDP uses a MPU (Memory Protection Unit) to detect memory failures
A timing failure shall be detected	TMDP uses a SoC internal safety watchdog with protected register sets to detect corruption of configurations
	TMDP uses an external chip performing watchdog functions to detect system timer stops
A SoC bus system failure shall be detected	TMDP includes a CRC mechanism on the SoC bus system to detect timing failures
A power supply failure shall lead to fail silent behaviour of the node	TMDP includes an external safety monitor to detect a power supply failure and then fail silently
A sensor failure shall be detected by input comparison	Input loss from a sensor is detected
	A sensor failure is detected by input comparison
A network failure shall detected	A network failure is detected by input comparison
There shall be plausibility checks at application level	There are plausibility checks at the critical application level
Application software shall be verified	Applications have been developed in compliance with ISO 26262 which requires software verification
Applications shall perform the functionality as specified	Applications have been developed in compliance with ISO 26262 and functionality has been verified
System architecture shall have redundant sensors for safety-critical application	The system architecture used a 2 out of 3 architecture pattern for critical sensors
System architecture shall have redundant actuators for safety-critical application	The system architecture ensures a 2 out of 3 architecture pattern for critical actuators
Transient faults shall be covered at ECU level	Transient faults are addressed at TMDP platform level
A platform shall not be woken up via the second processor board	The TMDP is not capable to be woken up via the second processor board
Overall time response for safety-critical functionality should be less than 50ms (fault detection 10ms; passivation 10 ms)	The failover simulations (see section 3.3.2) show that with a fault duration > 250 ms the driving is unsafe, and that the drivers starts perceiving the performance degradation when the fault duration is 150 ms, thus 50 ms provides a safe margin
The restart of the system by an ignition-cycle shall lead to system sanity check	A system restart by an ignition-cycle leads to a BIST (Built-in self-test) (core check, walk pattern test, etc.) to check the sanity of the system. In case of no error was found, the original run-schedule shall be started
The connection between the different platforms shall use redundant paths	The connection between the different platforms uses a redundant communication architecture

Table 9: Validation of safety assumptions

5 Evaluation of Efficiency

The previous chapters showed the functional integrity of the SafeAdapt approach. However, this does not indicate how the approach compares to others. To provide a baseline for comparison and to show that the effort to use the SafeAdapt technology is manageable and can even reduce the effort compared to other approaches, the evaluation methods described in D5.1 [1] are used to calculate the measurable results (MR). Thereby, the efficiency of the SafeAdapt approach is verified. First, this chapter details the properties of the system components used as basis for the evaluation. Next, a brief update of the architecture overview compared to D5.1 is given. Finally, these numbers are used to provide the measureable results.

5.1 Basics for the Evaluation

This section describes the properties of the components used to perform the evaluation of the measurable results. Where possible, the specific properties of the components actually used in the project are given. Otherwise, the properties of a component typically used for this purpose are considered.

Core Node

For SafeAdapt it was planned from the beginning on to use the TMDP. This board is a typical prototype platform to make a proof of concept or to evaluate architectures in a very early stage of a project (“rapid prototyping”). To estimate SafeAdapt’s effect on vehicle architectures, the first step is to perform a layout estimation and to find an applicable housing for such a kind of ECU. This was done within the project and the result can be found in the following Figure 43 and Figure 44. With this layout estimation the size of a TMDP can be assumed as 174.6 cm² plus the size for the mounting shoes.

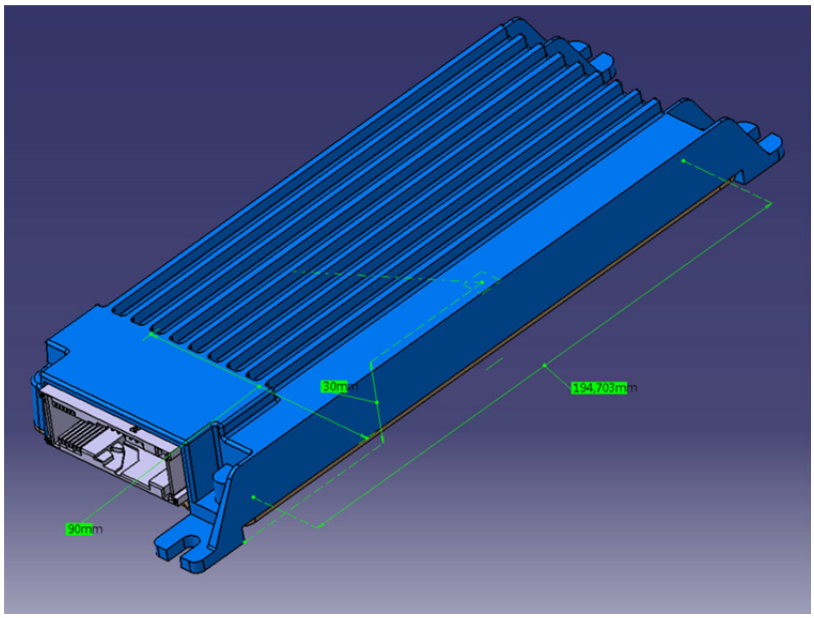


Figure 43: Dimensions of a “production ready” TMDP

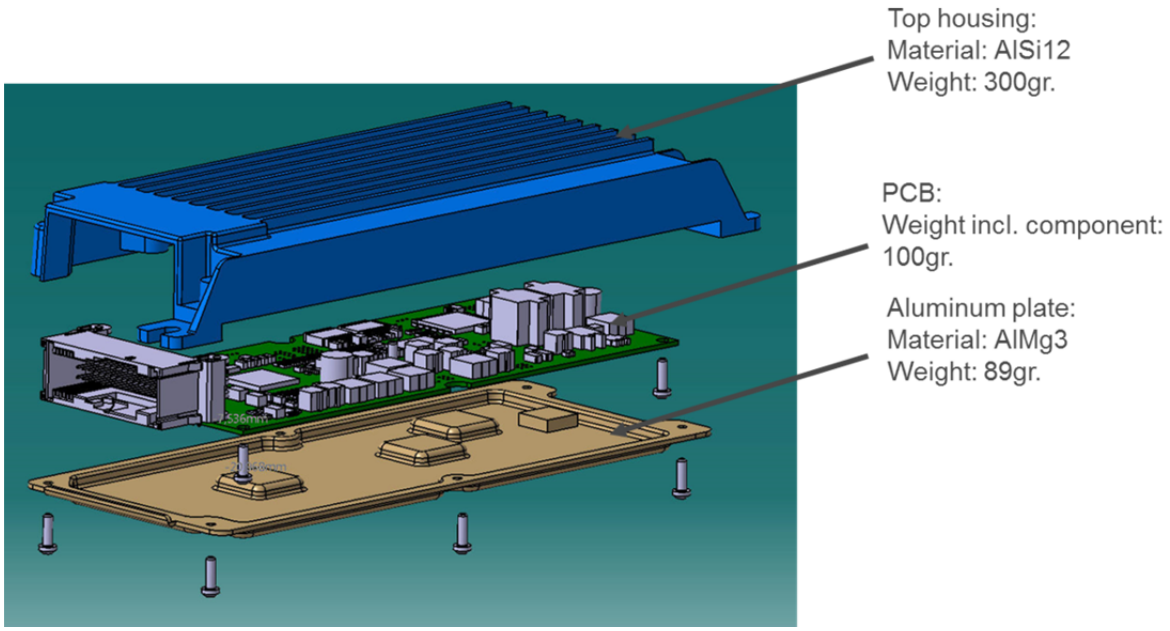


Figure 44: Weight of a “production ready” TMDP

Body computer

For the estimation of a state of the art system, a typical body computer was taken into account (see Figure 45). Due to the fact that the number of loads in the rear of the car is less than in the front of the car, the rear body computer will look slightly different. The dimensions for this kind of ECU are given by 23.6 * 15.6 cm = 368 cm^2. The weight is 510 gr.

Figure 45: Typical front body computer (size: 26.2 * 18.9 cm = 495$cm^{2,}$ weight: 790gr.)

Load box

For the SafeAdapt architecture the body computers will be changed to load boxes. The main difference is the kind of processor that will be needed. It can be significantly smaller due to the fact that the algorithms will run on the Core Nodes (CCC). The number of relays and solid state drivers will stay the same. Therefore, the cost of a load box will be slightly lower than the cost for a body computer. The differences in size and weight can be found in Table 10.

Wiring

To evaluate the difference in the wiring the following assumptions were made

I) Body computers are one by one replaced by load boxes. This means, there is no difference in the power wiring from architecture to architecture.

II) Ethernet connection: The number of dedicated Ethernet connections will stay the same in both types of architecture.

III) CAN connection: The number of dedicated CAN connections will stay the same in both types of architecture.

5.2 Fail-Operational Architecture Overview

Deliverable D5.1 [1] provided already an overview of a hypothetical state-of-the-art reference vehicle architecture. However, to provide a sound comparison, a minor update had to be performed to introduce a CAN bus into the architecture. The updated block diagram of the architecture without SafeAdapt technology is shown in Figure 46.

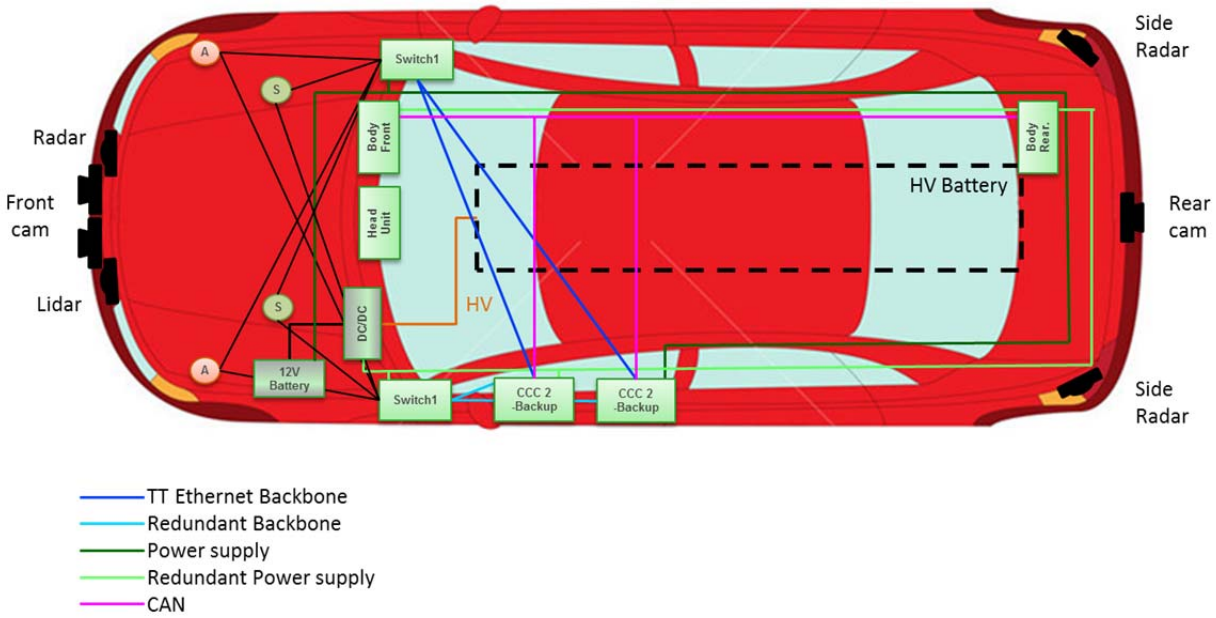


Figure 46: Updated block diagram of fail-operation architecture without SafeAdapt

With respect to the technology developed by SafeAdapt, an additional vehicle architecture diagram is shown in Figure 47. The main difference between the two systems is that the “spare” ECU is eliminated. The head unit is now in the SafeAdapt architecture the instance to take over this responsibility. As this unit will always be part of a vehicle because of its HMI functionality, the additional Core Node can be eliminated from the architecture.

Despite this, one problem with the current generation of head units is that they are not developed to host application up to ASIL D and exactly this is needed for this SafeAdapt architecture. However, the trend towards virtual clusters in the automotive industry will likely change this fact. This means, that more and more clusters with analogue gauges will be replaced with TFT screens in modern cars, and as such the head unit will become safety-critical because it will be part of the safety strategy to inform the driver. Therefore, it is also expected that future head units will have at least a safety island. With the very same mechanism of SafeAdapt it is of course possible to easily extend such a system with an additional ECU. As more calculation power is available in the system, more adaptation scenarios are possible, e.g., it would be possible to establish the redundancy after an adaptation again.

For the comparison of the architectures it is sufficient to stay with a simple system consisting of only two Core Nodes. An additional ECU will mainly increase the effort for configuration of such a system and the software development, but not change the metric-relevant facts.

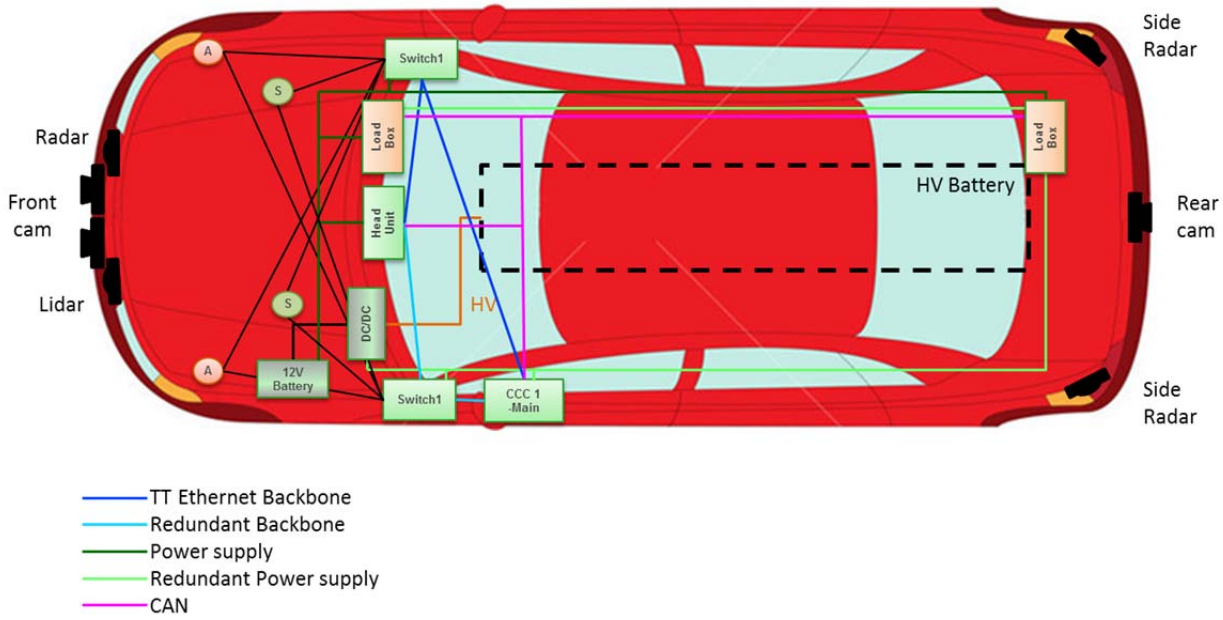


Figure 47: Block diagram of fail-operation architecture with SafeAdapt

With the updated information provided in the previous section, the properties of a fail-operational architecture with and without SafeAdapt's adaptive technology can be approximated as shown in the following Table 10. All values are based on experience with comparable ECUs. The costs are given as relative values because the exact values are confidential numbers of the partners. The meaning of these numbers is, if the cost of one Core Node is rated as 100%, then the Body computer price is around 70% of the cost of the Core Node cost.

System properties with and (without) SafeAdapt technology						
	Cn 1 (CCC)	**Cn 2 (CCC)**	**Load Box / Body comp. Front**	**Load Box / Body comp. Rear**	**Head Unit**	**Wiring**
Cost	**100%** (100%)	**0%** (100%)	**65%** (70%)	**55%** (60%)	**110%** (110%)	**150%** (150%)
Weight incl. Housing, brackets, connectors	**489g** (489g)	**0g** (489g)	**780g** (790g)	**500g** (510g)	**950g** (950g)	**700g** (700g)
Size	**175cm²** (175cm²)	**0cm²** (175cm²)	**490cm²** (495cm²)	**363cm²** (368cm²)	**355cm²** (355cm²)	n/a
Power consumption	**25W** (25W)	**0W** (25W)	**19W** (20W)	**14W** (15W)	**40W** (40W)	n/a

Table 10: Properties of a system with SafeAdapt (bold) and state-of-the-art system (in brackets)

5.3 MR1: Optimised Energy Consumption

The adaptation mechanisms provided by the SafeAdapt approach can be used to increase the energy efficiency, thereby, increasing the possible range of the vehicle. Section 3.3.3 describes how Dynacar was used to simulate the New European Driving Cycles (NEDC) to estimate the car's energy consumption.

If the SAPC switches off auxiliary systems like the air conditioning and multimedia, these subsystems will drop their power demands from 3.69 KW to 0.69 KW. For one NEDC cycle, this implies a total difference of 1 KWh. However, when applying the torque reduction criterion (torque degraded to 50% original), there is no significant effect on the vehicle's energy efficiency experienced, as it only adds 0.9 Km to the vehicle's range. Therefore, this should be avoided and there would not be any performance degradation in a slope or when overtaking. In consequence, the only implication in a slope should be related to the change in the braking distance. However, this it is not an issue in general because a slope sensor could be used to adjust the ideal braking curves.

To increase the energy efficiency, the utilisation of the generator should be as large as possible to reduce dissipation of kinetic energy in form of heat. Nevertheless, when shifting the brake balance, the vehicle dynamics must remain safe. Therefore, the maximum brake forces that could be applied in the frontal and rear axles of the vehicle were analysed. Assuming rigid body conditions as a reference starting-point, the vehicle has been modelled with its dimensions, weight and CoG (Centre of Gravity). The brake distribution curves obtained from the analysis in Matlab are shown in Figure 48.

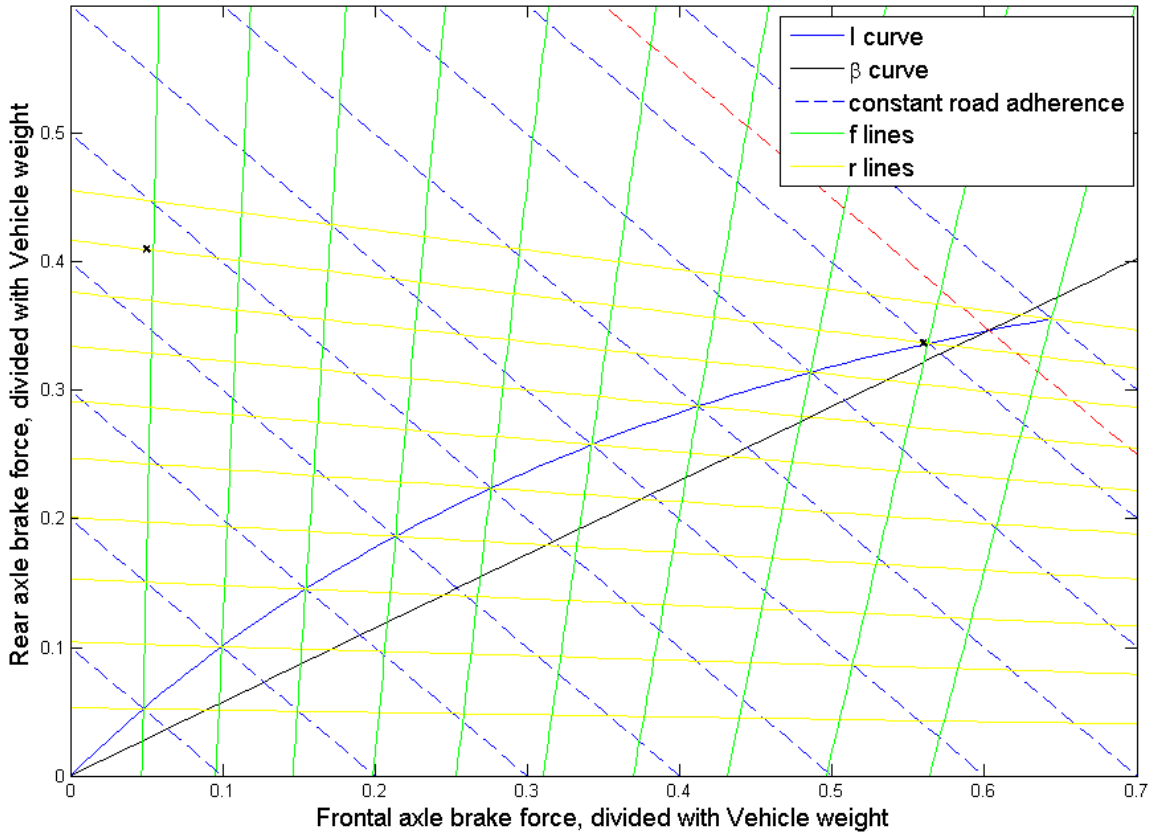


Figure 48: Braking force limits applied on the frontal and rear axle of the vehicle

The value of 0.9 has been selected as a reference of the adherence coefficient and the braking force distribution is obtained from previous measurements (respect to sum of the normal forces applied to the vehicle). In the Figure 48, the ideal performance curve ("I curve") assumes that the frontal and rear axles lock-up at the same time, so the braking distance is minimal, i.e., normal operation mode. The "f lines" and "r lines" plot the maximum brake forces for the frontal and rear wheel lock-up respectively. From this plot, the brake distribution has been selected to perform with a 0.9 adherence coefficient and 2 operation limits:

- Operation in the "I curve" (Shortest brake distance): Rear wheels 41% and frontal wheels 5% of the vehicle weight. In this operation limit, the Dynacar simulation result shows a power consumption of 1.58 KWh and a total range of 127.3 Km (without auxiliary power demand).
- Operation in the "r curve" (Maximum rear brake force to rear axle lock-up): Rear wheels 34% and frontal wheels 56% of the vehicle weight. In this operation limit, the Dynacar simulation result shows a power consumption of 1.42 KWh and a total range of 141.6 Km (without auxiliary power demand).

The selected optimal brake distribution strategy for range extension does not follow the typical design criterion, where the frontal axle wheels could lock-up if road adherence falls from 0.9 to 0.5 (e.g., due to rain). Instead, if the adherence falls and the brake demand is maximal, the rear wheels lock-up first and by consequence there is a stability loss issue. However, as adaptation is available, the road adherence could be measured by a sensor or estimation method and adjust the brake distribution accordingly in order to maintain safe dynamic performance. This could be an interesting future development because current research showed that there is a considerable range increase in the energy optimised mode.

In sum, the following is concluded from the combined evaluation of improved energy efficiency:

Mode	Driver Train per NEDC (kWh)	Auxiliary Systems per NEDC (kWh)	∑ Power Consumption per NEDC cycle of 10.39 km (kWh)	Range with 19.4 KWh battery (km)
Normal	1.58	1.23	2.814	71.6
Optimised	Ø 1,519	Ø0,868	Ø 2.373	89.46
SOC > 35%	1.58	1.23	2.814	46.56
SOC <= 35%	1.406	0.23	1.636	42.9

Table 11: Vehicle energy efficiency with adaptation

With SafeAdapt the vehicle range could be extended to 89.46 km, hence, it increases by 17.9 Km or **25%**. These calculations assume that when the battery SOC is higher than 35% there is a high use of air conditioning with a continuous power demand of 3 kW. In the case of low demand of auxiliary power, the range extension is still significant, from 127 Km to 142 Km (+12%). It should be noted that if using the SafeAdapt approach, the 25% increase in energy efficiency by adaptation are achieved on top of the 21.6% reduction of energy consumption in the vehicle's architecture.

5.4 MR2: Failures Handled by Adaptation

It is worth noting that current systems usually follow a fail-safe approach which means that in case a failure is detected, the function is either degraded for a limited time (e.g., braking, electronic power steering) or deactivated (e.g., adaptive cruise control, lane assist, airbag, air conditioning) until the safe state is reached. This fail-safe state usually consists of a complete system shutdown, as it is not possible to determine the valid output and, consequently, this strategy it is not acceptable for autonomous driving.

Consequently, emerging autonomous driving technologies require fail-operational functionalities. To do so, approaches like 2oo3 may be cost prohibitive in the automotive domain. SafeAdapt provides a vehicle-level E/E solution for generic or function-unspecific fail-operational or graceful degradation where the 1oo2D fault tolerance pattern is deployed. The introduced SAPC handles failures in a generic way, being capable of providing fail-operational functionality for the most critical vehicle functions such as Steer-by-Wire and Brake-by-Wire. Regarding functionalities with less criticality, the adaptation is used to enhance the "mission time" of the car.

In fact, each function is considered as a composition of fault containment regions (FCR) as described in D3.2 [7]. The information of which FCRs compose a certain function and which system resources are required by it, needs to be provided as a configuration parameter [8]. This scalable fault-tolerant E/E-architecture is based on TMDP and RACE platforms which ensure fail-silent functionality at component level and rely on different means for performing diagnostics. Different hardware and software safety mechanisms are embedded in each Core Node and a set of fault containment regions are defined at different levels. These built-in HW/SW safety mechanisms are able to either detect or correct faults related to each of the FCRs. If an error cannot be corrected at ECU level, the SAPC is notified to handle the failure in an appropriate way.

More specifically, the SAPC is able to handle platform failures, memory failures, power supply failures, sensor failures, and network failures and guarantees the fault tolerance of the aforementioned critical functions in the vehicle E/E systems. Following list summarises the types of fault regions that the SAPC concept can handle:

- *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 drops, that will make the Core Node reset.
- *Application failure*: there is a failure on the application due to unavailability of the input data or a random hardware failure makes it impossible for the application to run in a normal mode. Software design failures are not covered.

For further detailed information about the error handling strategy at system level, the following Table 12 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 reactions for each case are provided.

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 12: Error handling strategy at system level

In sum, any single failure within the system can be handled by the SafeAdapt approach, thus, proving **100% safety from random hardware failures** for functionalities with fail-operational requirements.

5.5 MR3: Cost Reduction

In Section 5.1 the cost of fail-operational systems with and without SafeAdapt's technology was already compared, coming to the conclusion that hardware costs will be reduced by 25% when creating fail-operational systems with the proposed adaptation concepts.

Next to these cost savings, and additional study on improved cable routing was conducted to determine if these costs could be reduced even more without endangering the safety of the system. Here, the focus is on creating an architecture that ensures the E/E-architecture's reliability while reducing the electrical distribution system complexity, wiring harness length, circuit breakers, connectors, ground points, shielding, fixing points, and mechanical wire protection.

For this study, the electric distribution function requirements together to legislative and manufacturing requirements were considered. In comparison with state-of-the-art E/E-architecture layouts for fail-operational systems (see Figure 49), the economic advantage of this enhanced architecture (see Figure 50) is around 30%. These results were obtains through the concentration of high voltage vehicle traction circuits in the rear engine compartment while the 12 Volt electrical/electronic circuits and the SafeAdapt electronics were placed in the front luggage compartment of the vehicle with the benefit of shorter wiring routing and better electromagnetic compatibility management.

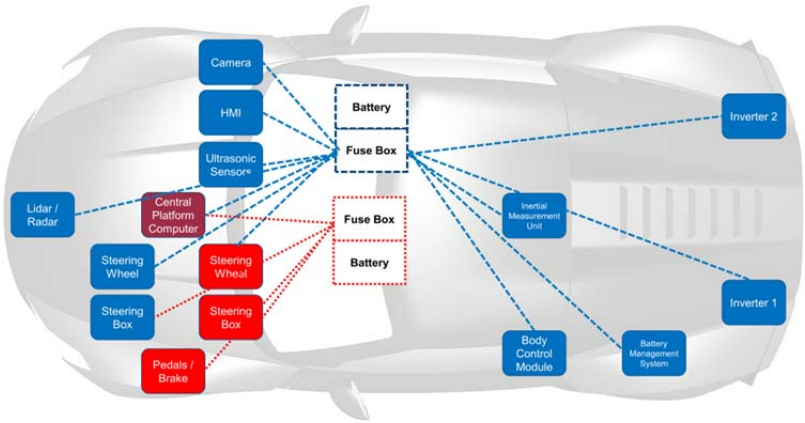


Figure 49: State-of-the-art fail-operational wiring harness layout

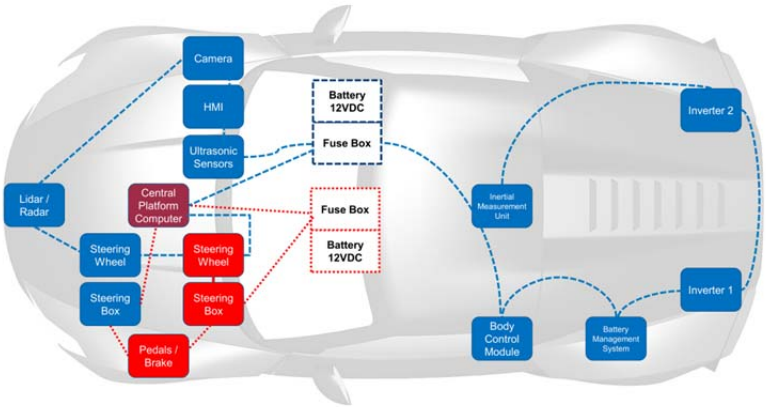


Figure 50: Reduced wiring harness layout

5.6 MR4: Reduced Certification Cost

As mentioned in the previous sections, one of the main challenges which SafeAdapt is facing is to check the feasibility to certify the new reference architecture resulting from the project. This is especially difficult as the outcomes from the project are prototypes and not market products. Thus, it is out of the scope of the project to certify the solution. Estimations have been used from public information in other domains such as aviation and from the automotive domain in order to identify four perspectives to evaluate the possible implications:

- Reusable system architecture
- Tool qualification effort
- Safety goal verification effort
- Functional safety management effort

5.6.1 Reusable System Architecture Perspective

As a result of the project, the so-called SAPC has been developed as a system level safety mechanism. The SAPC allows reuse in a safe manner without the need for a redundant ECU per vehicle function and provides a generic fail-operational behaviour for safety-critical applications.

For estimations the following assumptions are considered:

- Assume the integration of only X-by-Wire related applications with an ASIL D associated
- Assume integration of related functions of equal size & complexity with 25% error margin
- The ASIL compliance cost overhead is considered as a median between the minimum and maximum cost

According to [9] and [10] we can estimate the cost of the software development:

Basic SW Cost= 3.35 * HW Basic Cost

This figure is in the following only used as basic development cost as it does not take in to account the overhead for ASIL compliance. In order to take into account the overheads for ASIL compliance, the estimations from [11] have been chosen, where it is said that the overhead is related to the experience of the team. For further estimations the mean of the overhead is utilised:

Cost overhead	ASIL A	ASIL B	ASIL C	ASIL D
Min	5%	10%	20%	40%
Max	20%	36%	60%	100%

Table 13: ASIL overhead

According to [12], the estimated cost of an X-by-Wire system is around 10000$. Based on this number we can allocate the cost of the software and the hardware. Moreover, at PRICE Systems company, they are focused on Predictive Analytics for Improved Cost Management [13]. They have developed a model to estimate the cost of compliance with the avionics reference guidelines and standards such as ARP-4754, DO-254, and DO-178b/c. They mention that the cost of the hardware and software development is directly related with the complexity. That is also the exact

rationale behind SafeAdapt's estimations where the cost of developing a piece of hardware or software is multiplied by the level of complexity. To check the complexity of the hardware, the same effect estimated by [9] about integrating additional functions in an IMA (Integrated Modular Avionics architecture) are applied (see Figure 51 and Figure 52).

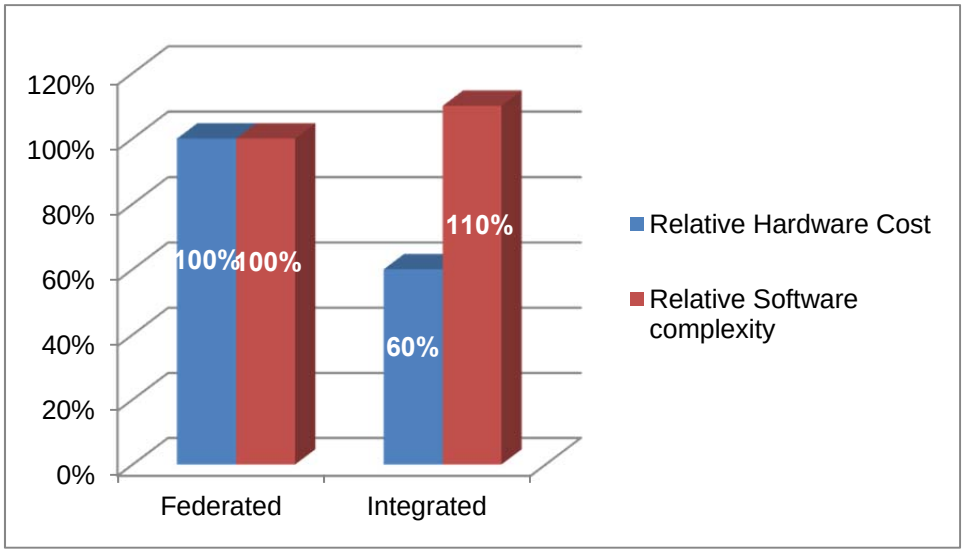


Figure 51: IMA enclosure + 1st application

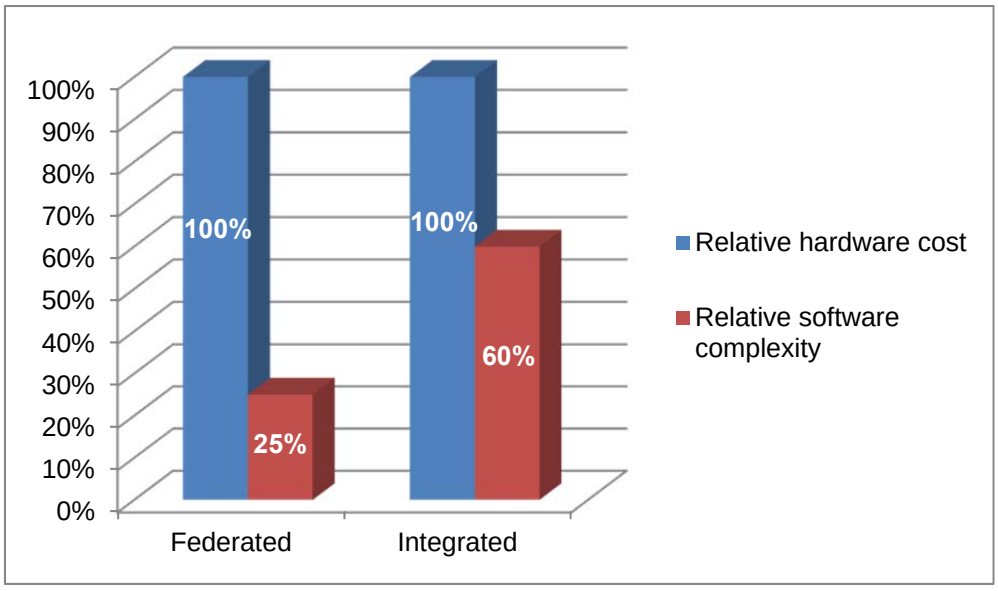


Figure 52: Each additional application

Based on the previous estimations gathered from literature, the cost per vehicle function that requires fail-operational behaviour has been estimated in Figure 53. Based on this result, it is obvious that the SAPC architecture is not the most suitable solution for single functions. However, the more functions are integrated (e.g., >3 functions), the more efficient it is.

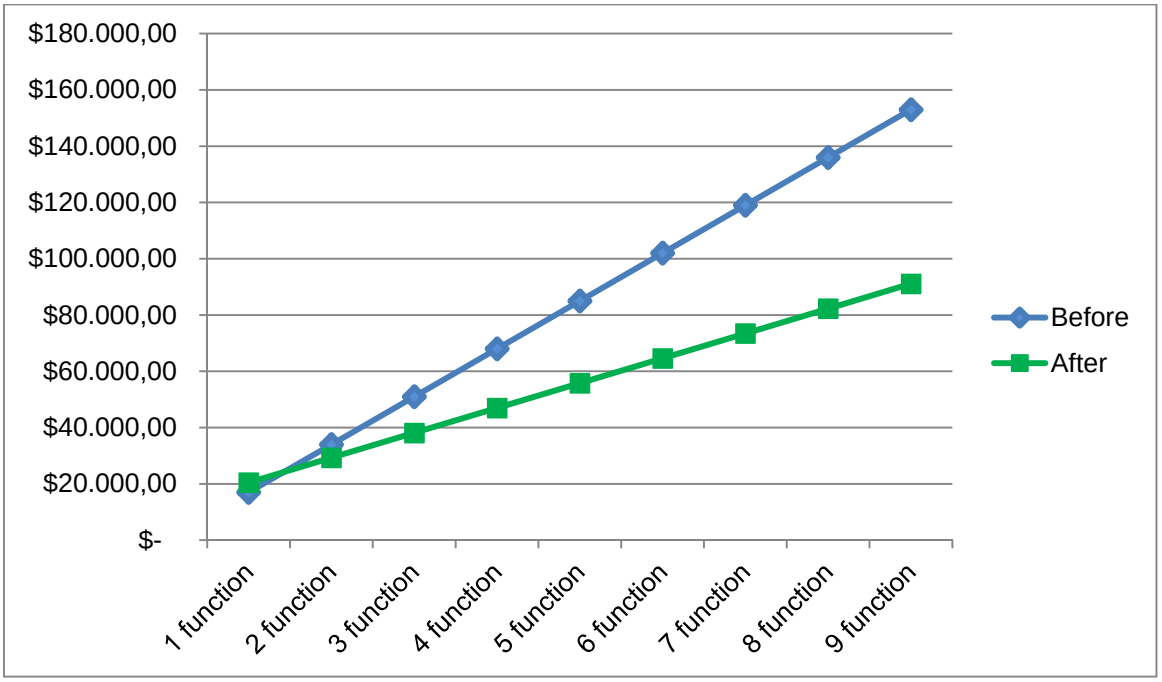


Figure 53 Estimations of SAPC architecture cost per application included

5.6.2 Tool Qualification Effort

During the SafeAdapt project, a tool chain together with a methodology has been specified. This tool chain provides modelling, design, and V&V support. It uses a model-based design flow, which is complemented by pre-existing AUTOSAR tool-chains. Moreover, this approach enables early verification and validation of non-functional requirements such as adaptability to support full certification for safety-critical systems. Among others, the major results expected from its usage are reduction of the development and testing costs and reduction in certification cost.

Current ISO 26262 methodologies and tool chains do not fully address the automotive functional safety standard from a global perspective. Recent studies such as [14] identify a need for reference tool chains and guidelines from which characteristics for tool qualifications can be derived. Even if there are supporting software tools for lifecycle activities, i.e., item definition, hazard and risk analysis, safety analysis, functional safety concept, or safety validation, there is no unified framework addressing the entire lifecycle [15]. In other words, current tools and methodologies cover only parts of the standard. This leads to error insertions due to badly integrated tools [16].

SafeAdapt has been working in providing a reference tool chain in accordance with ISO 26262 together with a methodology specially designed for adaptive systems. On the scale of current model-based safety analysis techniques [17], qualitative ones are often performed at the expense of quantitative techniques. ISO 26262 standard highly recommends quantitative safety analysis for high-level safety-critical applications. In this context, Siemens FMEDAexpress tool performs both qualitative and quantitative safety analysis based on system and component models [18].

Moreover, adaptive systems have not been in the scope of ISO 26262 when the standard was written and SafeAdapt methodology provides guidelines for ISO 26262 compliance when dealing with adaptive systems such as the SAPC. The certification process might become even more difficult and expensive, since adaptation is one way to mitigate the effect of failures and increase the availability of a system. In this area of application, there is less experience related to the use of ISO 26262. Furthermore, this standard provides no support for runtime hazards or an analysis of adaptation scenarios. Current software architectures only consider static reconfiguration with a

fixed set of modes that rely on static safety validation, e.g., as can be seen in [19] [20]. However, in adaptive architectures the number of possible system states and transitions cannot be determined in advance.

In order to enable certification of adaptive automotive systems, dependability and predictable behaviour of safety-critical applications, such as X-by-Wire systems, must be guaranteed at runtime. Therefore, the aim here is for a robust and comprehensive ISO 26262 methodology, supporting a semi-automatic tool chain for adaptive systems reducing tool qualification and certification costs. Since suitable methods and techniques are required to evaluate the dependability of adaptive modular run-time control systems.

Tools for model-based development are especially adequate to reduce the effort requirements in software development and verification tasks associated with standard compliance. Even more, according to [21] “model-based development is estimated to bring a cost savings of approximately 15-20% on the software application verification (which represents the main cost driver of certification).” Beyond all possible doubt, the cost of qualifying a tool is quite high. As stated in [21] just the qualification and the development of an automatic code generator is estimated to be approx. 2M€. It is foreseen that similar budgets will be necessary to actually qualify each tool. These costs are related to the avionics domain, so they are not directly comparable to automotive domain, where budgets are smaller. Nevertheless, according to [16] the cost of qualifying a tool chain is less than qualifying each of the tools independently.

5.6.3 Safety Goal Verification Effort

As previously pointed out, the SAPC has been developed following a Safety Element out of Context (SEooC) approach. This means that it has not been developed for a particular vehicle but considering the different environments where the SAPC might be deployed. This SW SEooC can run on different HW platforms, as it has for instance been deployed into the RACE and the TMDP platforms. Furthermore, it can adapt to different application functions. Following this approach, the SAPC development requires an ISO 26262 compliant SEooC process which is bases its assumptions on an intended functionality, context, and use, including the assumptions on the safety requirements. Later on, once the SEooC is integrated, all the corresponding assumptions shall be verified.

Further, the SAPC includes a safety case (see Figure 42) as required by the standard which summarises the safety argument supported by adequate evidence by means of modular argumentation techniques. To some extent, the main goal of this safety case architecture is to define the strategy for what 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 the safety goals are correctly addressed.

The use of modular argumentation introduces some benefits to the development according to [22]:

1. Modular safety case development supports work division and work sharing
2. An explicit modular safety case structure helps tracking dependencies between arguments
3. An explicit modular safety case structure reduces the rediscovery and review effort (e.g., the effort associated with locating and tracking inter-safety case dependencies)
4. An explicit modular safety case structure limits the effects of change
5. Modular safety case development improves top-level planning of the safety
6. An explicit modular safety case structure promotes reuse within, and across, argument application
7. An explicit modular safety case structure helps manage organisational and/or contractual safety case boundaries.

In [22] is also stated that the use of a modular approach in the safety case development process would require an increase in up-front project safety case costs of around 25%. It has been generally recognised that modular safety case development would result in increased initial costs associated with establishing a modular structure and identifying interfaces and tool support would need to be acquired. This cost has already been assumed in the SafeAdapt project. The SAPC safety case already includes the specification of the interfaces in form of assumptions and public claims or guarantees. [22] also mentions that a long-term cost benefit of 11-50% of current safety case development costs is expected from using a modular approach based safety case development.

5.6.4 Functional Safety Management Effort

Functional safety in smart cars such as the one targeted in SafeAdapt, needs to comply with ISO 26262. In Table 14 we have quantified the number of activities and work products resulted from the performance of the mentioned activities.

ISO 26262 Parts	No. Activities	No. Work products
Management	3	8
Concept	3	6
System	6	12
HW	6	11
SW	8	18
Production	0	0
Supporting	5	9
ASIL-oriented & safety analysis	4	5

Table 14: Number of activities and work products in terms of SafeAdapt (ISO 26262)

In SafeAdapt a semi-automatic tool chain was created together with a methodology which aims to guide the user using the tools along the development lifecycle and according to ISO 26262 "Road vehicles – Functional safety requirements. Figure 23 illustrates an overview of the tools used and in [23] it has been specified how each of the tools under analysis supports the ISO 26262 activities. Following the guidelines, the activities and work products which are supported by tools from the SafeAdapt tool-chain are extrapolate below:

ISO 26262 Parts	Activities Tool supported	Work products supported by tools
Management	3	6
Concept	3	6
System	4	8
HW	3	5
SW	4	5
Production	0	0
Supporting	2	2
ASIL-oriented & safety analysis	4	5

Table 15: Number of activities and work products supported by SafeAdapt tools (ISO 26262)

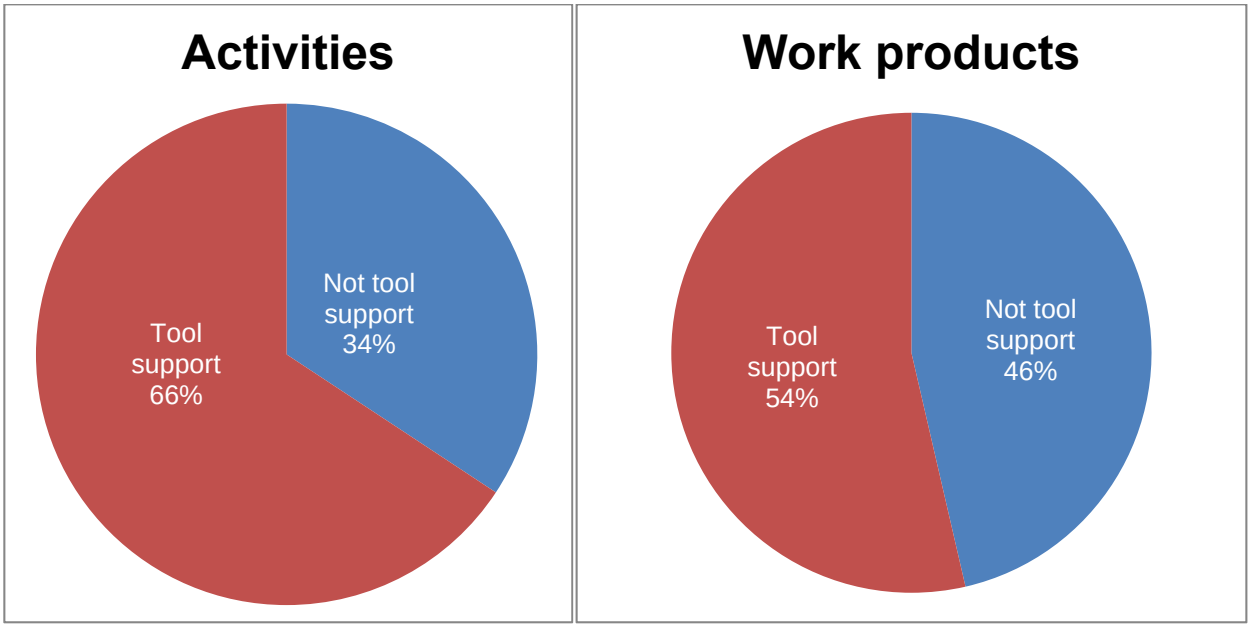


Figure 54: Relation of activities and work products

5.7 MR5: Reduced Complexity

As the complexity of a system spans a variety of aspects, this measurable result is a composite figure derived from the preceding evaluation results. Thus, the following Table 16 just summarises all categories in which complexity can be reduced by applying SafeAdapt's method for developing adaptive and fail-operational systems.

Category	Measurable Results & Description
Cabling	- 30% less cables - Shared infrastructure (e.g., Ethernet) simplifies cable routing
Hardware Diversity	- Two different types of generic computing units and load boxes can safely replace a multitude of function-specific ECUs
Certification	- SAPC as Safety-Element-out-of-Context (SEooC) may be only certified one, reducing complexity of recertifying fail-operational logic for every functionality
Software	- Reuse of generic fault handling (SAPC) prevent reimplementation of failover logic in every application individually - Design support and automation through toolchain reduces manual effort and sources of human error

Table 16: Categories of complexity reduction

5.8 MR6: Improved Redundancy Concept

As the current state-of-the-art system design can only attain fail-operational behaviour in a non-generic manner through wasteful duplication of components, SafeAdapt's adaptive approach for managing redundancy in E/E-architectures poses as a promising alternative. This insight is further underlined when analysing the measurable results of Section 5. In sum, SafeAdapt's adaptive technology allows substantial saving in cost, weight, size, and power consumption when compared to current state-of-the-art approaches. The results are summarised in the diagram in the following Figure 55:

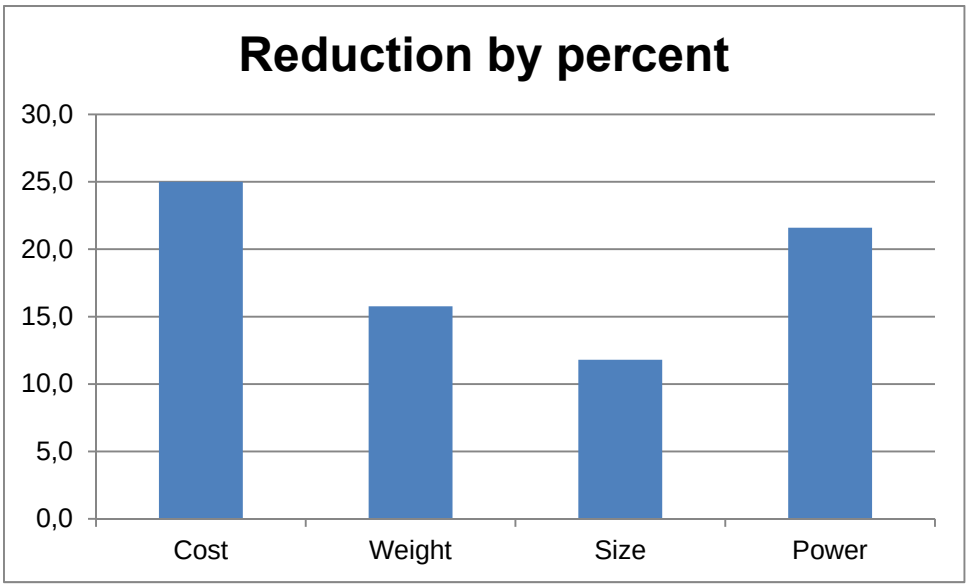


Figure 55: Measurable improvements of SafeAdapt compared to state-of-the-art

It can directly be derived that in the evaluated architecture scenario introduced in Section 5.2 all considered characteristics can be significantly reduced. Most reduction of 25% can be calculated for cost, followed by a reduction of around 21,6% of power. Weight can be saved up to 15,6% and size / volume about 11,8%. With these results the improvements of the SafeAdapt redundancy concept can prominently be proven.

6 Summary

The upcoming generations of vehicles will implement more and more novel functionalities electronically. As many of these functionalities, such as Automated Driving or Steer-by-Wire, are not allowed to fail when a single component fails, a fail-operational E/E-architecture is essential. With state-of-the-art technology, such fail-operational behaviour is currently only attainable by adding multiple special-purpose ECUs within a vehicle's network to compensate for a potential failure. Compared to this wasteful approach for fail-operational E/E-architectures, the adaptive method proposed by SafeAdapt instead focuses on proving redundancy in a generic and resource efficient manner.

This document has evaluated the feasibility of SafeAdapt's adaptive technology and which benefits it provides in comparison to state-of-the-art approaches. Foremost, multiple demonstrators were utilised to successfully demonstrate the general applicability of the SafeAdapt results in the automotive industry. Moreover, the evaluation determined that through applying SafeAdapt's adaptive concepts, it is possible to save 25% in hardware cost, 15.6% weight, 11.8% volume, and improve energy efficiency of the E/E-architecture by 21.6% when compared with current methods of designing fail-operational systems. When also applying adaptivity for the energy management, another 25% could be saved with respect to the entire energy consumption of the vehicle. In addition, 30% of cables can be saved when applying an advanced cable routing technique. Next to these physical dimensions, additional benefits can be attained from SafeAdapt's development processes. Here, it is possible to reduce the complexity of the system design in the categories of infrastructure layout, hardware diversity, software development, and safety certification through applying advanced modelling techniques, automated system synthesis methods, and efficient safety certification strategies. This rigorous and strongly automated process further improves the safety of the system by eliminating potential sources of human error during the design.

Overall, SafeAdapt's proposed methods provide the foundation for developing adaptive and fail-operational systems in the next generation of e-vehicles and highly automated cars, as the methods allow for an efficient and safe system design. In addition, all targeted goals indicated in the Description of Work could be met. As the proposed methods substantially outperform state-of-the-art approaches through the use of adaptivity and generic fault handling, an adoption of SafeAdapt's methods by the automotive industry is highly probable.

Bibliography

[1] SafeAdapt, "D5.1: Evaluation Methodology for the SafeAdapt Results".

[2] SafeAdapt, "D2.1: Definition of Use Cases and Scenarios for Safe Adaptation," 2014.

[3] K. Höfig, M. Zeller, and R. Heilmann, "ALFRED: A Methodology to Enable Component Fault Trees for Layered Architectures," *41st Euromicro Conference on Software Engineering and Advanced Applications*, pp. 167-176, 2015.

[4] SafeAdapt, "D2.2: Requirements for the Run-time Control for Safe Adaptation and Supporting Hardware Platforms," 2014.

[5] SafeAdapt, "D3.3: Specification of ISO 26262 safety goals for self-adaptation scenarios," 2015.

[6] A. Ruiz, A. Melzi, and T. Kelly, "Systematic Application of ISO 26262 on a SEooC: Support by Applying a Systematic Reuse Approach," *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*, pp. 393-396.

[7] SafeAdapt, "D3.2: Specification of Runtime Control for Enforcing Safe Adaptation," 2015.

[8] J. Frtunikj, V. Rupanov, M. Armbruster, A. Knoll, "Adaptive Error and Sensor Management for Autonomous Vehicles: Model-based Approach and Run-time System," in *Model-Based Safety and Assessment*, Munich, October 27-29, 2014.

[9] F. M. Dörenberg. (1997, Feb.) Integrated and Modular Systems for Commercial Aviation..

[10] C. R. Spitzer, *Digital Avionics Handbook (Vols. Avionics: Elements, Software and Functions).* CRC Press, 2007.

[11] LinkedIn. [Online]. https://www.linkedin.com/groups/Cost-versus-ASIL-2308567.S.92692199

[12] J. Morris and P. Koopman, "Representing design tradeoffs in safety-critical systems," *SIGSOFT Softw. Eng. Notes*, pp. 1-5, 2005.

[13] pricesystems. (2015, May) http://www.pricesystems.com. [Online]. http://www.pricesystems.com/Portals/1/Blog/02-05-15-A/DO-178bc%20and%20DO-254%20TP%20Modeling%20Guidance%20-%20DRAFT.pdf

[14] F. Asplund, M. Biehl, J. El-khoury, D. Frede, and M. Törngren, "Tool integration, from tool to tool chain with ISO 26262,," in *SAE 2012 Conference: World Congress and Exhibition*, 2012.

[15] D. Makartetskiy, D. Pozza, and R. Sisto, "An Overview of Software-based Support Tools for ISO 26262," in *Innovative Information Technologies: Theory and Practice*, Dresden, 2010.

[16] O. Slotosch, M. Wildmoser, J. Philipps, R. Jeschull, and R. Zalman, "ISO 26262 - Tool Chain Analysis Reduces Tool Qualification Costs," *Automotive*, 2012.

[17] E. Armengaud, et al., "Integrated tool-chain for improving traceability during the development of automotive systems," in *Proceedings of the 2012 Embedded Real Time Software and Systems Conference*, 2012.

[18] K. Höfig, M. Zeller, and L. Grunske, "metaFMEA - A Framework for Reusable FMEAs,," in *Proceedings of the 4th International Symposium on Model Based Safety Assessment (IMBSA).*, 2014.

[19] P. Cuenot, C. Ainhauser, N. Adler, S. Otten, and F. Meurville, "Applying Model Based

Techniques for Early Safety Evaluation of an Automotive Architecture in Compliance with the ISO26262 Standards," in *Proceedings from ERTS 2014*, Toulouse, 2014.

[20] P. Schleiss, M. Zeller, G. Weiss, and D. Eilers, "Safe Adaptive Software for Fully Electric Vehicles," in *Conference on Future Automotive Technology (CoFAT14)*, 2014.

[21] Rockwell Collins. (2009, Jan.) Eurocontrol. [Online]. https://www.eurocontrol.int/sites/default/files/content/documents/communications/29012009-certification-cost-estimation-for-fci-platform.pdf.pdf

[22] T. Kelly and S. Bates, "The Costs, Benefits, and Risks Associated With Pattern-Based and Modular Safety Case Development," in *Proceedings of the UK MoD Equipment Safety Assurance Symposium 2005*, 2005.

[23] SafeAdapt Project, *D4.2 Specification of the design process for safe adaptive embedded systems and tool support for V&V adaptive system behaviour*. 2014.

List of Abbreviations

Abbreviation	Definition
ACC	Adaptive Cruise Control
AEB	Automatic Emergency Brake
APP	Application
ASIL	Automotive Safety Integrity Level
BbW	Brake-by-Wire
BMS	Battery Management System
CCC	Central Computing Core
CDD	Complex Device Driver
CFT	Component Fault Tree
Cn	Core Node
DCC	Duplex Control Computer
DiL	Driver-in-the-Loop
EBC	Emergency Brake Control
E/E	Electric / Electronic
FEV	Fully Electric Vehicles
GW	Gateway
HiL	Hardware-in-the-Loop
HW	Hardware
I/O	Input / Output
MiL	Model-in-the-Loop
RACE	Robust and reliable Automotive Computing Environment
RTE	Runtime Environment
SAPC	Safe Adaptation Platform Core
SbW	Steer-by-Wire
SOC	State Of Charge
SW	Software
TMDP	Trusted Multi Domain Platform
TTE	Time-Triggered Ethernet