PART I

ADAS Development Platform

2

The DESERVE Platform: A Flexible Development Framework to Seemlessly Support the ADAS Development Levels

Frank Badstübner[1], Ralf Ködel[1], Wilhelm Maurer[1], Martin Kunert[2], André Rolfsmeier[3], Joshué Pérez[4], Florian Giesemann[5], Guillermo Payá-Vayá[5], Holger Blume[5] and Gideon Reade[6]

[1]Infineon Technologies AG, Germany
[2]Robert Bosch GmbH, Germany
[3]dSpace GmbH, Germany
[4]INRIA, France
[5]IMS/Hannover University, Germany
[6]ASL, U.K.

2.1 Introduction to the DESERVE Platform Concept

As outlined by Figure 2.1, the DESERVE platform is the key enabler for speeding up the development of next generation ADAS systems. The DESERVE platform represents an open platform to be used by anyone. This chapter therefore covers the entire spectrum of aspects to be considered for the use of this generic DESERVE platform.

Please kindly note that the extensive work on the DESERVE platform cannot be completely described here. Thus, reference to a manifold of DESERVE deliverables are made. As most of these deliverables are not publicly available, essential findings in these deliverable reports were included here to provide a complete view on the DESERVE platform.

The DESERVE platform relies on model-based design and virtual testing tools. Its openness is based on the compliance with AUTOSAR standards. All AUTOSAR members have access to these standardized interfaces.

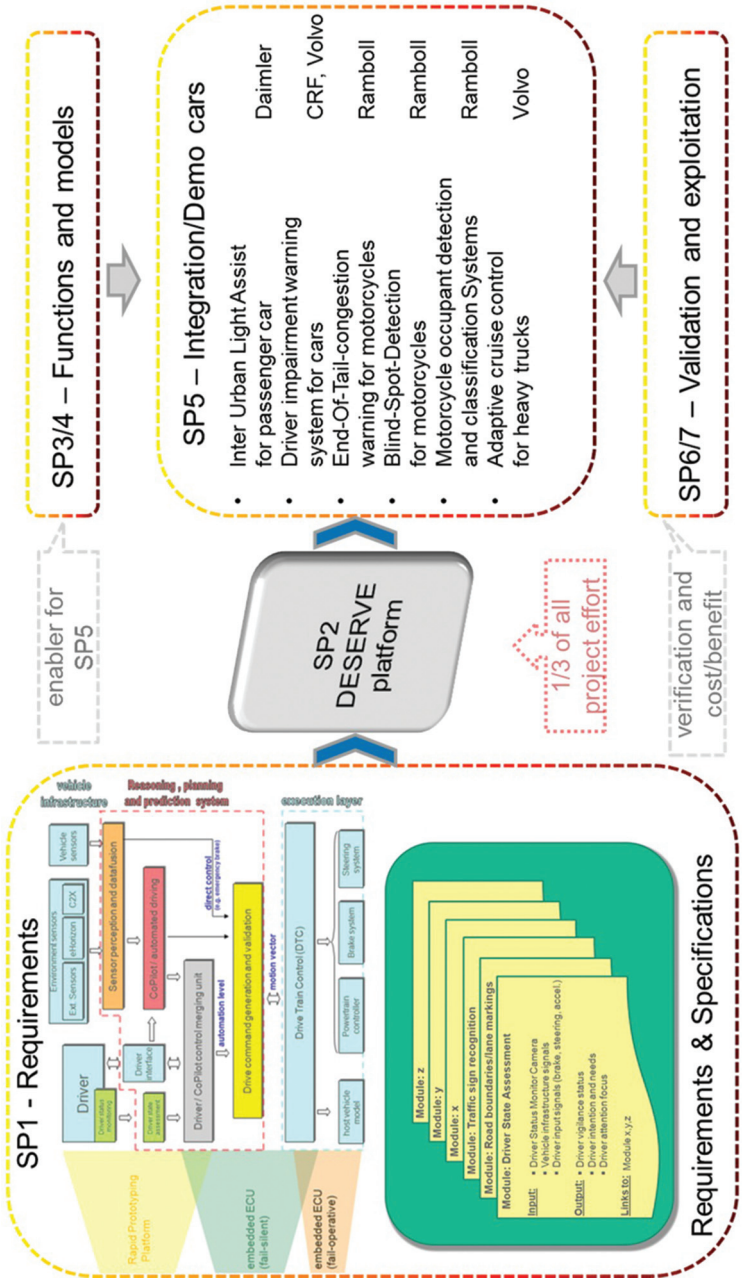

Figure 2.1 The DESERVE Platform – the enabler for next generation ADAS systems.

The DESERVE platform is not related to any specific hardware or software. In contrast, it is generic and represents a new methodology and concept to develop future ADAS systems more efficient and more flexible with maximum reuse of modules and components due to well-defined processes and standardizations on architecture and encapsulated module levels.

Requirements engineering is applied for next generation ADAS systems. By means of model-based design (e.g. Matlab/Simulink/ADTF/RTMaps) fast implementation in ADAS rapid prototyping framework is achieved (development level 2). Rapid prototyping results are evaluated by Hardware-in-the-Loop (HIL), Model-in-the-Loop (MIL) or Processor-in-the-Loop (PIL) test bench. In parallel, by making use of model based design space exploration, specifications and requirements for System-on-Chip (SoC) can be derived at a very early development phase, which supports cost prediction on basis of silicon area, throughput etc. Both, validation by virtual testing and cost prediction indicate important improvement potentials that need to be implemented in the next cycle of the iterative development process.

The situation before DESERVE can be characterized by the absence of model-based access to perception and fusion algorithms, missing AUTOSAR compatibility, there is no library with available algorithms (for composing and evaluating new algorithms). Rather, testing the application on real vehicles in real traffic scenarios is the approach followed, together with some recording feature to allow the capturing of the critical situations, where the solution fails for example, in order to reproduce them in some way later in laboratory.

The objectives of the DESERVE platform are driven by the market needs, which are enabling a further growth of embedded systems and more specifically advanced driver assistance systems (ADAS), mastering the complexity (both in system architecture and processing power) of ADAS, reducing costs of components and development time of ADAS as well as the seamless integration of the growing amount of functions within ADAS and the corresponding vehicle.

DESERVE strives to meet these markets needs by aiming at a novel design and more efficient development process that is enabled by a platform. A platform that provides a flexible development framework, reaching from early PC-based pre-developments down to close-to-production hardware implementations on final target systems on chip, to seamlessly support the ADAS development levels; that constructs a tool chain to allow for modelling and evaluation via virtual testing of new sensors, algorithms, applications and actuators during the whole design and development process; a platform; that forms a common in-vehicle platform for future ADAS functions based

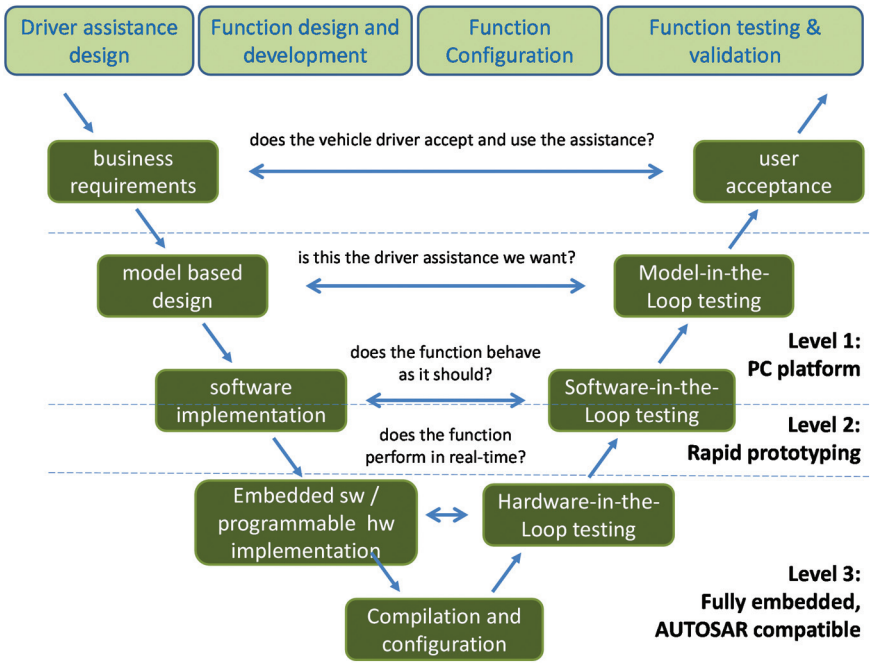

Figure 2.2 DESERVE platform enabled design and development process.

on a modular approach and an architecture and interface specifications that are compatible with AUTOSAR (access and easy-to-use also for non-project-partners); a platform that enables the integration of safety mechanisms for pre-certification (generic safety requirements e.g. for testing on public roads) and full requirements for ASIL D according to ISO 26262 (to prepare certification of later target platform) and security mechanisms for pre-certification of connected ADAS according to ISO 27001.

The novel design and efficient development process is based on the well-known V-model and fully DESERVE platform supported during all phases in the process. This is illustrated in Figure 2.2.

2.2 The DESERVE Platform – A Flexible Development Framework to Seamlessly Support the ADAS Development Levels

This section introduces into the development methods and guidelines associated with the DESERVE platform and outlines the benefits in terms of development cost and time savings from the OEM perspective. Basically, the

platform concept is based on three pillars which reflect the different development levels and the transition of ADAS algorithms from the prototyping to production phase in the automotive industry (see Figure 2.3).

The DESERVE platform is a generic platform that supports all development levels illustrated in Figure 2.3 as seamless as possible – from feasibility study to product development.

Level 1: PC platform

In the research and pre-development phase users typically require highly flexible tools with an intuitive user interface and the implementation of ADAS algorithms may not satisfy hard real-time requirements. Here, PC-based tools such as ADTF and RTMaps for data fusion often constitute the basis for ADAS development.

Such tools provide a high user comfort and allow developers to implement and verify algorithms directly on a standard MS Windows or Linux PC. Different kinds of sensors/actuators and vehicle bus interfaces are available so that the algorithms can directly be tested in a real environment. However, real-time calculation is not guaranteed, especially with complex perception, fusion and tracking algorithms. In addition, there is no direct support of Matlab/Simulink, AUTOSAR and the model-based design approach for application functions. Finally, PC platforms as described above are typically not tailored for stand-alone, in-vehicle use cases.

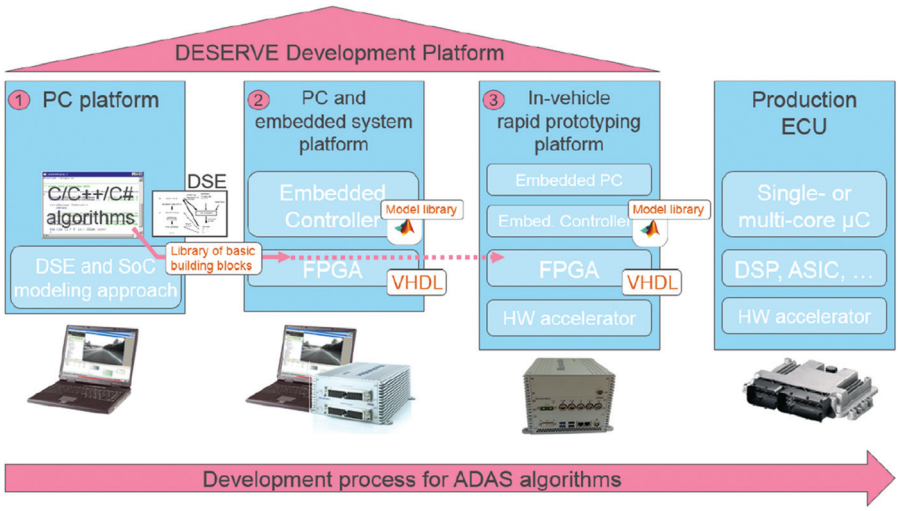

Figure 2.3 ADAS development process.

To avoid a time-consuming redesign of perception, fusion or tracking algorithms when implementing them on the final ECU hardware (production ECU), engineers are looking for ways to evaluate different target hardware architectures according to given cost criteria already in early development stages. This request is met by the design space exploration (DSE) methodology and the SoC modelling approach.

Level 2: Rapid prototyping platform including software superstructure (e.g. embedded PC/embedded controller with realtime operating system and FPGA)

In the second development stage engineers go one step closer to a real-time implementation. Complex and computationally intensive algorithms are shifted to a powerful FPGA to improve the realtime capability. In parallel to this, the FPGA platform allows different target hardware architectures to be evaluated in combination with the selected algorithms. To ensure a rapid implementation of the above mentioned perception, fusion, and tracking algorithms in the FPGA, basic building blocks in terms of a library are provided by the DSE framework. By means of this block-based modeling approach the time and effort for implementing the associated algorithms can significantly be reduced.

Using an embedded system platform in this stage featuring both an FPGA and an embedded controller also allows ADAS application algorithms to be designed by means of models so that the associated development time can further be reduced. Compared to the purely PC based framework real-time performance is almost guaranteed, though the user comfort with programming the FPGA may be restricted.

Level 3: Fully embedded, AUTOSAR compatible architecture (e.g. multicore controller with FPGA) for the evaluation of algorithms in realtime and implementation of safety requirements according to ISO 26262 (e.g. pre-certification for testing on public roads)

The goal of this stage is to go one step further to the final target hardware and to provide a stand-alone, in-vehicle rapid prototyping platform which, for example, can even be used during test drives. This stage reflects the users' need to evaluate and experience the driver assistance system directly in the vehicle itself.

The standard PC is replaced by an embedded PC that is qualified for in-vehicle use in terms of shock, vibration and temperature, similar to the other parts of the system. This platform also allows the integration of hardware accelerators so that even highly computational intensive algorithms may be tested in the vehicle. It is also possible to interface target microcontrollers of

production ECUs and to run certain algorithms there. The complete platform behaves like a prototype ECU which can be operated by test drivers which are not specifically instructed. For example, the platform can be started and shut down via the vehicle's ignition key.

The development platforms of all stages can be used together with the model-based design space exploration approach for system on chip and libraries of basic building blocks for the FPGA. By means of this the gap is closed when transferring perception, fusion and tracking algorithms from prototyping to production, similar to the model-based design approach with application functions using Simulink. Being able to use already tested and validated building blocks and software modules greatly facilitates and expedites the development process.

To support the model-based development of algorithms at all processing layers (perception, decision making, warning and control strategies) and to execute these algorithms in the vehicle, the DESERVE platform level 3 needs to be fully compatible to the AUTOSAR standard (note: as of today, no certified AUTOSAR 4.0 real-time operating system including memory protection is available; its development is not subject of DESERVE).

In addition, at this development level, safety mechanisms need to be developed: According to ISO 26262 the DAS system needs to be classified concerning the Automotive Safety Integrity Level (ASIL). Many DAS systems require the highest classification ASIL D. Suitable measures are required to fulfil the related strong requirements. As the certification process is very much related to the hardware, just pre-certification (e.g. for testing of the new DAS on public roads) is possible at this development level.

As a result, OEMs are able to define early and precise enough the distinct requirements for the final ECU hard- and software (e.g. required interfaces – which I/O and bus system; computational power; memory requirements), including the safety mechanisms (e.g. memory protection, lockstep operation).

Level 4: Target production platform (e.g. multicore controller ECU with integrated custom ASIC/FPGA/hardware accelerator)

On basis of the production hardware, the final certification of the ADAS takes place. Within the DESERVE project, the generic DESERVE platform concept was validated. Starting with purely PC-based development, algorithms can be outsourced step by step to an FPGA or embedded controller prototyping system. In addition to the hardware concept, a design space exploration and an analytical modelling approach for system on chip is proposed. This software framework allows different target hardware architectures for the implementation of perception algorithms to be evaluated according to given

cost criteria in early development phases. The software framework is coupled to the FPGA of the DESERVE platform. The associated workflow will be supported by a library of basic building blocks for the FPGA by means of which perception algorithms can be composed and implemented quickly.

To validate the platform concept, three different realization instances of the generic DESERVE platform are considered in the project:

- Level 1: Purely PC based solution
- Level 2: Mixed PC/embedded control based on dSpace Micro Autobox with FPGA framework (this platform will be extensively used for the ADAS vehicle demonstrators)
- Level 3: Fully embedded platform based on multicore controller plus FPGA. This instance of the DESERVE platform provides realtime operating system and basis software fully compatible to the AUTOSAR standard. Thus it is open and easy to use for all AUTOSAR members. It will also feature safety concepts required for ASIL D and consider new radar/camera interfaces.

2.3 DESERVE Platform Requirements

The next step in the definition process for the DESERVE platform concerned the translation of the previously defined platform needs into generic requirements for the DESERVE platform based on common software architecture and suitable for the development and simulation of the 33 DAS functions investigated in the beginning.

The generic requirements for the DESERVE platform were defined utilizing the following approach (see deliverables D1.2.1 [1]).

The DESERVE development platform has been defined taking into account that general requirements such as AUTOSAR compatibility [6], SPICE compliance and functional safety (ISO 26262) [7, 8] are mandatory for industrial use. These requirements apply for the "industrialized platform". The generic DESERVE platform addresses a functional software architecture based on Perception, Application and IWI platforms.

2.3.1 DESERVE Platform Framework

The DESERVE platform has been defined taking into account general requirements such as AUTOSAR compatibility, SPICE compliance and functional safety (ISO 26262), which are mandatory for the later industrial use. The AUTOSAR standard comprises a set of specifications describing software

architecture components and defining their interfaces. DESERVE aims at using AUTOSAR to integrate applications from different suppliers inside a single processing unit.

DESERVE addressed also to be compliant with the SPICE standard, which represents a set of technical standards documents for the computer software development process and related business management functions. The ISO 26262 standard was considered in the implementation of DESERVE platform in order to improve the safety in the development of methods and tools. The ISO 26262 standard defines the "Functional Safety Assessment" at the completion of the item development with the scope to assess the functional safety that is achieved by the element under safety analysis.

The baseline for DESERVE is represented by the results of past and ongoing research projects [9, 10], and in particular of interactIVe addressing the development of a common perception framework for multiple safety applications with unified output interface from the perception layer to the application layer [11].

Figure 2.4 presents the DESERVE platform framework. In this generic architecture the perception platform processes the data received from the sensors that are available on the ego vehicle and sends them to the application platform in order to develop control functions and to decide the actuation

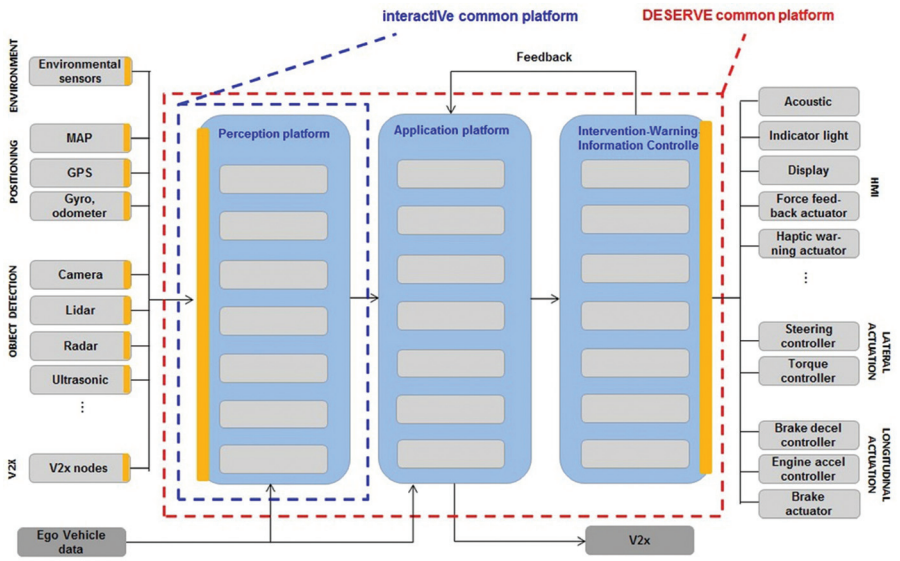

Figure 2.4 DESERVE platform framework.

strategies. Finally, the output is sent to the IWI platform informing the driver in case of warning conditions and activating the systems related to the longitudinal and/or lateral dynamics.

2.3.2 Generic DESERVE Platform Requirements (Relevant to all Development Levels)

Different clusters of requirements were defined following the structure of the DESERVE platform framework. Please note that each of the following requirements was divided in sub-requirements, which are described in detail in DESERVE deliverable D1.2.1.

General software requirements

General software requirements: Among others, these cover the previously mentioned software requirements for modularity, reusability, AUTOSAR, SPICE process assessment (ISO/IEC 15504), functional safety (ISO 26262), platform independence (the application software needs to be independent from the processing hardware), standardized interfaces (i.e. the software needs to have interfaces to sensors and actuators that are standardized and published), operating system independence (cross platform libraries are recommended), programming language, communication technologies independence, automatic start-up/shut-down, configuration of sensors position, software versioning and licenses.

General hardware platform requirements

These cover the aspects power supply, list of supported sensors, processing unit, unit size and number of included components etc.

Perception module requirements

These requirements include 3D reconstruction of the scene in front of the vehicle, ADASIS horizon, assignment of objects to lanes, detection of the free space, driver monitoring, enhanced vehicle positioning, environment, front near range perception, frontal object perception, lane course, lane recognition, moving object classification, occupant monitoring, parking lot detector, recognition of unavoidable crash situations, relative positioning of the ego vehicle to the road, road data fusion, road edge detection, scene labelling, self-calibration, side/rear object perception, traffic sign detector, vehicle filter/state, vehicle light detector, vehicle trajectory calculation, vulnerable road users detection and classification.

The functional architecture of the perception layer is illustrated in Figure 2.5. Depending on the ADAS system to be realized, some of the

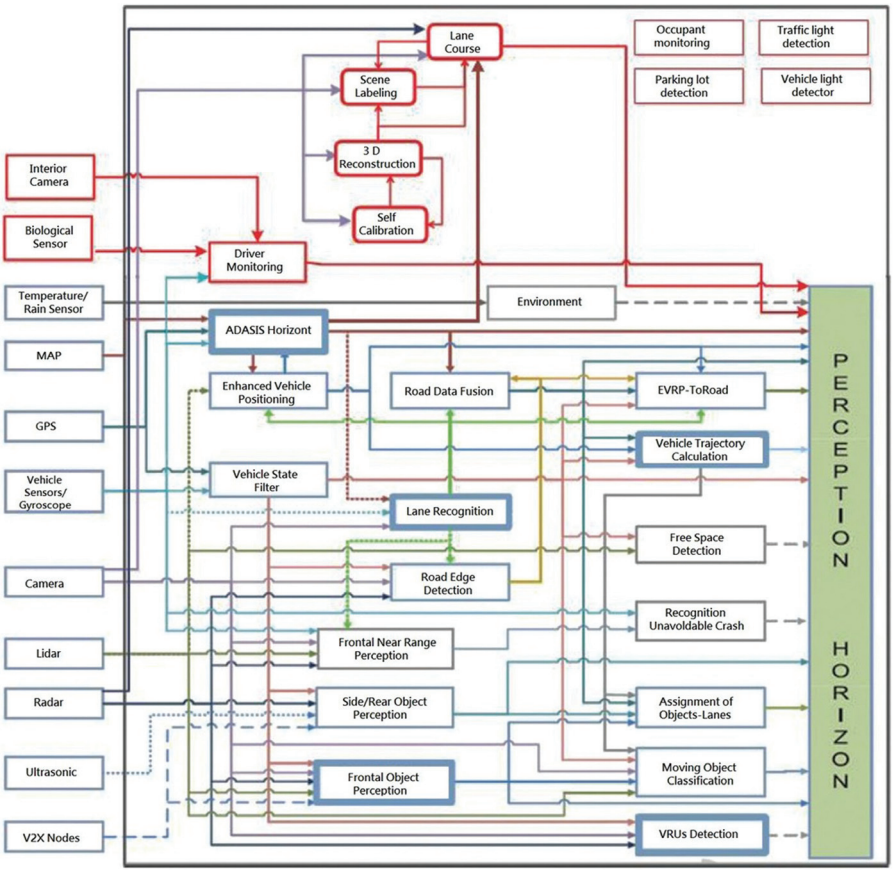

Figure 2.5 Perception platform functional architecture.

components in the generic perception platform architecture may be omitted (without losing generality). The modules developed in the project to build the demonstrators are highlighted by thicker boxes.

The number and variety of the different perception sources is manifold and requires special care and precaution to transport the available information in the subsequent data processing modules. Two main aspects have to be taken into consideration when connecting perception sources to the DESERVE platform: The information content may differ from sensor to sensor even when the same technique (e.g. radar, video camera or ultrasonic sensor) is used. Based on the physical concept used the individual sensors may have an intrinsic lack of information that can never be provided, independent of the

effort spent to improve the sensor performance (e.g. radar sensors can never "visually" read the road signs content while video sensors can never provide direct speed measurements).

By using the general interface descriptor approach the data input structure for the perception layer processing module becomes independent from the real sensors connected to the DESERVE platform. This kind of concept is used in PC architecture since several years under the term hardware abstraction layer that completely decouples data information from the physical hardware in use.

The flexibility and scalability of the overall system is much better and reusability of SW components that are already developed is higher. Improvements and changes within the subgroups (i.e. environmental sensors or perception input processing module) can be conducted on a standalone basis without modifying or adapting the whole data processing chain at all. General adoption of the whole data processing chain is thus only needed in the case that the interference descriptors between the modules have to be updated or modified due to recently emerging needs.

As the diversity of the already existing environmental sensors is already huge and many products are already in series production, the change of the sensor output signals is often not possible at all. To connect already existing sensing devices or sensors with an IP-protected signal output to the open DESERVE platform, a work-around with converter or breakout boxes can be applied. Using such interface converter/breakout boxes almost any kind of sensor system can be attached to the standardized and abstracted input channels of the generic DESERVE platform.

Application module requirements

The application module needs to consider the following requirements: ACC control, activation control, advance warning generator, calculation of required evasion trajectory, decision unit, driver intention detection, driving strategy, intervention path determination, IWI manager, reference maneuver, situation analysis, target selection, threat assessment, trajectory control, trajectory planning, vehicle model and vehicle motion control.

The functional scheme of the application platform modules is depicted in Figure 2.6. The modules are divided in clusters having the same scope. Some of them have mainly the objective to select the driver intention and the most dangerous target. Other modules execute control operations and make an evaluation about the current situation of warning and eventually decide specific actions. Then the type of information to provide to the driver and the

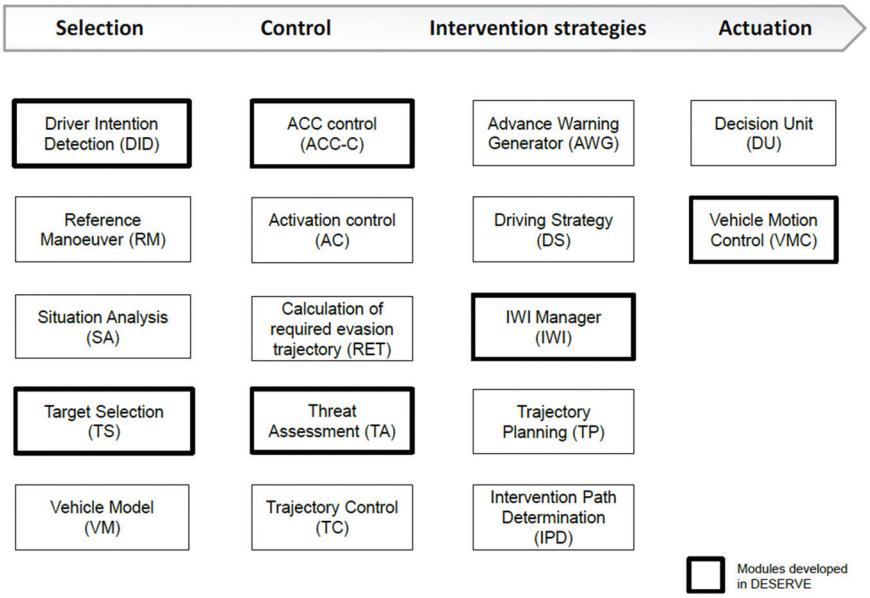

Figure 2.6 Application platform functional architecture.

intervention strategy are decided. Finally, the kind of actuation to adopt is provided to the IWI Platform modules.

IWI module requirements

The IWI module is dedicated to suit requirements regarding the HMI (acoustic, displays, telltales, haptic steering wheel, haptic accelerator pedal, haptic safety belt), actuation of external lights, lateral actuation (steering angle and steering torque controller) and longitudinal actuation (engine acceleration controller). The functional architecture of the IWI platform is depicted in Figure 2.7.

Different levels in the development process of ADAS require different instances (i.e. realizations) of the generic DESERVE platform – from PC based (development level 1) to production hardware (development level 4). With increasing development levels, additional requirements need to be addressed. This principle shall be explained in the next two subsections.

2.3.3 Rapid Prototyping Framework Requirements (Development Level 2)

This section shortly outlines the main requirements for the DESERVE rapid prototyping platform. The main intention here is to specify a flexible and

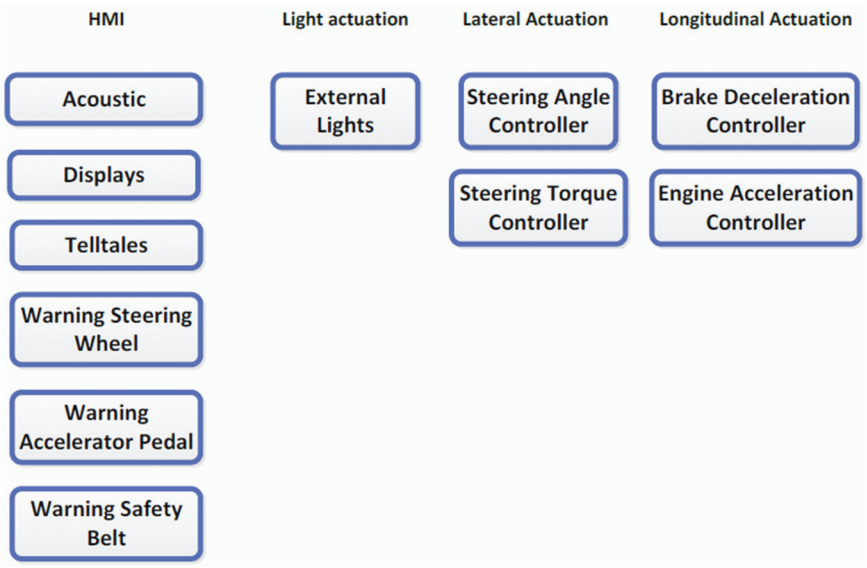

Figure 2.7 DESERVE IWI platform.

modular rapid prototyping environment allowing ADAS related perception, application and intervention algorithms to be developed in short iteration cycles and to be prototyped directly in the vehicle. In order to do so, there is a need to connect different kinds of sensors to the development framework, to pre-process and fuse the sensor data, to calculate the actual ADAS applications and to finally drive the respective actuators.

The structure for the generic requirements in the previous section, the rapid prototyping system requirements are structured in hardware, software and FPGA code requirements. In addition, a distinction is made between perception (i.e. sensor data processing) and application algorithms.

2.3.4 Additional Requirements for Embedded Multicore Platform with FPGA (Development Level 3)

While the main focus of development level 2 is on evaluation of algorithms in real-time on public roads, thus on ADAS functionalities and use in the DESERVE DAS function demonstrators, levels 3 (and 4) go significantly ahead in terms of fulfilling "critical" requirements like AUTOSAR compatibility, SPICE compliance and functional safety (ISO 26262) which are mandatory for industrial use of the platform. Due to limited resources and

limited project duration, these requirements cannot be fully implemented in DESERVE. Nevertheless all the work done for the "non-industrialized" DESERVE platform can be (partly) reused or carried over to the industrialized version of the DESERVE platform (level 4).

2.4 DESERVE Platform Specification and Architecture

The generic platform requirements were translated into specifications, which represent the starting point for the development of modules for the DESERVE platform. The specifications were included into an Excel file which is accessible to all project partners via the project server. By means of an iterative process, both specifications and software design were refined and improved. A summary of the specification approach and of the specifications derived from the DESERVE platform requirements is provided in deliverable D1.3.1 [2].

2.4.1 DESERVE Platform Architecture

The architecture of the DESERVE development platform shall follow both the principle of standard DAS development cycles and the mappings of application building blocks to final, often heterogeneous hardware implementations. To date there is no tool or framework available that covers both requirements at the same time on the same platform.

In the early concept and implementation phase the basic development, specification and validation (e.g. with MIL, SIL or HIL) is often done with another development framework (both for SW and HW) than the one applied for the final target platform. Little is known or taken into account from the final embedded system characteristics when first application algorithms are programmed and very often the SW modules written in this first development environment have to be reprogrammed from the scratch when porting it to the embedded system on chip. If the software, mostly written in a high-level programming language, finally fits the target system one has selected for series production, is a game of pure chance and not rarely during the series product development cycle a larger target system or some "add-ons" have to be chosen. With the new design space exploration methodology the certainty to select the suitable embedded target system at first time is significantly increased.

The DESERVE development platform architecture has to comply with the following basic needs:

- Enough flexibility to encompass different development environments in a common, seamless framework for both the high-level algorithm

development and the easy porting of these SW modules to the embedded target platform.
- Real time recording and playback capabilities for both the high-level and embedded system implementations.
- A communication architecture that is capable to shift SW portions from the high-level development side to the embedded target system as required (i.e. bypassing with HW accelerators).
- A seamless interoperability and replacement between the high-level (i.e. PC-based) and embedded target systems both for development and validation purposes.

The basic idea and intention of this hardware architecture is to standardize the interfaces between the three different development concept levels as good as possible.

Inputs from proprietary ADAS sensor systems and information sources are analyzed via a generic interface no. 1 to the PC based development environment. Here the ADTF tool with its filter programming concept is used to develop or improve SW modules on a high-level programming language. The partitioning and optimization of parts of the SW modules is consecutively done by shifting such portions over the generic interface no. 2 to the embedded controller framework that is already much nearer to the final commercial product. Via this bidirectional interface bypassing techniques like PIL (embedded Processor In the Loop) can be realized. In a final step, dedicated HW accelerators can be linked in via the generic interface no. 3 by applying the same bypassing concept. Especially computationally intensive tasks can so be "outsourced", so that even the PC-based platform is capable to keep the stringent real-time constraints.

Depending on the performance of the PC either all or only specific parts of the SW modules can be executed there. During the development process more and more SW parts are transferred to the HW-Accelerator level, which, in the final development stage, results in the next generation embedded ADAS target system. At this last development step, the level 1 (PC) and level 2 (embedded controller) platform will only serve as a shell to keep up the overall development framework.

Reuse of already existing components from former ADAS generations may be used in the early development phase as HW accelerators for computational intensive calculations. Mainly standard algorithms that are fixed and receive no further modifications are preferred candidates for such specific HW accelerators.

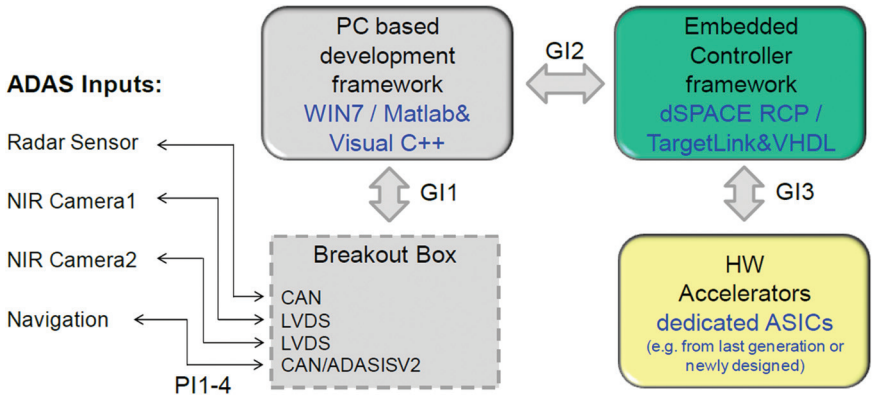

Figure 2.8 DESERVE platform (e.g. for development Level 2 – rapid prototyping system based on mixed PC and embedded controller framework).

This section summarizes the DESERVE platform architecture aspects. It considers hard- and software architecture aspects. The platform architecture is described in detail in deliverable D25.2 [4].

2.4.1.1 Hardware architecture

DESERVE has to be flexible enough to be implemented in a distributed and scalable architecture (several modules, each of them able to sense and/or process and/or actuate) or a concentrated one (sensors and actuators all linked with a single unit of processing and control). Task 2.5.1 identifies which conditions have to be satisfied by the individual subsystem architectures in order to be compliant with the DESERVE generic hardware platform.

For maximum reusability the DESERVE concept and hardware architecture was designed in such a way that subsystems of different generations (or respectively the kernels of it) can be used in parallel, thereby enabling the rapid and effective creation of next-generation innovative ADAS systems by using well tested and certified kernel functions of the “old” system which partly could be already implemented as SoC (System on Chip). The DESERVE development platform can be seen as a flexible rapid-prototyping environment that enables fast and efficient development of next generation ADAS functions in a continuous iteration cycle between the current and next-generation embedded subsystem components.

Furthermore, the DESERVE concept is flexible enough for different DESERVE partners to make different implementations. These would be of forms that might in future be interoperable, although DESERVE will not

attempt to define detailed standards which would be necessary for actual interoperability.

The main DESERVE idea concerns the use of one common platform system (Figure 2.9) for all ADAS functional modules, instead of the current approach to have one platform for each individual ADAS system. Basically, three main hardware architecture challenges arise from this idea:

- Automotive quality: The platform needs to provide high reliability over the complete automotive temperature range, power supply and environmental conditions. As ADAS systems address safety aspects, the platform should implement as far as possible the ISO 26262 requirements, i.e. at least the hardware components that are near to the final product unit shall support the required ASIL level.
- Possibility to extend hardware capabilities: The platform needs to be designed up-front to support the possibility to include additional hardware into the system. Standard sensor interfaces are needed, for instance, but also standardized interfacing to external FPGA/DSP for performance enhancement is required. For scalability purposes, such external devices need to be cascadable. Similar considerations hold for the memory interface capability.
- A special case of hardware extension capabilities is the reuse of serial parts from earlier generations to speed up the development process or to increase the sensor perception by placing more sensors on the car.
- Finally, a seamless environment tool chain is needed. One key requirement lies in the reuse of the existing tool ecosystem over several platform generations. Further, we should target adaptability of the tools to the broad industry use cases, e.g. next generation video and radar sensors. Additionally, real-time monitoring and debugging of interface and processing for development purposes represent key challenges.

2.4.1.2 Software architecture

As for hardware architecture, the characteristics and constraints that the software architecture has to fulfill to accept an application based on modules developed inside the DESERVE platform (Figure 2.10) were identified. AUTOSAR standards were considered[1].

[1]Note: Being a research project, the development work conducted in DESERVE is discharged from being fully compliant with the AUTOSAR standard. Where possible and easy to implement, inputs from AUTOSAR were considered, of course. A mandatory request for AUTOSAR compliance is, however, not up for discussion.

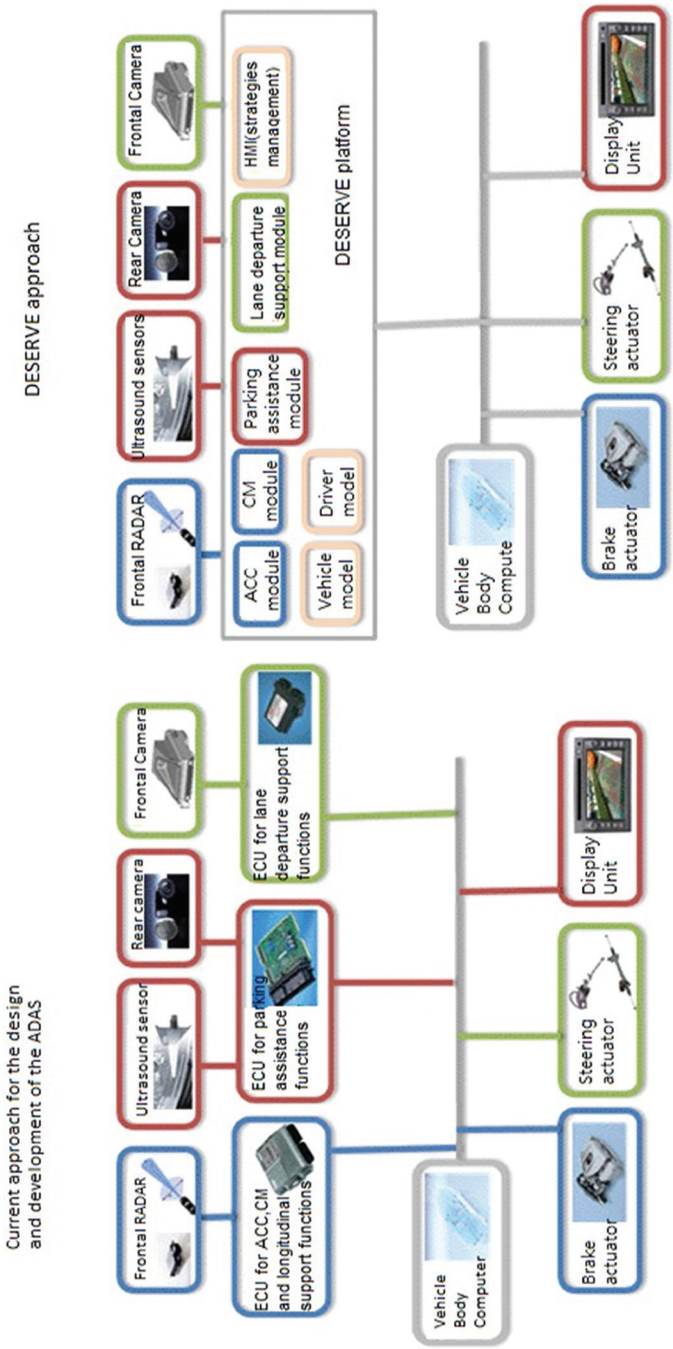

Figure 2.9 DESERVE approach – use of common platform for all ADAS modules.

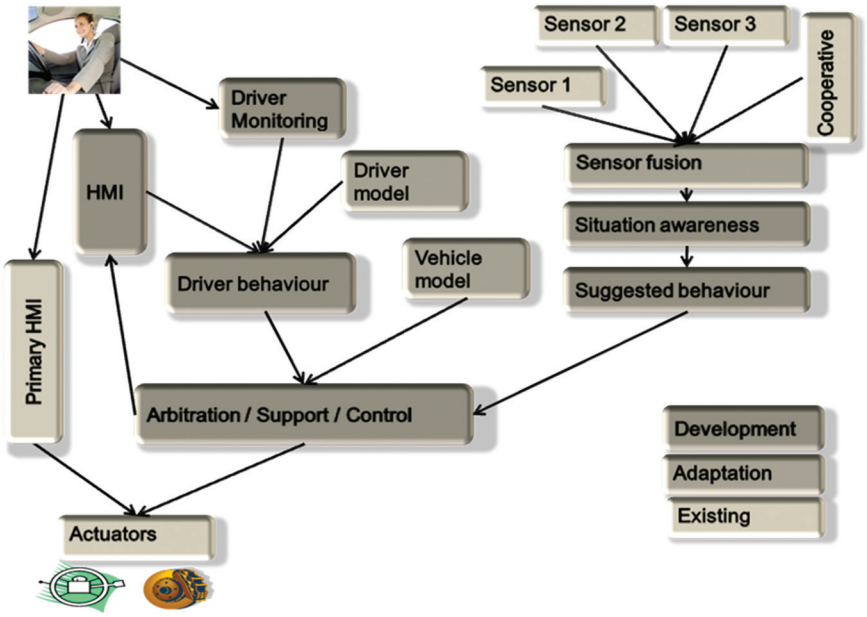

Figure 2.10 DESERVE platform architecture.

The key architecture challenges are: AUTOSAR Standards Architecture for the full platform system including performance accelerators, request for high SW re-usability/testability including re-use of older generation software blocks, fast time to market, highly optimized library for optimal performance, automatic code generation, standard compiler/tool chain and finally, hardware tool software support for realtime debugging, high speed parallel sensor data capture for validation and on-system debugging is required.

Application Software Modules

On the base of AUTOSAR standard, the general software architecture can be represented in three main layers: low level (basic software: this level abstracts from the hardware, provides basic and complex drivers and services for high level, i.e. memory, I/O), middle level (virtual function bus and runtime infrastructure) and high level (application software components).

The AUTOSAR standard introduces two architectural concepts (respects to other embedded software architectures) that facilitate infrastructure independent software development. Namely, these are the Virtual Function Bus (VFB) and the Runtime Infrastructure (RTE) that are closely related to each other.

In order to realize this degree of flexibility against the underlying infrastructure, the AUTOSAR software architecture follows several abstraction principles. In general, any piece of software within an AUTOSAR infrastructure can be seen as an independent component while each AUTOSAR application is a set of inter-connected AUTOSAR components.

Further, the different layers of abstraction allow the application designer to disregard several aspects of the physical system on which the application will later be deployed on, like type of micro controller, type of ECU hardware, physical location of interconnected components, networking technology/buses or instantiation of components/number of instances.

The middle level, VFB (Figure 2.11), provides generic communication services that can be consumed by any existing AUTOSAR software component. Although any of these services are virtual. They will in a later development phase be mapped to actual implemented methods that are specific for the underlying hardware infrastructure. The RTE (runtime environment) provides an actual representation of the virtual concepts of the VFB for one specific ECU.

An AUTOSAR software component in general is the core of any AUTOSAR application. It is built as a hierarchical composition of atomic software components. The AUTOSAR software component can be divided in Application Software Component and AUTOSAR Interface. It is important for DESERVE to preserve (and build up during the prototyping phase of the applications) the AUTOSAR modularity concept. Consequently, DESERVE focuses on the development of modular Application Software Components.

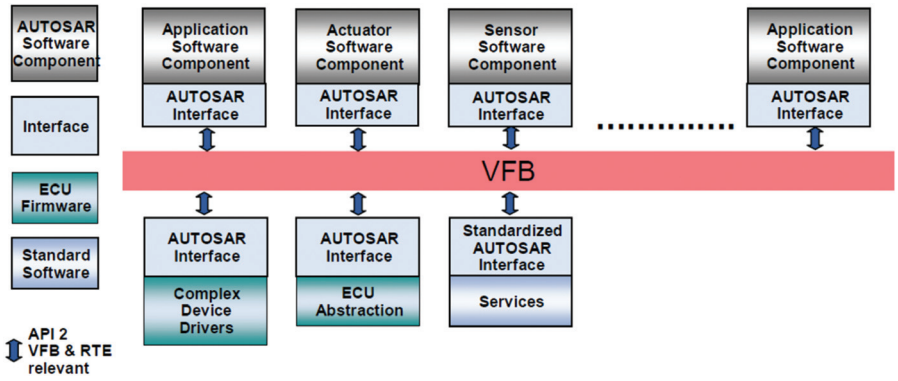

Figure 2.11 Overview on the principles of virtual interaction using the AUTOSAR.

Multi-task option to permit adding and removing of functionalities
The modularity is one the most important directive in the design of a global architecture, their functions and modules for embedded systems. Different multi-tasks (called processes) can be executed by sharing common processing resources in the same CPU. In this line, multi-thread languages as C++ are used by different developers around the world.

The software environments used in the DESERVE platforms (e.g. ADTF and RTMaps) are able to transfer functions already programmed in C and C++. These tools are multi-sensory software, designed for fast and robust implementation in multitask systems. They use functional blocks (called components) for data flowing between different types of modules: video, audio, byte streams, CAN frames, among others.

This multi-threaded architecture allows the use of multiple asynchronous sensors within the same application (see RTMaps and ADTF sections in D1.3.2 [3]). Moreover, they take advantage of multi-processor architecture for more computing power.

Based on the Development Platform Requirements [1], there are three main stages in the control architecture: perception, application and IWI platform. The goal of the DESERVE approach is to add different functions (Multi-task) in the same platform.

2.4.2 DESERVE Platform Interface Definition

The definition of the DESERVE interface architecture is described together with state of the art ADAS interfaces and next generation interfaces in deliverable D2.5.4 [5]. Due to the high relevance of the interface architecture for the DESERVE platform concept, a brief description is included in the next paragraphs.

2.4.2.1 Definition of DESERVE interface architecture

The definitions of the interface architecture plays a central role for the communication and data exchange between the different DESERVE platform modules and sensor components. In the DESERVE deliverable D2.2.1 [12] the abstracted interface descriptors are already defined on a content-based hierarchical level. With standardized information data flow between the numerous platform modules both the development time and the extension in performance and scope of the encapsulated modules can be realized very efficiently and in a well-structured way. The architecture of the interface has

to be defined individually for each of the existing OSI layers, starting from the physical layer up to the application layer.

For modules that only communicate within the same hardware unit the physical data and communication layer are no longer needed. Instead, a message box oriented data transfer link is proposed for usage in the DESERVE project. The data to be transmitted is written in a predefined message box descriptor field and message flags trigger the synchronization and data updates in the concerned modules. The message box principle is sketched in Figure 2.12.

The interfacing concept of the AUTOSAR standard is considered and incorporated in the DESERVE platform where useful and appropriate. The AUTOSAR mode of operation, as depicted in Figure 2.13, fits already quite well with the general DESERVE approach proposed in this document.

In order to achieve a good reusability of embedded software functions, it has proven to be efficient in the industry to separate the "function software" from parameters defining the behavior of the software (= calibration data). This allows generating embedded systems with generic software functionalities by "embedded systems suppliers" (e.g. Continental, Bosch or others). Such systems are bought by OEMs for building their ADAS systems. The OEM can adapt the generic function to the individual behavior significant for his customers "just by calibration". In this process via an application system (market leader is INCA for example), the calibration data can be changed while the embedded system is running – regardless if simulated on a PC or

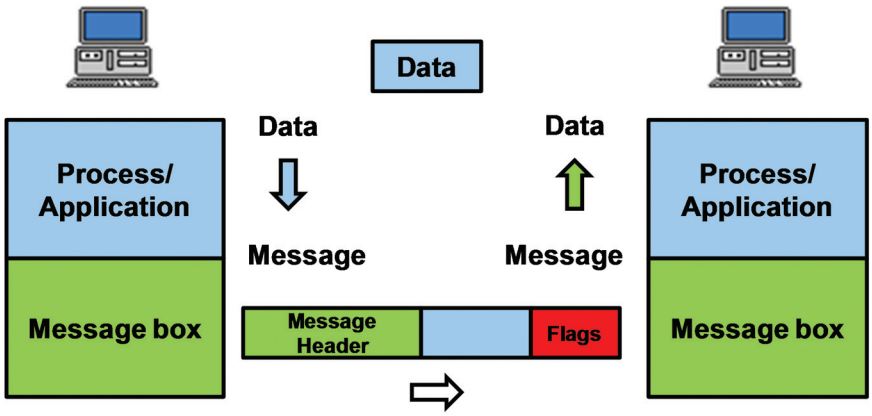

Figure 2.12 Message box principle for intra-unit communication.

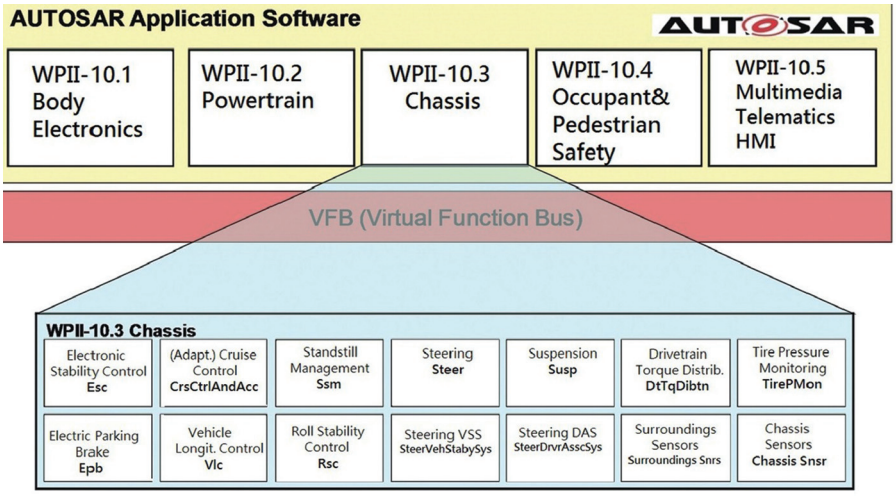

Figure 2.13 AUTOSAR application software concept.

running already on the target hardware. The separation of calibration date and function software is also allowed according to the AUTOSAR concept.

2.4.2.2 Existing ADAS interfaces

All electronic embedded systems used to control vehicle functions (specifically ADAS) need communications networks and protocols to manage all the process information. The modules receive input information from a network of sensors (e.g. for engine speed, lasers, cameras, etc.) and send commands to the control stage (Application platform in DESERVE), and finally to the actuators or warning systems that execute the commands (IWI platform) [1].

Due to the increasing complexity of modern ADAS applications, point-to-point wiring has been replaced by multiple networks and communications protocols. These protocols use different physical media to provide safe connection among components on the vehicle. These include single wires, twisted wire pairs, optical fiber cables, and communication over the vehicle's power lines.

Communication protocols

Some of the most known and used communication protocols and standards used in nowadays vehicles are:

- CAN (controller area network)
- VAN (vehicle area network)

- FlexRay
- LIN (local interconnect network)
- SAE-J1939 and ISO 11783
- MOST (Media-Oriented Systems Transport)
- Keyword Protocol 2000 (KWP2000)

Recent vehicles have installed multiple networks (with different protocols) to communicate among electronic control units (ECU) onboard. The networks are isolated from one another for several reasons, including bandwidth and integration concerns.

Existing interface standards

Current ADAS systems are designed and built to provide a dedicated answer to specific functionalities. Most ADAS are including in the same box the sensor itself and the processing unit. So, the raw data provided by the sensor (camera, radar) are directly loaded inside the ECU unit and processed. Only high level (processed) information is available on the communication buses. Raw data (e.g. pixel information of images) is not available.

The ADAS modules are dedicated products which communicate mainly within the same hardware unit. Nevertheless, to adjust the algorithms in function of the vehicle status, it's necessary to provide the ADAS modules with some vehicle information as: speed, yaw rate, direction indicator status, etc.

To manage the vehicle information acquisition and sending of the outputs, various communication interfaces are available, depending on the product, e.g. CAN or FlexRay communication interfaces.

The communication bandwidth requirements increase more and more with more and more complex applications, the existing network are not specified to cover the increasing demands for bandwidth, and the Ethernet price. Ethernet seems to be an alternative to the existing communication hardware.

2.4.2.3 Definition of next generation interfaces

The definition of next generation high speed sensor interfaces is the key to enable the improvement for next generation driver assistant systems. An optimized interface leads to optimized dataflow and system performance. For each sensor family (Camera/RADAR) there is a dedicated interfacing needed.

Parallel camera interface (CIF)

The Camera Interface (CIF) represents a complete video and still picture input interface transferring data from an image sensor into video memory. Furthermore, several hardware blocks – performing image processing operations on the incoming data – are provided (Figure 2.14).

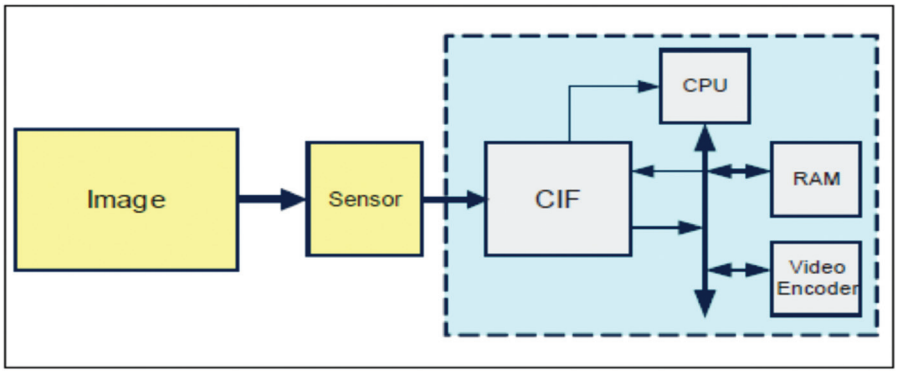

Figure 2.14 Camera Interface (CIF) overview.

Apart from providing the physical interfacing to various types of camera sensor modules, the CIF block implements image processing and encoding functionalities. The integrated image processing unit supports image sensors with integrated YCbCr processing. Additionally, the CIF also supports the transfer of RAW (e.g. Bayer Pattern) images and non-frame synchronized data packets. The CIF block features a 16 bit parallel interface. All output data are transmitted via the memory interface to a BBB (Back Bone Bus) system using the master interface. Programming of the CIF is done by register read/write transactions using a BBB slave interface.

The CIF provides a sensor/camera interface for a wide variety of video applications and it is optimized for high speed data transmission under terms of low power consumption. This module is designed to be used for the following use cases: video capturing/encoding, still image capturing in YCbCr with on-the-fly JPEG encoding and RAW frame data capturing.

The CIF requires fast system memory for image storage in either planar, semi-planar or interleaved YCbCr or RAW planar format or as JPEG compressed data. The iJPEG encoding engine should be able to generate a full JFIF 1.02 compliant JPEG file that can be displayed directly by any image viewer. Important YCbCr formats – which are used for video compression (e.g. MPEG4) for instance – are supported. For on-the-fly encoding macro block line interrupts are generated to trigger video encoding.

Serial RADAR interface (RIF)

Analog-to-digital converter (ADC) sample rates have been increasing steadily for years to accommodate newer bandwidth-hungry applications in communication, instrumentation, and consumer markets. Coupled with the need to

digitize signals early in the signal chain to take advantage of digital signal processing techniques, this has motivated the development of high-speed ADC cores that can digitize at clock rates higher than 100 MHz to 200 MHz with 8 to 12 bit resolution.

In standalone converters, the ADC needs to be able to drive receiving logic and accompanying PCB trace capacitance. Current switching transients due to driving the load can couple back to the ADC analog front end, adversely affecting performance. One approach to minimize this effect has been to provide the output data at one-half the clock rate by multiplexing two output ports, reducing required edge rates, and increasing available settling time between switching instants.

Use of LVDS for ADC high speed data output
A new approach to providing high-speed data outputs while minimizing performance limitations in ADC applications is the use of LVDS (low voltage differential signaling). Infineon is incorporating LVDS output capability in new RF devices ADCs—and will include LVDS input capability in its new micro-controller designs.

Standards
Two standards have been written to define LVDS. One is the ANSI/TIA/EIA-644 which is titled "Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits." The other is IEEE Standard 1596.3 which is titled "IEEE Standard for Low-Voltage Differential Signals (LVDS) for Scalable Coherent Interface" (SCI).

Generic interface to communicate between ADTF project and FPGA based hardware platform
In order to allow an easy and standard communication between an ADTF-Project and the FPGA-based hardware platform, a generic interface is used. The generic interface realizes the communication with different processing elements implemented in the FPGA-based hardware platform transparent to the user.

2.5 Safety Standards and Certification Concepts

Some concepts related to modular certification have already been adopted by current standards and thus have found their way into the state of the practice. This is particularly true for the fields of automotive systems because the trend towards modularized architectures has been particularly strong in this field.

2.5.1 Safety Impact of DESERVE

Modularization of a common ADAS platform comes with a clear impact on safety. Modules will interact, for example on Missed Trigger Interaction, Shared Trigger Interaction, Sequential Action Interaction and/or Looping Interaction.

Module interaction implies that any change in operation of one module (feature) can be attributed in part or in whole to the presence of any other module (feature) in the operational environment, as illustrated in the Figure 2.15.

2.5.2 Functional Safety of Road Vehicles (ISO 26262)

The international standard ISO 26262 for the functional safety of street vehicles contains the so-called concept of Safety Element out of Context (SEooC).

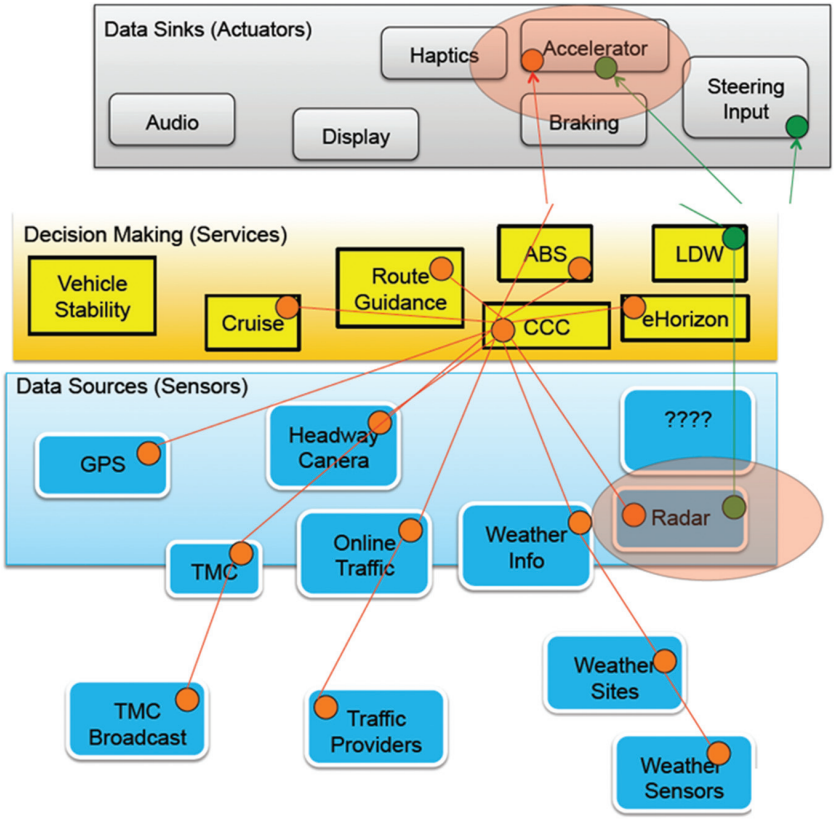

Figure 2.15 Module interaction implies changes in system behavior.

A SEooC is defined as a component for which there is no single predestinated application in a specific system. Therefore, the SEooC developer does not know the concrete role the product has to play in the safety concept. Subsystems, hardware components, and software components may be developed as SEooCs. Typical software SEooCs are reusable, application independent components such as operating systems, libraries, or middleware in general.

For SEooC development, the standard suggests specifying assumed safety requirements and developing the system according to these requirements. When the SEooC is to be used in a specific system, the system developer has to specify the demanded requirements, which can subsequently be checked against the assumed requirements. If there is a match between the demanded and the guaranteed (assumed) requirements, system and component are compatible.

The standard does not provide any suggestions or methods on how to identify safety requirements such as to increase the chance that assumed and real requirements will actually match. The standard specifies a relatively coarse-grained process for embedding a SEooC development into the standard's safety lifecycle. This approach deals with hierarchical modularization since it focuses on the SEooC's role as a sub-component of a system.

In general, integration of the SEooC is expected to be done at development time and thus there is no explicit support for open systems where components are to be integrated dynamically.

2.5.3 Guidelines Related to ISO 26262

ISO 26262 is a derivative of IEC 61508, the generic functional safety standard for electrical and electronic (E/E) systems. Ten volumes make up ISO 26262. It is designed for series production cars, and contains sections specific for management, concept and development phase, production, operation, service and decommission.

The ISO 26262 requires the application of a "functional safety approach", starting from the preliminary vehicle development phases and continuing throughout the whole product lifecycle.

The DESERVE project focuses on the concept and development (at system, hardware and software level) phases of the lifecycle. During these phases, the main steps defined by the Standard are:

Item definition: the Item has to be identified and described. To have a satisfactory understanding of the item, it is necessary to know about its functionality, interfaces, and any relevant environmental conditions.

Hazard analysis and risk assessment: to evaluate the risk associated with the item under safety analysis, a risk assessment is required. The risk assessment considers the functionality of the item and a relevant set of scenarios. This step produces the ASIL (Automotive Safety Integrity Level) level and the top level safety requirements.

The ASIL is one of the key concepts in the ISO 26262. The intended functions of the system are analyzed with respect to possible hazards. The ASIL asks the question: "If a failure arises, what will happen to the driver and to associated road users?".

The risk of each hazardous event is evaluated on the basis of frequency of the situation (or "exposure"), impact of possible damage (or "severity") and controllability.

The ASIL level is standardized in the scale: QM: quality management, no-risk and A, B, C, D: increasing risk with D being the most demanding. The ASIL shall be determined without taking into account the technologies used in the system. It is purely based on the harm to the driver and to the other road users.

Identification of technical safety requirements: the top level safety requirements are detailed and allocated to system components.

Identification of Software and Hardware safety requirements: The technical safety requirements are divided into hardware and software safety requirements. The specification of the software safety requirements considers constraints of the hardware and the impact of these constraints on the software.

To take into account the functional safety approach, the DESERVE applications should consider the application of the following main points: analyze risk early in the development process; establish the appropriate safety requirements and consider these requirements in software and hardware development.

The impact of the standard is different for the development of warning functions, control functions or automated driving functions.

2.5.4 Safety and AUTOSAR

In the automotive domain, Östberg and Bengtsson [14] propose an extension to AUTomotive Open System Architecture (AUTOSAR) which consists of a safety manager that actively enforces the safety rules described in dynamic safety contracts. Their main contribution is a conceptual model of safety

architecture suitable for runtime based safety assessment. Openness and Adaptivity were both addressed.

Also in the automotive domain, Frtunikj et al. [15] present a runtime qualitative safety assessment that considers Automotive Safety Integrity Level (ASIL) and its decompositions in open automotive systems. In their solution, the authors consider the modularization of safety-assessment using Safety Elements out of Context (SEooC) from ISO 26262. In their approach, the SEooC was extended and the safety-assessment is done at runtime by a Safety Manager component.

2.5.5 Safety Mechanisms for DESERVE Platform

As an example, this paragraph summarizes some features of the safety mechanisms that are available by Infineon's multi-core platform AURIX which represents a potential instance of DESERVE platform (development level 3). Its safety documentation includes:

- Safety case report providing the arguments with evidence that the objectives of the ISO 26262 and the safety requirements for a component are complete and satisfactory.
- FMEDA (customer and Infineon proprietary document)
- Safety manual including an overview of the assumed application use cases and guidance for the application level, a summary of safety features and mechanisms and their recommended use as well as the summary of achieved safety metrics and resulting ASIL compliance [13].

The AURIX microcontroller platform is developed as a SEooC (Safety Element out of Context) and provides the safety mechanisms summarized in Figure 2.16. It provides a Safe Computation Backbone compliant with ISO 26262 ASIL D (this includes Single Point Fault Metric fully supported by HW mechanisms and Latent Fault Metric supported by SW (SafeTlib), Logic MIST, MBIST). Support criteria for coexistence of elements are enabled through a layered protection system (covering CPU tasks, Shared Memories, Peripherals), CPU supervisor/user privileges, Safety Task Attribute and a rich set of counters & watchdogs for program flow & temporal monitoring. SEooC deliverables are the Safety Library (SafeTlib), Safety Manual to support SEooC integration and FMEDA to support computation of the ISO 26262 Metrics.

Top Level Safety Requirements (TLSR) related to the Microcontroller I/O sub-system are specified by the system integrator, as these vary for

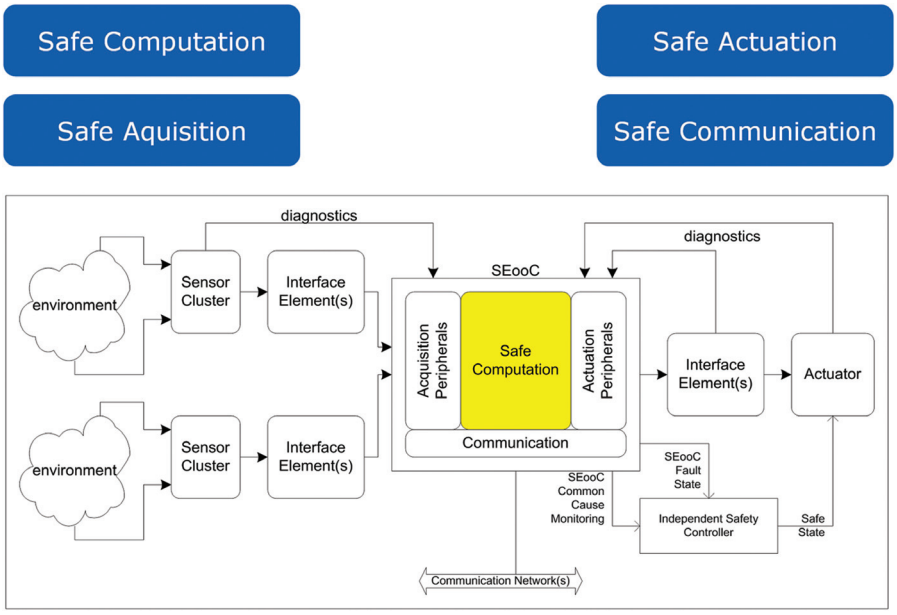

Figure 2.16 SEooC safety mechanisms.

each application. TLSR1 (ASIL D) requires to avoid false output of the microcontroller for longer than the FTTI (Fault Tolerance Time Interval, Figure 2.17), while TLSR2 (ASIL B) only require to avoid unavailability of a safety mechanism for longer than one driving cycle.

The Fault Tolerant Time Interval is more precisely defined by Figure 2.18. The application dependent fault detection time worst case is the diagnostic time interval. The fault detection time depends on the safety mechanism. The fault reaction time is the sum of failure signaling time and failure reaction time. Failure signaling time depends on the microcontroller architecture, while failure reaction time depends on the application. The failure signaling time is composed by the alarm forwarding time plus the alarm processing time plus the failure signaling time.

Safety requirements

With the AURIX as basis for DESERVE platform realization, it fulfils the targets according to ISO 26262-5, 8.4.5, which defines requirements for ISO 26262 metrics. To achieve ASIL D, for instance, the single point failure metric (SPFM) needs to reach minimum 99% and the latent fault metric (LFM) needs to reach 90% or above. The minimum values of SPFM and LFM shall

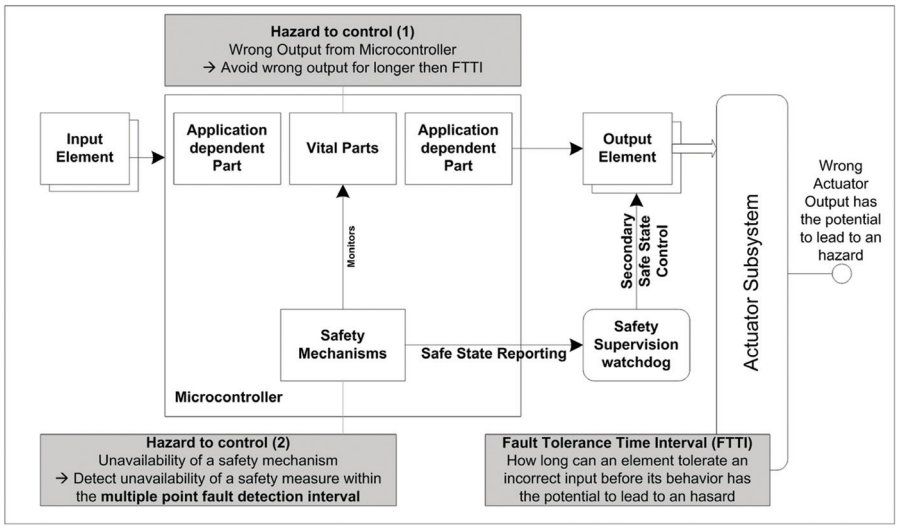

Figure 2.17 Top level safety requirements.

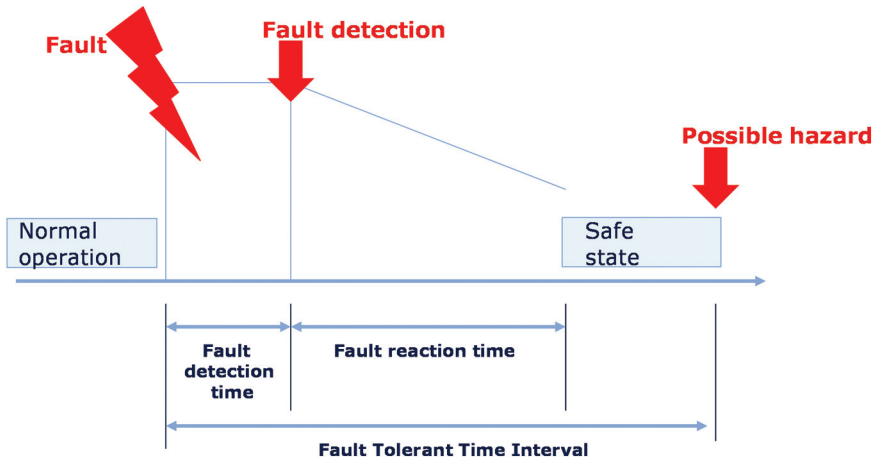

Figure 2.18 Fault tolerant time interval (FTTI) definition.

be reached by every vital part. The SPFM threshold levels shall be reached both for permanent and for transient faults. For a given ADAS application SPFM, LFM and PMHF (probabilistic metric related to hardware failures) metrics are estimated based on the vital, critical and application-dependent parts utilization.

In terms of PMHF for ASIL D safety goal, ISO 26262-5 requires a metric of less than 10 FIT (failure in time, referring to 10ˆ9 hours). ISO 26262-5 9.4.3.6 and 9.4.3.7 specify the relationship between ASIL and FCR and DC (Residual Faults). To meet ASIL D requirements the diagnostic coverage for a FCR5 part shall be $> 99.99\%$. The safety mechanisms are designed to achieve coverage of 99.99%.

Safety architecture

The safety architecture goal is to provide a safe computation platform for up to ASIL D safety applications according to ISO 26262, as this ASIL level is required for most next generation ADAS. To achieve this level, safe computation hardware and software, safe operating system as well as safe software architectures are required.

The generic elements (vital parts) of a safe computation hardware platform are summarized in Figure 2.19. Safe CPU requires hardware redundancy, realized by delayed lockstep CPU with enhanced timing and design diversity. Safe SRAMs allows information redundancy (realized by standard SECDED ECC, address signatures). Also safe Flash memory is needed for information redundancy (realized by an enhanced ECC with more than 99% coverage of arbitrary multiple-bit fault). Enhanced error detection codes for covering data & addressing faults lead to safe interconnects and support information redundancy. The clock system frequency range monitors using internal high precision independent clock source, internal & external watchdogs.

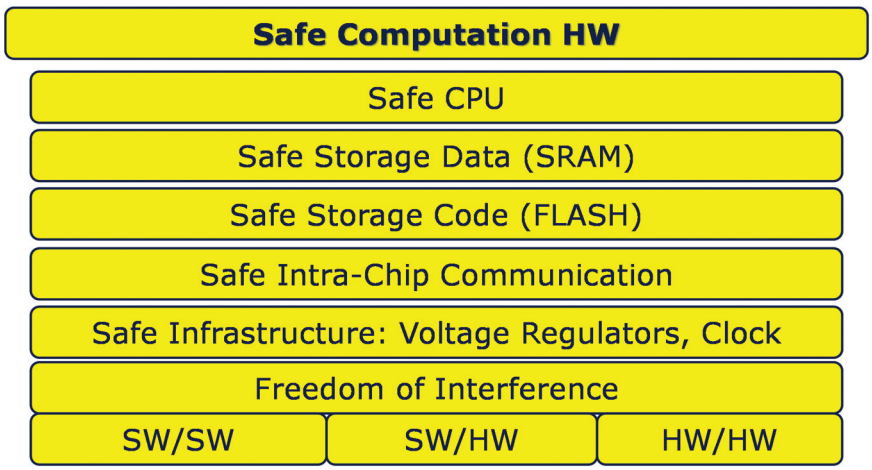

Figure 2.19 Generic elements of safe computation hardware platform.

Finally power supply range monitoring is implemented for the internal regulators.

To achieve a safe computation software platform an ASIL D compliant operating system needs to be used featuring memory protection and time protection. Further it needs to provide services for program flow monitoring, end-to-end communication safety protocols as well as safe interrupt vector generation. ASIL D compliant software is required to be developed according to ISO 26262 part 6.

The AURIX platform ensures freedom of interference at software level by means of SW isolation, while freedom of interference at hardware level is guaranteed by HW isolation. The CPU MPU (memory protection unit) monitors the direct access to the local memories, applies to software tasks and allows dynamic re-configuration. The bus MPU monitors the SRAM accesses via interconnect. Finally register access protection monitors write access rights to module registers.

References

[1] DESERVE deliverable D1.2.1 – Development platform requirements.
[2] DESERVE deliverable D1.3.1 – Development platform specification.
[3] DESERVE deliverable D1.3.2 – Method and tools specifications.
[4] DESERVE deliverable D2.5.2 – Platform system architecture.
[5] DESERVE deliverable D2.5.4 – Standard interfaces definition.
[6] AUTOSAR, http://www.autosar.org
[7] ISO 26262, Road vehicles – Functional safety (www.iso.org).
[8] A. Sandberg, D. J. Chen, H. Lönn, R. Johansson, L. Feng, M. Törngren, S. Torchiaro, R. Tavakoli-Kolagari, A. Abele – Model-based Safety Engineering of Interdependent Functions in Automotive Vehicles Using EAST-ADL2, Lecture Notes in Computer Science, Volume 6351, Series: Computer Safety, Reliability, and Security (SAFECOMP), Pages 332–346. Springer Berlin/Heidelberg, 2011. ISSN 0302-9743.
[9] www.interactive-ip.eu
[10] www.haveit-eu.org
[11] S. Durekovic (NAVTEQ), Perception Horizon: Approach to Accident Avoidance by Active Intervention, Workshop "How can new sensor technologies impact next generation safety systems?" IEEE IV 2011, June 5 2011, Baden–Baden.
[12] DESERVE Deliverable D2.2.1 – Perception layer Preliminary Release.

[13] AURIX Safety Manual, Infineon confidential document, no. AP32224, v1.1, dated Sept. 2014.

[14] K. Östberg und M. Bengtsson, "Run time safety analysis for automotive systems in an open and adaptive environment," in SAFECOMP 2013 – Workshop ASCoMS (Architecting Safety in Collaborative Mobile Systems), Toulouse, France, 2013.

[15] J. Frtunikj, M. Asmbruster und A. Knoll, "Data-Centric Middleware support for ASIL assessment and decomposition in open automotive systems".