An Integrated Architecture for Future Car Generations

P. Peti, R. Obermaisser
Vienna University of Technology
Austria
{php,ro}@vmars.tuwien.ac.at

F. Tagliabo, A. Marino, S. Cerchio
Centro Ricerche Fiat
Orbassano, Italy
{fulvio.tagliabo,reti1,stefano.cerchio}@crf.it

Abstract

Depending on the physical structuring of large distributed safety-critical real-time systems, one can distinguish federated and integrated system architectures. The DECOS architecture combines the complexity management advantages of federated systems with the functional integration and hardware benefits of an integrated approach. This paper investigates the benefits of the DECOS integrated system architecture as an electronic infrastructure for future car generations. The shift to an integrated architecture will result in quantifiable cost reductions in the areas of system hardware cost and system development.

In the paper we present a current federated Fiat car E/E architecture and discuss a possible mapping to an integrated solution based on the DECOS architecture. The proposed architecture provides a foundation for mixed-criticality integration with both safety-critical and non safety-critical subsystems. In particular, this architecture supports applications up to the highest criticality classes (10^{-9} *failures per hour), thereby taking into account the emerging dependability requirements of by-wire functionality in the automotive industry.*

1. Introduction

One can distinguish two classes of systems for distributed applications, namely *federated* and *integrated systems*. In a federated system, each application subsystem has its own dedicated computer system, while an integrated system is characterized by the integration of multiple application subsystems within a single distributed computer system. Federated systems have been preferred for ultra-dependable applications due to the natural separation of application subsystems, which facilitates fault-isolation and complexity management.

Integrated systems, on the other hand, promise massive cost savings through the reduction of resource duplication. In addition, integrated systems permit an optimal interplay of application subsystems, reliability improvements with respect to wiring and connectors, and overcome limitations for spare components and redundancy management. An ideal future system architecture would thus *combine the complexity management advantages of the federated approach, but would also realize the functional integration and hardware benefits of an integrated system* [7, p. 32]. The challenge is to devise an integrated architecture that provides a framework with generic architectural services for integrating multiple application subsystems within a single, distributed computer system, while retaining the error containment and complexity management benefits of federated systems.

There is a steady increase in electronics in automotive systems in order to meet the customer's expectation of a car's functionality. Cars are no longer simple means of transportation but rather need to convince customers with respect to design, performance, driving behavior, safety, infotainment, comfort, maintenance, and cost. In particular during the last decade, electronic systems have resulted in tremendous improvements in passive and active safety, fuel efficiency, comfort, and on-board entertainment. In combination with a "1 Function - 1 Electronic Control Unit (ECU)" design philosophy that is characteristic for federated architectures, these new functionalities have led to electronic systems with large numbers of ECUs and a heterogeneity of communication networks.

However, in order to satisfy the industrial demands on performance, dependability and cost with respect to a large variety of different car platforms, the current state-of-the-art system development methodology is heavily imposed to be reviewed, because of

- the strong competition among the carmakers;
- the requirement to continuously improve comfort functionality with stringent time-to-market constraints;
- the introduction of by-wire vehicle control and those functions introduced following normative

pressure (e.g., fuel consumptions);

- a demand of greater versatility of the vehicle, conceived in a new view about modularity and standardization.

In particular, a low number of ECUs offers significant benefits with respect to architecture complexity, wiring, mounting, hardware cost and many others. Thus, a reduction of the number of ECUs is of great interest.

It is the objective of this paper to present the DECOS integrated architecture for dependable embedded control systems for future automotive systems. This integrated architecture is based on a time-triggered core architecture and a set of high-level services that support the execution of newly developed and legacy applications across standardized technology-invariant interfaces. Rigorous encapsulation guarantees the independent development, seamless integration, and operation without unintended mutual interference of the different application subsystems. The integrated architecture offers an environment to combine both safety-critical and non safety-critical subsystems within a single distributed computer system. The architecture exploits the encapsulation services to guarantee that software faults cannot propagate from non safety-critical subsystems into subsystems of higher criticality.

In this paper, we map a present automotive infrastructure onto the DECOS architecture and elaborate on the respective benefits, such as independent development, the assignment of integration responsibility, and an optimized use of resources through architectural gateway services. In order to maximize the economic impact, we propose a portable architecture that can be deployed in all segments of the car manufacturer (segment from A to E and also for luxury cars). Thereby a reduction of cost due to the increase in volume is expected. To achieve the required flexibility and portability, the architecture is be based on general purpose hardware and a modular software concept allowing to protect the intellectual property of vendors.

This paper is structured as follows. Section 2 gives an overview of the DECOS integrated architecture, presenting the main underlying concepts. In Section 3, we discuss the current state-of-the-art of automotive architectures. The mapping to an integrated solution is the focus of Section 4. The discussion presented in Section 5 evaluates the integrated architecture based on prevalent E/E automotive trends.

2. The DECOS Integrated Architecture

The DECOS architecture [13] offers a framework for the development of distributed embedded real-time systems integrating multiple Distributed Application Subsystems (DASs) with different levels of criticality and different requirements concerning the underlying platform. Structuring rules guide the designer in the decomposition of the overall system both at a functional level and for the transformation to the physical level. In addition, the DECOS integrated architecture aims at offering to system designers generic architectural services, which provide a validated stable baseline for the development of applications.

2.1. Functional System Structuring

Until now, the introduction of *structure and hierarchical relationships* represents the only promising approach for understanding complex systems with large numbers of parts and interactions between these parts [26, chap. 8]. This insight applies to all technical systems and in particular to the mastering of large, complex real-time computer systems. The complexity of a large real-time computer system can only be managed, if the overall system can be decomposed into nearly-independent subsystems with linking interfaces that are precisely specified in the value and time domain [14]. Near-independence is the ability of a subsystem to serve its purpose independently from the detailed structure of other subsystems, i.e. only based on the specification of the linking interfaces of the subsystems.

For the provision of application services at the controlled object interface, the real-time computer system is divided into a set of nearly-independent subsystems, each providing a part of the computer system's overall functionality. We denote such a subsystem as a *Distributed Application Subsystem (DAS)*, since the implementation of the corresponding functionality will most likely involve multiple components that are interconnected by an underlying communication system. The implementation as a distributed system is a prerequisite for establishing fault-tolerance by redundantly performing computations at separate components that fail independently. Furthermore, a distributed solution becomes a necessity, when the resource requirements of the application providing the subsystem's functionality exceed the available resources of a single component.

An example for a DAS in the automotive domain is the steer-by-wire subsystem. With steer-by-wire [8] the transmission of the wheel rotation to a steering movement of the front wheel is performed with the help of electronically controlled actuators at the front axle. The main advantages in comparison with conventional steering systems are improvements with respect to crashworthiness, weight, and interior design.

In analogy to the structuring of the overall system,

we further decompose each DAS into smaller units called *jobs*. A *job* is the basic unit of work that employs the communication system for exchanging information with other jobs, thus working towards a collective goal. The interface between a job and the communication system is denoted as a *port*. Depending on the data direction, one can distinguish input ports and output ports. A job employs input ports for exploiting the services of other jobs, while output ports enable a job to provide its own services. Every job has access to its relevant transducers, either directly via the controlled object interface or via a communication system with known temporal properties.

2.2. Physical System Structuring

During the development of an integrated system the functional elements must be mapped to the physical building blocks of the platform. These building blocks are *clusters*, *physical networks*, *components* and *partitions*. A *cluster* is a distributed computer system that consists of a set of components interconnected by a *physical network*. A *component* is a self-contained computational element with its own hardware (processor, memory, communication interface, and interface to the controlled object) and software (application programs, operating system) [14], which interacts with its environment by exchanging messages across Linking Interfaces (LIFs). The behavior of a component can be specified in the domains of value and time. Components are the target of job allocation and provide encapsulated execution environments denoted as *partitions* for jobs. Each partition prevents temporal interference (e.g., stealing processor time) and spatial interference [22] (e.g., overwriting data structures) between jobs. In the DECOS architecture, a component can host multiple partitions and host jobs that can belong to different DASs.

2.3. Architectural Services

Generic architectural services separate the application functionality from the underlying platform technology in order to facilitate reuse and reduce design complexity. This strategy corresponds to the concept of platform-based design [25], which proposes the introduction of abstraction layers, which facilitate refinements into subsequent abstraction layers in the design flow.

The DECOS architectural services depicted in Figure 1 are such an abstraction layer. The specification of the architectural services hides the details of the underlying platform, while providing all information required for ensuring the functional and meta-functional

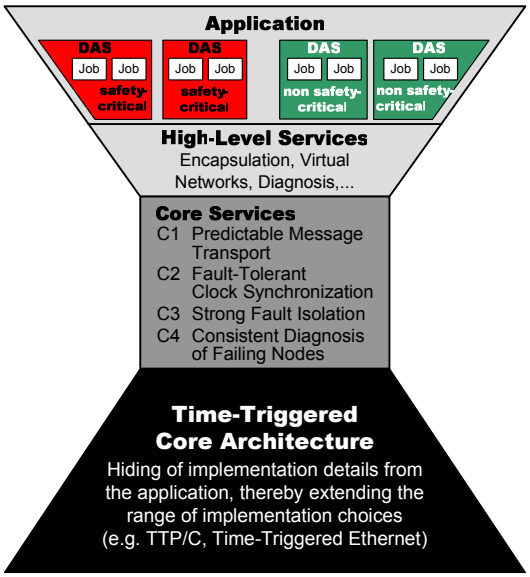

Figure 1. The DECOS Integrated System Architecture

(dependability, timeliness) requirements in the design of a safety-critical real-time application. The architectural services serve as a validated stable baseline that reduces application development efforts and facilitates reuse, because applications build on an architectural service interface that can be established on top of numerous platform technologies.

In order to maximize the number of platforms and applications that can be covered, the DECOS architectural service interface distinguishes a minimal set of *core services* and an open-ended number of *high-level services* that build on top of the core services. The core services include predictable time-triggered message transport, fault tolerant clock synchronization, strong fault isolation, and consistent diagnosis of failing components through a membership service. The small number of core services eases a thorough validation (e.g., permitting a formal verification), which is crucial for preventing common mode failures as all high-level services and consequently all applications build on the core services. Any architecture that provides these core services can be used as a core architecture [23] for the DECOS integrated distributed architecture. An example of a suitable core architecture is the Time-Triggered Architecture (TTA) [11].

Based on the core services, the DECOS integrated architecture realizes high-level architectural services, which are DAS-specific and constitute the interface for the jobs to the underlying platform. Among the high-level services are virtual network services and encapsulation services. On top of the time-triggered physical

network, different kinds of virtual networks can be established and each type of virtual network can exhibit multiple instantiations (see Figure 1). The encapsulation services ensure spatial and temporal partitioning for virtual networks in order to obtain error containment and control the visibility of exchanged messages.

3. Today's Automotive Architectures

To give an impression of the complexity and the amount of electronics in today's luxury cars take for example the electronic infrastructure of a Fiat car depicted in Figure 2. The distributed ECUs of each federated cluster of the car are interconnected via communication networks with different protocols (e.g., Controller Area Network (CAN) [21], Local Interconnect Network (LIN) [17]), physical layers, bandwidths (10 kbps–500 kbps), and dependability requirements.

The *Body Control* cluster as well as the *Telematic Info* cluster, are typically implemented via a low speed CAN bus (125 kbps), while the *Dynamic Vehicle Control* cluster is implemented via a high speed CAN bus (500 kbps). The *Infotelematic* system also uses a high speed CAN to exchange camera and video information. These multiple federated clusters are interconnected with a central gateway inside the *Body Computer*, allowing controlled data exchange between the *Dynamic Vehicle Control* cluster and the *Body Control* cluster, and access to the On-Board Diagnosis (OBD) system of each ECU. For on-board diagnosis either a dedicated serial line interconnects the ECUs or the diagnostic protocol is executed via the low speed CAN network.

Each of these clusters consists of nodes that are typically dedicated to a single job. This is frequently referred to as "1 Function - 1 ECU" design strategy. In combination with the need to adapt products to emerging trends and customer requests, manufactures are forced to steadily increase the number of deployed ECUs in order to improve the functionality of the car. For example, Fiat cars contain up to 40 ECUs. However, this trend of increasing the number of ECUs is coming to its limits, because of complexity, wiring, space and weight restrictions. For example, electrical connections are considered to be one of the most important failure causes in automotive systems. Field data from automotive environments has shown that more than 30% of electrical failures are attributed to connector problems [28]. With an average cost of 30-50 Euros per ECU this high number of ECUs bears significant potential for cost reduction.

3.1. System Integration

During system integration significant efforts are caused by unanticipated interactions between subsystems provided by different vendors. The sharing of communication resources in today's cars across different subsystems (e.g., systems based on the CAN protocol) makes it hard to fully test the functionality of a subsystem in isolation as it will be integrated in the car. As a consequence, there is the need for a comprehensive integration test by the car manufacturer to determine possible mutual interference of subsystems. In contrast, a system architecture with rigorous operational interface specification [14] and error containment can avoid the introduction of mutual interference during system integration. Such a temporally composable architecture [12] exhibits the benefit of dramatically decreasing integration costs, because the validity of test certificates from suppliers is not invalidated during system integration.

3.2. Complexity Control

Each subsystem (e.g., engine control, brake assistant) possesses a functional complexity that is inherent to the application. The functional complexity of a subsystem when implemented on a target system is dramatically increased in case the architecture does not prevent unintended architecture-induced side effects at the communication system. Since federated systems employ a dedicated computer system for each subsystem, the complexity of the system is lower compared to the integrated systems approach. The absence of interactions and dependencies between subsystems reduces the cognitive complexity to a manageable level. In today's cars we do not find a totally federated architecture nor an integrated one. In fact, the economic pressure in the automotive industry requires system designers to utilize the available communication resources for more than one subsystem without protecting the resources from mutual interference. For a deeper understanding consider an exemplary scenario with two subsystems. If the two subsystems share a common CAN bus [21], then both subsystems must be analyzed and understood in order to reason about the correct behavior of any of the two subsystems. Since the message transmissions of one subsystem can delay message transmission of the other DAS, arguments concerning the correct temporal behavior must be based on an analysis of both subsystems. In a totally federated system, on the other hand, unintended side effects are ruled out, because the two subsystems are assigned to separate computer system.

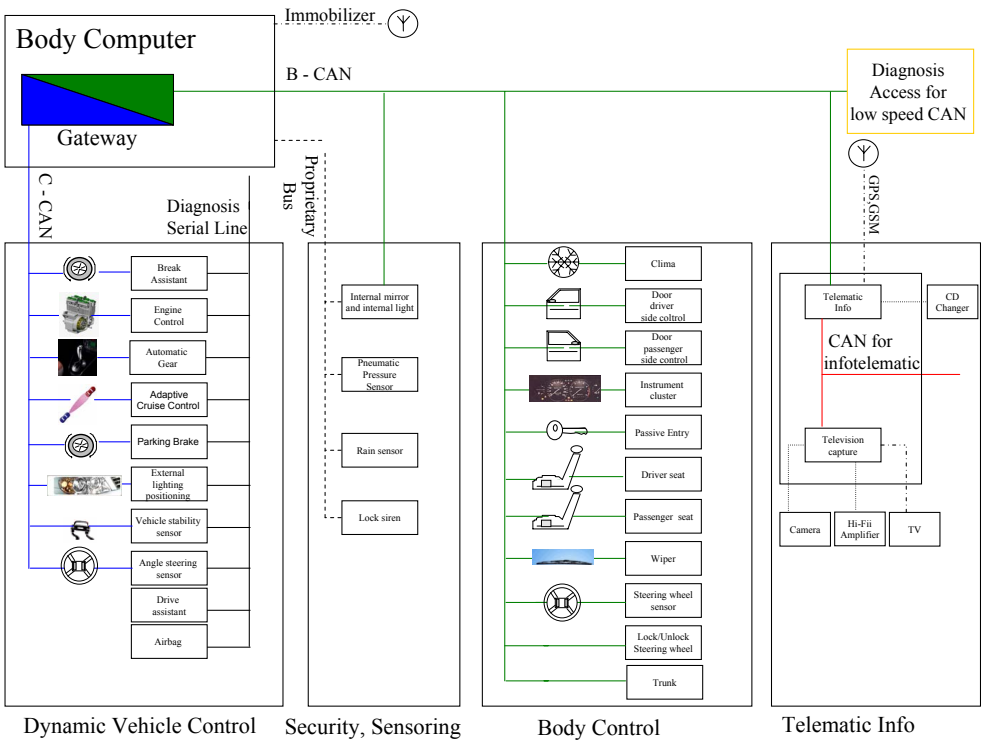

Figure 2. The Electronic Infrastructure of a Fiat Car

4. An Architecture for Future Car Generations

This section describes the proposed integrated architecture for future car generations. We start by discussing a hybrid top-down/bottom-up design strategy, improving the currently prevalent ECU-centric development process. The decomposing during the process results in a set of DASs. Thereafter, we elaborate on the DASs of a hypothetical future car and its constituting gateways. Finally, we show the physical structure of the integrated system.

4.1. Design Flow

The design flow of automotive distributed systems can be decomposed into three phases, the requirement analysis, the subsystem design, and the system integration phase [6] (see also Figure 3). As described in [20, 24] an ECU-centric design process prevails in the automotive industry. Such a bottom up process, however, bears significant drawbacks such as resource duplications, local instead of global quality-of-service optimization, and exponential growth in terms of system integration costs. Furthermore, the number of the deployed ECUs steadily increases to satisfy recent market trends and the customer's demand for new functionality.

The DECOS architecture, by contrast, also supports a top-down design approach. During the requirement analysis the system integrator captures the requirements of the overall system (i.e. the car electronics) and decomposes the system into nearly-independent subsystems (i.e. DASs). The requirement analysis provides the foundation for all later design stages. Here, the overall functionality of the system is specified and subsystems are identified to enable an independent development of DASs. As depicted in Figure 3 the result of this design phase is a set of DASs that comprise the electronic infrastructure of the car.

The structuring of the overall application functionality into DASs is guided by the following principles:

1. **Functional Coherence.** A DAS should provide a meaningful application service (e.g., brake-by-wire service of a car) to its users at the controlled object interface. By associating with a DAS an application service that is relevant in the actual application context, the mental effort in understanding the various application services is reduced. An application service can be analyzed by solely considering the jobs of the DAS, the interactions to the controlled object and the gateways to other DASs (inter-DAS interfaces). In particular, it is not nec-

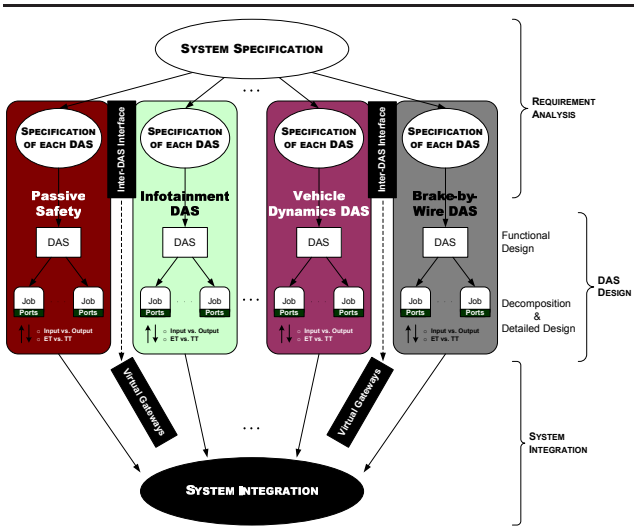

Figure 3. Design Flow

essary to possess knowledge about the internal behavior of DASs, other than the one providing the application service that is of interest.

2. **Common Criticality.** In general, the realization of safety-critical services is fundamentally different from the design of non safety-critical services. While the first incorporate fault-tolerance functionality and focus on maximum simplicity to facilitate validation and certification, the latter are usually characterized by a larger amount of functionality and the requirement of flexibility and resource efficiency. The integrated architecture takes this difference into account by distinguishing between safety-critical and non safety-critical DASs along with dedicated architectural services.

3. **Infrastructure Requirements.** A DAS possesses common requirements for the underlying infrastructure. A single virtual network is employed for exchanging message within the DAS. Consequently, common requirements (e.g., with respect to dependability, bandwidth and latency requirements, flexibility) are a prerequisite for deciding on a particular virtual network protocol (e.g., time-triggered or event-triggered) and a corresponding configuration (e.g., bandwidth).

Whenever significant differences in the above aspects are present, such as missing functional coherence or differences with respect to the infrastructure requirements, a DAS is split into smaller DASs for resolving these mismatches.

This *divide and conquer* principle can only be realized if the dependencies between DASs are made explicit in order to avoid hidden interactions (e.g., via the controlled object) that may prevent a seamless system integration. These DASs are then assigned and independently developed by different vendors. In general, each vendor may also depend on subcontractors to deliver the subsystem.

In order to ensure correct system integration the specification of inter-DAS relationships is of high importance. Inter-DAS interfaces as indicated in Figure 3 are used to specify common information within DASs (e.g., sensor information), possible interrelationships via the controlled object, and meta-functional aspects. This way, resources can be shared among DASs, thus avoiding resource duplication by eliminating sensors or using redundant sensory information to improve dependability.

The DAS design is typically performed by different vendors with expert knowledge in particular application domains (e.g., infotainment, braking systems). Independent development of a DAS allows to adopt the benefits of the federated systems design approach to be incorporated into the integrated systems design approach.

Finally, the system integrator needs to unify the separately developed subsystems into the overall system. System integration unites the separately developed subsystems into the overall system. An integrated system approach must provide solutions that reduce integration time and efforts (and consequently reduce integration costs). Smooth system integration is only possible, if the inter-DAS interfaces have been precisely specified and all vendors have performed implementations adhering to these interface specifications. During system integrations three main tasks need to be performed by the system integrator: the physical allocation of the jobs (of all DASs) to partitions taking dependability and resource constraints into account, the configuration of the virtual communication networks, and the realization of the virtual and physical gateways in order to provide emerging services.

4.2. Integrated System Structure of Car

In this subsection, we map the introduced electronic infrastructure of today's cars onto the DECOS integrated architecture. In addition, we replace state-of-the art powertrain domain functionality by by-wire subsystems to emphasize the suitability of the proposed DECOS architecture for mixed-criticality applications (i.e. safety-critical and non safety-critical applications). We split up the existing domains (e.g., powertrain, body) into smaller DASs. Smaller DASs are a key element to achieve the DECOS goals with respect to com-

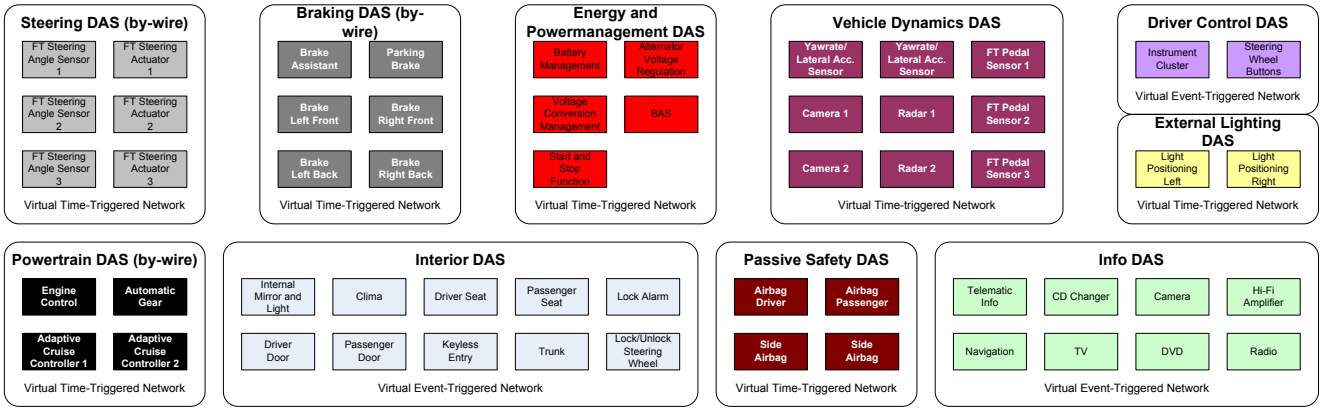

Figure 4. The Distributed Application Subsystems of the Integrated Automotive System

plexity management, independent development and error containment.

As described in Section 2.1, a DAS represents a nearly-independent subsystem [26, chap. 8], because it can be understood independently from the detailed structure of other DASs, i.e. only based on the specification of the jobs of the DAS and the gateways to other DASs. The controlled export of information through gateways enables the designer to abstract from the jobs in other DASs, considering only the link specification of the gateways [18].

The DECOS architecture encapsulates DASs both at the level of the communication activities (through encapsulated virtual network services [19]) and at the level of the computational activities (partitions in application computers [13]). Therefore, a design fault in a DAS (e.g., a job with a babbling idiot failure [10]) cannot affect the communication resources (e.g., bandwidth, guarantee of latencies) and computational resources (e.g., CPU time) available to other DASs. Naturally, the finer the subdivision into DASs, the more effective the encapsulation through the architecture becomes.

In a federated architecture, which assigns each DAS to its own dedicated computer system, a strategy with a large number of small DASs would not be feasible due to the cost resulting from increasing resource duplication (via separate networks and ECUs). However, in our proposed integrated architecture, the resulting larger number of DASs does not induce a larger number of physical networks and ECUs. Each DAS is provided as a virtual network on top of the physical time-triggered network of the core architecture. Similarly, the virtual gateways (see Section 4.3) for coupling DASs do not induce any additional ECUs and connectors.

Based on this line of reasoning, we introduce a finer granularity of DASs compared to the domains of today's automotive architectures (see Figure 4):

- **Steering DAS.** With steer-by-wire the transmission of the wheel rotation to a steering movement of the front wheel is performed with the help of electronically controlled actuators at the front axle. The main advantages in comparison with conventional steering systems are improvements with respect to crashworthiness, weight, and interior design.
- **Powertrain DAS.** The functionality of this DAS includes engine control, automatic gear control, and adaptive cruise control.
- **Braking DAS.** The braking DAS comprises the brake-by-wire functionality. Brake-by-wire systems remedy deficiencies of conventional hydraulic braking systems, such as aging of braking fluids, difficulties in routing of pipes, and the inconvenient feedback during ABS braking. Brake-by-wire systems incorporate brake power assist, vehicle stability enhancement control, parking brake control, and tunable pedal feel.
- **Vehicle Dynamics DAS.** In the vehicle dynamics DAS all sensory information that is relevant for controlling the dynamics of the vehicle is captured. By exporting these real-time images to the other DASs of the system the problem of resource duplication can be significantly reduced. In addition, sensor-fusion algorithms [5] can combine the measurements of different sensors to obtain more accurate real-time images.
- **Energy DAS.** The main purpose of this DAS is the optimization of the power distribution (electrical energy and power management techniques) for conserving the power available in the vehicle.

- **Passive Safety DAS.** The passive safety DAS intends to keep the passengers in the car and effectively decelerates the occupants in order to minimize harm in case of a crash.
- **Driver Control DAS.** In every car the instruments inform the driver about the current status of the car. It is the task of this DAS to update the instruments according to the information provided by the other DASs of the car. Furthermore, the commands of the driver derived from the buttons on the steering wheel are processed and disseminated to the respective DASs.
- **Interior DAS.** This DAS comprises the body electronics of the passenger compartment and accesses fieldbus network, such as those embedded in the doors and seats of the car. The functionality of this DAS includes the control of the doors (e.g., mirrors, window lifters), the seats (e.g., seat adjustment, the position memory), the climate control, and the lighting of the passenger compartment. Furthermore, this DAS ensures that only authorized persons have access to the car.
- **Info DAS.** Car drivers are no longer satisfied with cars being simple means of transportation. Today's luxury cars include GPS navigation systems, DVD players and high-end audio systems. In addition, voice control and hands-free speaker phones relieve the driver from concentrating on multimedia devices instead of traffic.
- **External Lighting DAS.** The external lighting DAS controls the rear lights of the car, as well as the position of the adaptive forward lighting.

The communication infrastructure provided to these DASs depends on the respective criticality and regularity of the communication activities. Time-triggered virtual networks handle the communication exchanges of all safety-critical and safety-relevant DASs. It has become widely accepted that safety-critical automotive applications employ the time-triggered control paradigm [23], because this control paradigm permits to guarantee a deterministic behavior of all safety-related message transmissions even at peak-load. In addition to hard real-time performance time-triggered control also supports temporal composability and facilitates the realization of fault-tolerance mechanisms. For this reason the communication infrastructure for the steer-by-wire, the brake-by-wire, the powertrain, the vehicle dynamics, the passive safety and the external lighting DAS are *time-triggered virtual networks* [19].

Event-triggered virtual networks, on the other hand, are the communication infrastructure of choice for those DASs having less stringent dependability requirements. Here, the flexibility and the efficient use of resources is more important than the determinism provided by the time-triggered control paradigm. For this reason, the jobs of the interior, power management, driver control, and infotainment DAS are interconnected by respective *event-triggered virtual networks* [19] in the presented architecture.

4.3. Gateways

By splitting the overall functionality of the car into multiple DASs the need for a coupling of individual DASs emerges. The presented architecture supports gateways as a generic architectural services for the interconnection of DASs. Gateways have significant advantages with respect to the elimination of resource duplication and the tactic coordination of application subsystems. In a large automotive system, different application subsystems typically depend on the same or similar sensory inputs and computations. Gateways allow to exploit system-wide redundancy of sensor information in order to increase reliability or reduce resource duplication. In addition, gateways permit the coordination of DASs in order to improve quality of control.

In the DECOS integrated architecture, we sharply distinguish between architecture level and application level. Based on this differentiation, we can identify two choices for the construction of a gateway. A hidden gateway performs the interconnection of virtual networks at the architecture level. Generic architectural services – although parameterized by the application requirements – are transparent to the jobs at the application level. A visible gateway, on the other hand, performs the interconnection at the application level.

A *virtual gateway* [18] interconnects two virtual networks of two respective DASs by forwarding information contained in the messages received at the input ports of one virtual network onto the output ports towards the other virtual network.

In general, the semantic and operational properties of the input ports at one virtual network can be different to the semantic and operational properties of the output ports at the other virtual network. The resulting property mismatch [4] is resolved by the gateway by performing transformations on the information passing through the gateway. For syntactic transformations, the gateway requires a description of the syntactic format (i.e. the data types) of the messages passing through the gateway and rules for transforming the different syntactic transformations into each other. If the DASs interconnected by the gateway exhibit different operational specification styles [14], the gateway requires additional buffering functionality for the exchange of information between virtual networks with

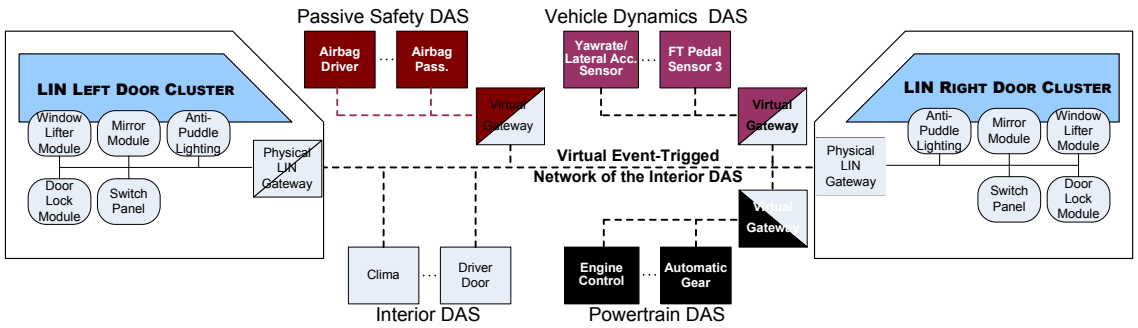

Figure 5. Physical and Virtual Gateways of the Interior DAS

varying rigidity of temporal specifications. For example, such a scenario occurs, if one DAS operates time-triggered and the second DAS operates event-triggered.

In addition, virtual gateways ensure encapsulation by using a filtering specification in the value and temporal domain in order to restrict the redirection of messages through the gateway. In general, only a fraction of the information exchanged at one virtual network will be required by jobs connected to the virtual network at the other side of the gateway. By restricting the redirection through the gateway to the information actually required by the jobs of the other DAS, the gateway not only improves resource efficiency by saving bandwidth of unnecessary messages, but also facilitates complexity control.

In addition to the interconnection of virtual networks, the presented integrated architecture offers architectural gateway services for interacting with the environment via *physical gateways*. In general, the interaction of the computer system with the controlled object and the human operator can occur either via a direct connection to sensors and actuators or via a fieldbus network. The latter approach simplifies the installation – both from a logical and a physical point of view – at the expense of increased latency of sensory information and actuator control values.

Since the prevalent low-cost fieldbus protocol in the automotive domain is LIN [17], the proposed architecture supports physical LIN gateways, each acting as a master for the slaves of the physical LIN bus. Figure 5, which exemplifies the role of hidden gateways in the functional structure of the future automotive architecture, depicts two of these LIN gateways for the interconnection of the interior DAS with the LIN fieldbusses located at the front doors. The driver door job and passenger door job exchange information with the actuators and sensors in the doors in order to control door locks, window lifters, mirrors, and anti-puddle lighting.

In addition, Figure 5 also contains virtual gateways for the interconnection of virtual networks. The interior DAS constructs real-time images capturing the state of the passenger compartment of the vehicle (e.g., status of the doors, seats, lighting, climate control) that are also important to other DASs. For example, the driver's weight as measured at a seat is an important parameter for the passive safety DAS in order adapt air bags to different passengers (e.g., children). The current temperature inside and outside the car, which is captured by the climate control subsystem, is another example for a real-time entity that is significant beyond the interior DAS. Temperature measurements are an essential input for physical models of sensors in other DASs (e.g., powertrain DAS) and permit to improve the precision and plausibility of sensory information.

Adversely, other DASs need to be able to control body electronics in the interior DAS. In hazardous situations, e.g., after the detection of a potential crash as indicated by yaw rate and lateral acceleration sensors (e.g., during skidding and emergency braking), the vehicle dynamics DAS causes the tensioning of seat-belts, realigning of seats to a safer positions.

4.4. Physical System Structure

The mapping from the functional to the physical system structure needs to assign jobs to ECUs and virtual networks to the time-triggered physical core network. As exemplified in Figure 6, each ECU can host multiple jobs of different DASs. For instance consider the ECU located in the left front of the car. On this ECU one of the three jobs comprising the steering Triple Modular Redundancy (TMR) system and the job providing the braking service of the left front tire are located. Furthermore, the job controlling adaptive forward lighting and the camera system of the left side are scheduled on this component. In order to tolerate arbitrary single component failures it is mandatory to devise a mapping of jobs to ECUs in a way, that redundant jobs are not hosted on the same ECU.

Similarly, the physical core network is the basis for multiple virtual networks. In order to achieve the de-

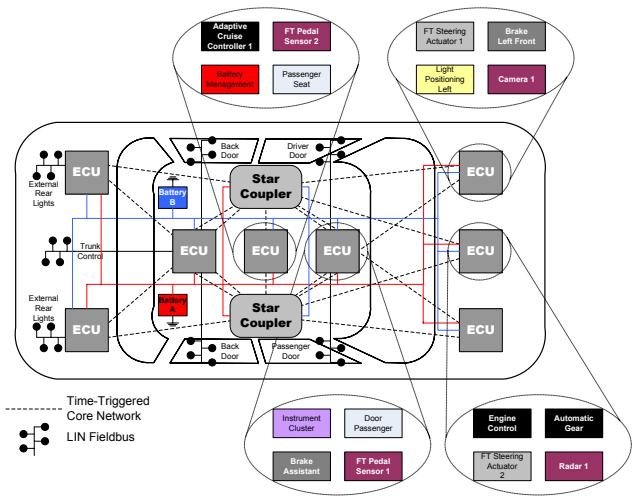

Figure 6. Physical System Structure

pendability requirements of safety-critical DASs, the physical core network relies on two central guardians [2] for achieving fault isolation even in case of arbitrary ECU failure modes. Fault injection experiments have shown that for ultra-dependable applications restrictions concerning the failure modes of ECUs are unjustified [1].

Another prerequisite for safety-critical applications is the accumulation and generation based on dual battery solutions. A redundant power supply scheme provides the foundation for fault-tolerant energy supply of safety-critical by-wire functionality also in case of a failure of one power supply network.

In addition, Figure 6 shows the role of gateways in the physical structure of a car. Physical LIN gateways connect physical LIN clusters to the integrated system, e.g., the LIN fieldbus within the doors of the cars are connected to the ECUs located in the center of the car.

5. Discussion

This section summarizes the main benefits of the introduced integrated automotive architecture and highlights the major design decisions. We will show that the proposed architecture is inline with prevalent architectural trends, such as reuse of functionality across different car segments and introduction of safety-critical applications.

5.1. Economic Benefits

Present day automotive systems follow a federated philosophy with different types of automotive networks [15]. The multitude of communication protocols is a result of the different requirements with respect to functionality, dependability, and performance of the automotive DASs, such as the infotainment DAS, the comfort DAS, or the powertrain DAS.

In contrast to these federated architectures, the integrated architecture for future automotive systems presented in this paper allows to share network and component resources among different DASs in order to evolve beyond a "1 Function – 1 ECU" strategy and achieve a significant reduction in the overall number of ECUs. With an average cost of 40 Euro per ECU in today's cars and 0.2 Euro per wire, the total hardware cost of a federated automotive system with 40 ECUs and 800 wires is about 1800 Euro. If we assume that the number of ECUs will be reduced to 30 nodes and the number of wires to 500 in the integrated system, with an increased average cost per ECU of 50 Euro, then total hardware cost is reduced by 200 Euro per car.

5.2. Complexity

A minimal complexity, i.e. a minimal mental effort to analyze and understand a system, is one of the most important design drivers of the presented integrated architecture. Today, the unmanaged complexity of systems is the major cause of design faults. *Complexity increases the likelihood of serious, yet latent, design flaws* [9, p. 39]. In [16, p. 131] it is stated that *complexity of software and hardware causes a non linear increase in human-error induced design faults.* High system complexity also prolongs development, which is detrimental to a company's economic success, because today's business realities demand a short time-to-market. In addition, complexity complicates validation and certification as state-of-the-art formal verification tools are limited in the size of a design that they can handle.

In order to manage complexity, the presented integrated architecture enables designers to proceed in such a way as if they were realizing DASs in a federated system. A DAS along with the corresponding communication resources (virtual networks) and computational resources (partitions in components) is encapsulated and interactions between DASs are limited to the exchange of messages via precisely specified hidden gateways. Similar to a federated architecture, the integrated architecture decouples each DAS from other DASs. Each DAS possesses a dedicated encapsulated virtual network. A virtual network is private for a DAS, i.e. other DASs cannot perceive or effect the exchanged messages other than those being explicitly exported via a gateway. By exercising this strict control over the interactions between DASs, only the behavior of the DAS's virtual network and the behavior of gateways is of relevance when reasoning about a DAS. The message transmissions on other virtual networks can be

abstracted from. Similarly, each job executes in a corresponding partition, which forms an encapsulated execution environment with guaranteed component and network resources. The activities of jobs executing on the same component cannot affect the component and network resource that are available to other jobs in the component.

By not only supporting error containment between DASs, but error containment between jobs within a DAS, the integrated architecture exceeds the error containment capabilities of most federated systems. Furthermore, the integrated architecture offers generic architectural services as a base line for application development, thus decreasing the functionality that must be realized at the application level. Such a slimmer application is easier to comprehend and leaves less room for software design faults. Only the interface between the architecture and the application is visible to the application developer, while the realization of the architectural services remains hidden.

5.3. Dependability

Carmakers are on the verve of deploying by-wire technology to improve the functionality of the vehicle that goes beyond traditional hydraulic and mechanic systems. This trend requires the deployed E/E architectures to meet the requirement for ultra-high dependability [27] (a maximum failure rate of 10^{-9} critical failures per hour is demanded). Today's technology does not support the manufacturing of electronic devices with failure rates low enough to meet these reliability requirements. Since ECU failure rates are in the order of 10^{-5} to 10^{-6}, ultra-dependable applications require the system as a whole to be more reliable than any one of its ECUs. This can only be achieved by utilizing fault-tolerant strategies that enable the continued operation of the system in the presence of ECU failures [3].

For this reason, the integrated automotive architecture is based on a time-triggered base architecture with a fault-tolerant and deterministic communication system. The consistent distributed state with respect to a global sparse time base is the basis for active redundancy, such as TMR. Replicated star couplers use the a priori knowledge about the points in time of communication activities to contain timing message failures, thus offering fault isolation even with arbitrary ECU failure modes [1]. In addition, the proposed architecture offers redundancy of the power nets, supported by electrical energy management techniques for the generation, the distribution, and the accumulation of power.

Furthermore, by decreasing the number of ECUs and physical networks, the integrated automotive architecture offers increased reliability by minimizing the number of connectors and wires. Field data has shown that a significant amount of electrical failures are attributed to connector problems.

5.4. Flexibility

A newly developed E/E architectures is expected to be deployable on different car segments and models. Consequently, the reuse of applications in a modular way is of critical importance. This issue also has an impact on "Tier 1 system suppliers" that need to adapt their subsystem design according to this trend. The usage of validated and possibly certified application software in combination with standard physical ECUs on different models and segments of vehicles will lead to advantages in terms of increased volume, multi-supplier management, and multi-platform management. In addition, not only software modules but complete electronic subsystems can be made available for different vehicle platforms. This strategy also eases the design choices for the interior of the car, due to the freedom of decoupling application software from a particular physical node. The integrated automotive architecture provides a high degree of freedom in the allocation of jobs to ECUs and supports the migration of jobs between ECUs (constrained by dependability requirements).

6. Conclusion

Future car generations require computer architectures to accommodate the need for mixed criticality applications, i.e. supporting applications with ultra-high dependability requirements as well as applications where flexibility and resource efficiency is of primary concern (e.g., comfort electronics). The introduced DECOS architecture establishes such an infrastructure and also enables physical integration by combining multiple DASs and virtual networks within a single distributed real-time computer system. Thus, the architecture reduces the number of different networks and protocols.

The proposed integrated architecture exhibits flexibility and supports reuse of application software across different car segments. The key element for this flexibility, as well as for complexity management and the independent development of subsystems, are small DASs. Instead of the typical domain oriented system structure, we show that we can subdivide the overall functionality of a car into smaller DASs, each equipped with dedicated architectural services. By transforming a today's automotive system onto the future E/E architecture, we have demonstrated the feasibility of the integrated architecture for a future automotive system.

Acknowledgments

This work has been supported by the European IST project DECOS under project No. IST-511764.

References

[1] A. Ademaj, H. Sivencrona, G. Bauer, and J. Torin. Evaluation of fault handling of the time-triggered architecture with bus and star topology. In *Proceedings of the 2003 International Conference on Dependable Systems and Networks*, pages 123–132, June 2003.

[2] G. Bauer, H. Kopetz, and W. Steiner. The central guardian approach to enforce fault isolation in a time-triggered system. In *Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (ISADS 2003)*, pages 37–44, Pisa, Italy, Apr. 2003.

[3] R. Butler, J.L.Caldwell, and B. Vito. Design strategy for a formally verified reliable computing platform. In *Proceedings of the 6th Annual Conference on Computer Assurance (COMPASS) Systems*, pages 125–133, Gaithersburg, MD, USA, June 1991. NASA Langley Res. Center.

[4] C. Jones et al. Final version of the DSoS conceptual model. *DSoS Project (IST-1999-11585)*, Dec. 2002.

[5] W. Elmenreich. *Sensor Fusion in Time-Triggered Systems*. PhD thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/3/182-1, 1040 Vienna, Austria, 2002.

[6] P. Giusto, A. Ferrari, L. Lavagno, J.-Y. Brunel, E. Fourgeau, and A. Sangiovanni-Vincentelli. Automotive virtual integration platforms: why's, what's, and how's. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 370–378, Sept. 2002.

[7] R. Hammett. Flight-critical distributed systems: design considerations [avionics]. *IEEE Aerospace and Electronic Systems Magazine*, 18(6):30–36, June 2003.

[8] H. Heitzer. Development of a fault-tolerant steer-by-wire steering system. *Auto Technology*, 4:56–60, Apr. 2003.

[9] S. Johnson and R. Butler. Design for validation. *IEEE Aerospace and Electronic Systems Magazine*, 7(1):38–43, Jan. 1992.

[10] H. Kopetz. *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[11] H. Kopetz and G. Bauer. The time-triggered architecture. *IEEE Special Issue on Modeling and Design of Embedded Software*, Jan. 2003.

[12] H. Kopetz and R. Obermaisser. Temporal composability. *Computing & Control Engineering Journal*, 13:156–162, Aug. 2002.

[13] H. Kopetz, R. Obermaisser, P. Peti, and N. Suri. From a federated to an integrated architecture for dependable embedded real-time systems. Technical Report 22, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2004.

[14] H. Kopetz and N. Suri. Compositional design of RT systems: A conceptual basis for specification of linking interfaces. In *Proceedings of the 6th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 51–60, May 2003.

[15] G. Leen and D. Heffernan. Expanding automotive electronic systems. *Computer*, 35(1):88–93, Jan. 2002.

[16] N. Leveson. Software safety: why, what, and how. *ACM Comput. Surv.*, 18(2):125–163, 1986.

[17] LIN Consortium. *LIN Specification Package Revision 2.0*, Sept. 2003.

[18] R. Obermaisser, P. Peti, and H. Kopetz. Virtual gateways in the decos integrated architecture. In *Proceedings of the Workshop on Parallel and Distributed Real-Time Systems 2005 (WPDRTS)*. IEEE, Apr. 2005.

[19] R. Obermaisser, P. Peti, and H. Kopetz. Virtual networks in an integrated time-triggered architecture. In *Proceedings of the Tenth IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS2005)*, Feb. 2005.

[20] G. Reichart and M. Haneberg. Key drivers for a future system architecture in vehicles. In *Convergence International Congress*, Detroit, MI, USA, October 2004. SAE.

[21] Robert Bosch Gmbh, Stuttgart, Germany. *CAN Specification, Version 2.0*, 1991.

[22] J. Rushby. Partitioning for avionics architectures: Requirements, mechanisms, and assurance. NASA Contractor Report CR-1999-209347, NASA Langley Research Center, June 1999. Also to be issued by the FAA.

[23] J. Rushby. A comparison of bus architectures for safety-critical embedded systems. Technical report, Computer Science Laboratory, SRI International, Sept. 2001.

[24] A. Saad and U. Weinmann. Intelligent automotive system services: Requirements, architectures and implementation issues. In *Convergence International Congress*, Detroit, MI, USA, October 2004. SAE.

[25] A. Sangiovanni-Vincentelli. Defining platform-based design. *EEDesign of EETimes*, February 2002.

[26] H. Simon. *The Sciences of the Artificial*. MIT Press, 1996.

[27] N. Suri, C. Walter, and M. Hugue. *Advances In Ultra-Dependable Distributed Systems*, chapter 1. IEEE Computer Society Press, 10662 Los Vaqueros Circle, P.O. Box 3014, Los Alamitos, CA 90720-1264, 1995.

[28] J. Swingler and J. McBride. The degradation of road tested automotive connectors. In *Proceedings of the 45th IEEE Holm Conference on Electrical Contacts*, pages 146–152, Pittsburgh, PA, USA, Oct. 1999. Dept. of Mech. Eng., Southampton Univ.