

Achieving a Scalable E/E-Architecture Using AUTOSAR and Virtualization

Dominik Reinhardt and Dirk Kaule
BMW AG

Markus Kucera
University of Applied Science Regensburg

ABSTRACT

Today's automotive software integration is a static process. Hardware and software form a fixed package and thus hinder the integration of new electric and electronic features once the specification has been completed. Usually software components assigned to an ECU cannot be easily transferred to other devices after they have been deployed. The main reasons are high system configuration and integration complexity, although shifting functions from one to another ECU is a feature which is generally supported by AUTOSAR. The concept of a Virtual Functional Bus allows a strict separation between applications and infrastructure and avoids source code modifications. But still further tooling is needed to reconfigure the AUTOSAR Basic Software (BSW). Other challenges for AUTOSAR are mixed integrity, versioning and multi-core support.

The upcoming BMW E/E-domain oriented architecture will require all these features to be scalable across all vehicle model ranges. Entire vehicle E/E clusters, with heterogeneous software environments, must be encapsulated and shifted from one ECU to another while still meeting all postulated requirements. Virtualization technology could be a reasonable compromise between freedom from interference among integrated systems, scalable system architecture and efficient runtime environment.

This work gives a brief vision of domain oriented architecture within the automotive domain by means of AUTOSAR technology and discusses the requirements for such a scalable system. It is discussed, whether the requirements can be met with AUTOSAR or if additional abstraction or separation layers are needed. Finally we outline the possible usage of a hypervisor in combination within the AUTOSAR Basic Software using multicore technology and propose a mixed system architecture supporting the domain oriented architecture.

CITATION: Reinhardt, D., Kaule, D. and Kucera, M., "Achieving a Scalable E/E-Architecture Using AUTOSAR and Virtualization," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 6(2):2013, doi:10.4271/2013-01-1399.

INTRODUCTION

To tackle the complexity associated with the high number of functions and ECUs in a modern vehicle, the current electronics architecture is based on functional domains like chassis, power-train and infotainment. This architecture has to be scalable not only within a single vehicle with varying optional equipments, but also throughout all product lines ranging from subcompact class cars to high-end luxury vehicles. Each product line has unique requirements regarding available hardware components, installation space, memory capacity or energy consumption. Therefore, the assignment of logical functions to physical ECUs is always specific to the target vehicle. It must be at the same consistent with the underlying E/E-architecture.

According to the AUTOSAR standard [21], software components can easily be shifted between electronic control units. ECU developers have to accept some integration overhead in order to get flexible, high scalable systems, prepared for upcoming changes in the future. Some restrictions, like different Basic Software versions or mixing software components with different Automotive Safety Integrity Levels (ASIL), will slow down or block this process. AUTOSAR 4.1.1 will offer several mechanisms to support functional system safety and offers plenty of software enhancements to fulfill the requirements of ISO 26262 [9] out of the box, like partitioning for isolating software components and Basic Software. In contrary to it, common known virtualization techniques are able to separate multiple

AUTOSAR integrations from each other, which were former located on different ECUs. Multi-core technologies will further enrich this process.

Virtualization creates an additional abstraction layer between hard- and software, in special the operating system, and enables the integration of multiple, virtual ECUs on a single hardware platform. It also provides a safe and secure separation of software partitions [26], supports a scalable architecture and enables Large Scale Software Integration (LSSI) [22] on single hardware devices. This technology is already employed in industries like avionics and military, but not in the automotive domain [12], [13], [14] and [26].

In this paper we present an analysis on different approaches for scalable electronics architectures. We cover several scenarios with impact on a stable architecture like merging or omitting of domain controllers. For each scenario it is discussed whether already existing AUTOSAR features can achieve the requirements of a LSSI system or need to use an additional hypervisor layer. If the operation of a qualified hypervisor in an automotive E/E-system is justified, the resulting requirements for operation will be given.

CHALLENGES FOR DOMAIN ORIENTED E/E ARCHITECTURE

To master the upcoming E/E system complexity, vehicle functions shall be ordered by a domain specific manner as proposed in [22] and [23]. Thus, for any automotive domain, an individual domain controller is supposable (see Figure 1). Onto these ECUs mainly OEM specific functionality should be integrated. Domain controllers are typically featured with powerful hardware resources (e.g. multi-core processors). Every domain cluster can control ancillary sub controllers and is strictly connected to its domain controller. Sub controller should generally integrate commodity functionality. They are featured with less powerful hardware resources.

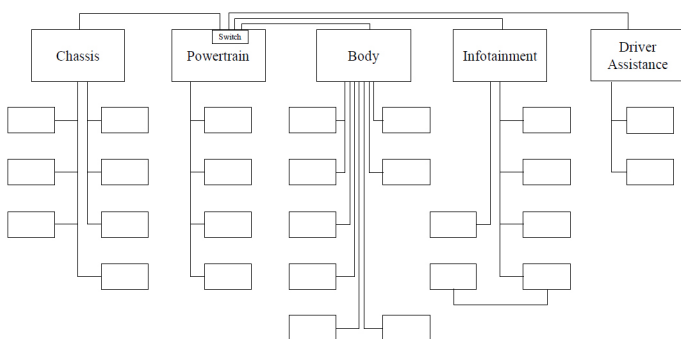


Figure 1. Vision of domain oriented architecture

Domain controllers will be interconnected via a fast transport medium. Therefore, data network protocols like Ethernet come into consideration. Sub controllers are connected to their domain controller with typical field buses (e.g. CAN, LIN, FlexRay).

To save energy and reduce carbon dioxide emissions the partial networking concept is introduced to AUTOSAR version 4.0. Selected nodes can be individually powered down according to the active vehicle state. Domain controllers identify this mode and react with activating or deactivating specific sub controllers within their area of influence.

The systems scalability is a proper requirement. Therefore interfaces, both between domain and sub controller and between domain controllers, shall be well defined. In order to support robust interfaces for in-vehicle hard- and software, the basic structure of domain architecture (see Figure 1) has to be consistent across all car lines. In contrary, the system must be flexible to be applicable for every vehicle class and its individual special equipment. To achieve this goal, respectively every domain cluster must provide different levels of scalability. For cars with less built-in functionality, the amount of ECUs can be cut down and indispensable software components have to be shifted to another domain controller.

As a result of this approach, in an upper-class car two functions may be partitioned on separate ECUs, while in a lower-class car the same function is less demanding and could run on a single CPU. Assume that in a given upper-class model range the driver assistance domain consists of a domain controller and two sub-ECUs (see Figure 2). If the E/E-architecture should be kept consistent across the model portfolio, even a car with no or few driver assistance functions will have to implement at least one driver assistance domain controller to keep the interfaces stable that are supplied for other vehicle domains. This makes sense from an architectural point of view, but not in terms of bill of material. In alternative, the remaining functions are transferred to another domain controller, e. g. power-train.

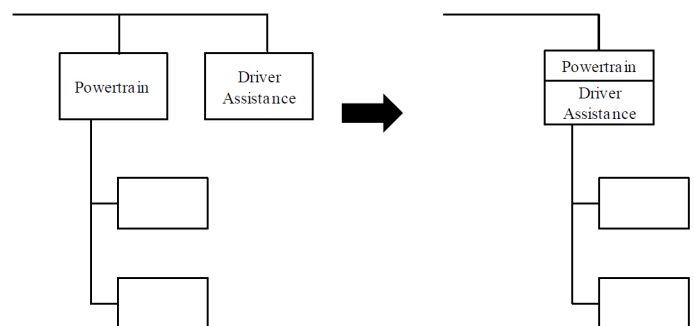


Figure 2. Repartitioning of domain controller functionality

POSSIBLE SAFETY REQUIREMENTS FOR DOMAIN CONTROLLERS

The ISO 26262 is the internationally agreed safety standard of the automotive industry and represents a complete safety project management cycle. Within the process step of the hazard analysis and risk assessment of vehicle functions,

Automotive Safety Integrity Levels are classified corresponding to their respective hazard. The classifications A, B, C and D exist, where ASIL D is specified as having the highest level of potential danger or safety risk. The determined classification is required for following assessments within the functional safety concept to derive local safety goals. Within the technical safety concept, solutions are to elaborate how to reach the safety goal in a technical manner.

Architectural Safety Requirements of ISO 26262

In the following section we discuss safety requirements of ISO 26262 based on software architecture, which come into consideration for a domain controller. Virtualization technologies and the introduction of an additional layer are well known software architecture principles [15] and shall be analyzed more specific within this work. Therefore, requirements concerning the needs of well known system designs at software application level are in focus of the following comparison. Particularly, seven selected requirements of ASIL C and D of [9] are listed.

The ISO 26262 demands that software components in general should be hierarchically structured to support software reusability (1). This means that software architecture is divided into layers. Furthermore, the combination of software components shall be loose coupled (2), in order to avoid problems of ramified dependencies (cascading failures).

On a functional perspective memory protection concepts must be introduced to the system (3). To isolate memory sections the usage of a hardware based memory protection unit (MPU) is recommended. It is a main condition for the following safety mechanisms.

To prevent timing failures, a suitable scheduling technique must be applied (4). Therefore, heterogeneous process quantities (synchronous and asynchronous tasks) have to be handled with care, because as they are caused by sporadic events and are difficult to predict. Thus, for safety-relevant applications, which are integrated together on one platform, there is a need for special arrangements to monitor the program flow and guarantee task execution times.

Early error detection at architectural level represents one of the most important requirements for safety-relevant software development in ISO 26262 (5). Monitor concepts reflect an additional way to detect incorrect behavior or corrupt signals and bring the system to a safe state in a proper way. To achieve reliable information, redundant voting mechanisms are conceivable and can be realized by diverse calculation implementation.

A typical example is given by the E-Gas Monitoring Concept [1] as illustrated in the AUTOSAR standard 4.0. There, an additional signal processing takes place in the context of a complex device driver module, concurrent to the actual received signal within the AUTOSAR BSW.

In addition, there shall be the possibility to deal with defective software during its running time (fault tolerant system). Besides static recovery mechanisms, parallel execution paths are required, which are executed independently (6). The decisive factor is the level of abstraction for independence and parallelism to achieve the particular safety goal. This must be defined in the respective safety concept.

Furthermore, possibilities for graceful degradation [9] scenarios have to be created (7), in order to keep the system's behavior in a defined and safe way even if the fault hypothesis is violated. This is a very important but complicated procedure. Therefore, a prioritization concept for tasks or even functions must be introduced. In case of error less important functionality shall be interrupted to ensure a safe system design.

Additional Requirements for Domain Oriented Systems

Within a domain controller many vehicle functions are integrated, which are detached by different integrity levels. To counteract the increased development costs, safety-relevant functions shall be decomposed in an appropriate manner: For this purpose, a function which is specified by ASIL D can be decomposed into two independent components specified by ASIL B (D). In addition, other special requirements have to be observed [9]. Particularly by using a multi-core system, independent and redundant software systems can be built.

A domain controller is a dynamic system which is implemented on a static integration platform. Therefore, to ensure freedom from interference, safety-relevant applications shall not be corrupted by non safety-relevant applications and must be secured to do their specified execution. Appropriate scheduling mechanisms must be introduced for this purpose [10] and [11]. To achieve this goal, several algorithms could be hybridized. Furthermore, the operating system must ensure that tasks meet their deadlines and can respond in case of error in appropriate ways. This requires individual functional clusters to be restarted. A total breakdown of the software, a core, or the whole system shall be avoided in any case.

Due to the Large Scale Software Integration of functions installed in the control unit, safety mechanisms are not allowed to compromise the efficient task or system execution. Therefore, patterns such as monitor concepts, redundant software components or isolated, respectively detached outsourced software units, must be used in a sustainable manner. Simultaneously, through the omission of a vehicle function, the safety goal of adjacent applications shall not be breached.