Experimental Platform for Innovative ICT Car Architecture

Gerd Kainz, Christian Buckl and Alois Knoll

***Abstract*—The ever rising complexity of the historically grown architecture of nowadays cars leads to massive problems concerning reliability, costs and development time. The observable trend to electric cars offers the opportunity to redesign the information and communication technologies (ICT) infrastructure of cars to encounter these problems. Within this paper, we present an evaluation platform for innovative ICT car architectures. This platform can be used to evaluate different concepts, such as the usage of smart sensors or the integration of several software functions on one controller using virtualization, their applicability and implications for the overall ICT architecture.**

***Index Terms*—automotive embedded systems, embedded systems, architecture standards.**

I. Motivation

Currently great efforts are taken to move from ordinary cars with combustion engine towards fully electric cars. The standard approach by the automotive industry is a simple replacement of the combustion engine and gasoline tank by an electric engine and batteries. The results are comparably small changes to the architecture of modern cars. All the mechanical safety mechanisms are working as before and ensure necessary functionality in situation of system blackouts. On the other side this type of cars cannot fully utilize the high potential of the electric engines, which makes it possible to use separate motors for each axle or even for each wheel. Depending on the selected variant the motors can be placed either at a central position, near the wheels or at the wheels. These new options can help to get rid of a lot of classical mechanical components like steering rod, differential, axle, etc. and leads to reduction of weight and space. On the other hand, many mechanisms that were previously implemented by mechanics, such as synchronization of the wheels, must now be implemented by software leading to high safety requirements. To accomplish these new requirements, the car's information and communication technologies (ICT) architecture has to change and adapt to the new needs. Because of the shift from functions implemented by hardware to functions implemented in software, the software architecture will change considerably. This change offers the opportunity to redesign the whole ICT architecture that has been developed incrementally over the years without a specific blue print. Reliability and safety are central points for this new ICT architecture, as well as simplicity and flexibility. Tool integration and automation are also a major interest.

One major goal will be to reduce the complexity of the ICT architecture and to reduce the number of electronic control units (ECUs). Nowadays the on-board network consists of up to 70 - 100 ECUs [7] strongly connected with each other to fulfill the complex control tasks. All the ECUs are constructed and programmed by different suppliers and integrated by the car manufacturer (original equipment manufacturer - OEM). For the communication of the ECUs, many different communication networks are used. CAN (Controller Area Network) is the most widely used technology for control tasks. LIN (Local Interconnect Network) realizes a cheap way to communicate with small smart sensors and actuators. In the domain of infotainment the MOST (Media Oriented Systems Transport) standard is used. FlexRay is a time triggered bus introduced for highly reliable and high-bandwidth communication. Based on the used communication technology the introduction of new network communication can lead to unexpected changes in the behavior of the functions also using the same communication resources. The huge heterogeneity and complexity leads to additional effort to ensure the correct function of the overall system after new components are introduced. Furthermore sensors or functionalities are often duplicated not for reasons of reliability or safety, but due to the complex architecture and the non-accessibility of sensor data due to black-box ECUs.

G. Kainz and C. Buckl are with the Department of Cyber-Physical Systems, fortiss - Munich Software and Systems Institute, 80805 Munich, Germany (e-mail: {kainz, buckl}@fortiss.org)

A. Knoll is with the Department of Robotics and Embedded Systems, Institut für Informatik, Technische Universität München, 85748 Garching, Germany
(e-mail: knoll@in.tum.de)

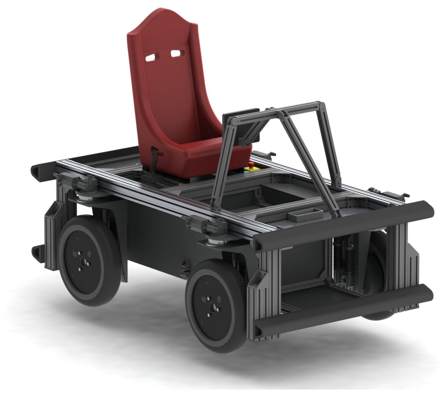

Fig. 1. Design of the Evaluation Platform

Within this paper, we present an evaluation platform for innovative ICT infrastructures. This platform can be used evaluate different concepts, such as the usage of smart sensors or the integration of several software functions on one controller using virtualization, their applicability and implications for the overall ICT architecture. The hardware setup of this evaluation platform is described in section II. Section III focuses on the research objectives concerning the ICT infrastructure. Afterwards the proposed software architecture and its corresponding tooling are explained in section IV. The conclusion consists of an overview about the current development state and an outlook on future work.

II. Description of Hardware Setup

To demonstrate the applicability of possible ICT infrastructures, the most challenging hardware setup has been selected: a car with four independently controllable driving / steering units interconnected only by a network. The hardware setup of the evaluation platform called 'eCar' is based on the eCorner concept from Siemens VDO [4]. It consists of four eCorner modules, which can be controlled independently of each other. Each of the eCorner modules is composed of a drive and a steering motor and weighs around 50 kg. As the eCar does not have any mechanical brakes, the braking is realized via the drive motors. No mechanical axes are used for the synchronization of the eCorner modules. Instead, the whole control system is based on a distributed system (X-by-Wire). No mechanical fallback solutions

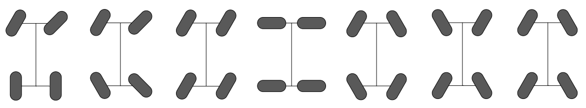

Fig. 2. Different drive modes of the evaluation platform (two-wheel steering, four-wheel steering, vector, park, turn on place, parking brake and emergency brake mode)

were integrated in this car, so that the whole reliability and safety must be ensured by software. Figure 1 shows the eCar evaluation platform.

The drive power of the eCar is 8 kW (4 x 2 kW). Each of the wheels has a maximum torque of 160 Nm. The energy comes from four truck batteries, which are used to generate 48 V for driving, 24 V for steering and 24 V, 12 V and 5 V for the ECUs. The car is controlled via side stick. The current state of the evaluation platform is presented on a 10" touch screen, which can also be used to change between different drive modes. The outline of the car is around 2.25 x 1.25 x 1.75 m (LxWxH) and its weight is about 600 kg. The eCar is constructed to carry one passenger with a maximum speed limited to 50 km/h.

Due to the fact that all eCorner modules can be controlled independently it is possible to realize different drive modes. Additional to the conventional two-wheel steering a four-wheel steering can be implemented. But also totally unconventional drive modes like a vector mode (all four wheels are moving parallel to each other), parking mode (used to get sideways into a parking lot), turn on place mode (rotating around the center of the car), parking brake mode (used to fix the car on place without mechanical brakes) and an emergency brake mode (like snowplow in skiing). The different drive modes are illustrated in figure 2.

The modular and open construction of the evaluation platform makes it possible to easily attach or detach sensors and ECUs as needed. In addition, different network topologies can be easily integrated into the car. Hence it is suitable for the evaluation of different ICT architectures and can be used to compare them with each other.

III. Objectives

The goal of this project is to develop an ICT architecture and related development methodology to mitigate the current problems arising from the current ICT infrastructure. In the following, the major concepts used to reach this goal are explained shortly:

- **Layered architecture:** To minimize the dependencies and interconnections between the different software components, a layered architecture should be introduced. The robotic domain offers adequate blue prints for such an architecture consisting of at least an hardware abstraction layer, a layer to implement low-level functionality (smart sensors / actuators) and a layer to implement high-level functionality (e.g. driver assistance functionality).
- **Data-centric approach:** To enable the reuse of existing (sensor) data, for all components the developers should specify the required and provided data. The specification must be based on strong data types and consist of information with respect to timing (e.g. required / provided data rates), accuracy, and quality-of-service (QoS). Based on this specification, development tools could automate the configuration of the communication and the scheduling of the software components.
- **Hardware abstraction:** In current automotive systems, most functions are executed on dedicated ECUs (one-to-one mapping). In future ICT architectures, the software should be developed independently of the concrete hardware. The mapping of software functions to controllers should be based on the requirements and allow also the mapping of several functions to one ECU similar to the approaches in the avionic domain - integrated modular avionics (IMA) architecture.
- **Simplified network structure:** Instead of using a large number of different communication standards, the network structure should be simplified and unified. Adequate communication protocols supporting both reliable and unreliable, event- and time-triggered communication, such as FlexRay, should be used.
- **Fault Recognition and Redundancy Management:** An adequate concept for fault recognition and redundancy management taking into account the cost constraints of the automotive must be developed to satisfy constraints from safety standards, such as ISO 26262 [3]. This concept should be based on a separation of fault-tolerance mechanisms and application functionality [6] to simplify the development.
- **Tool support:** The development methodology should be based on development tools for an automatic and flexible mapping from logical architecture to target hardware. The tool support in combination with the layered architecture and the hardware abstraction should enable scalability and portability of the approach across vehicle classes.

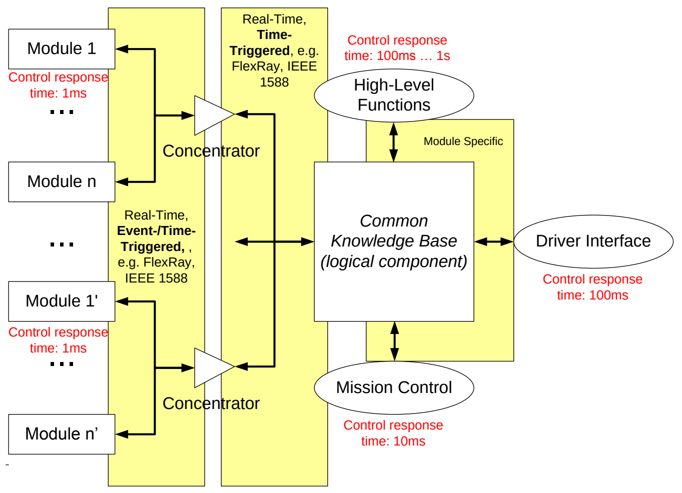
Module 1
Control response time: 1ms
Module n
Module 1'
Control response time: 1ms
Module n'
Real-Time, Event-/Time-Triggered, , e.g. FlexRay, IEEE 1588
Concentrator
Concentrator
Real-Time, Time-Triggered, e.g. FlexRay, IEEE 1588
Control response time: 100ms … 1s
High-Level Functions
Module Specific
Common Knowledge Base (logical component)
Driver Interface
Control response time: 100ms
Mission Control
Control response time: 10ms

Fig. 3. Proposed Software Architecture of Electrical Cars

IV. ICT Architecture & Development Process

The proposed ICT architecture for electrical cars, see Figure 3, is inspired by robotics and builds upon a layered software architecture. The bottom of the layered architecture consists of sensors and actuators and implements the interaction with the physical environment. Above the sensors and actuators is a hardware abstraction layer, which introduces a standardized interface for sensors and actuators / motors of the same type. By this method, sensors and actuators can be easily replaced without having to change the software. On top of the hardware abstraction layer, software components enhance the functionality of the sensors / actuators to provide a certain level of intelligence (smart sensors / actuators). This intelligence can for example abstract control loops to control an eCorner module based on the control commands coming from upper layers. The communication between the smart sensors / actuators and the upper layers is realized using a real-time network.

Instead of creating for the data dependencies between the different modules a communication relation by hand, the communication relationship shall be automatically established by the system. This can be either done statically at development time or dynamically at runtime. The concept behind this idea is a common knowledge repository, which is responsible for collecting all the available data and offering all the components access to available data and take care of the correct data alignment. It is important to note that this component is a logical component for most of the communication in the car, since the majority of the communication will be statically defined. In this case, the communication is directly performed between the components, however the specification and configuration of the required messages are automated by according development tools. Since for all components, only the required data, but not the specific sensor is specified, unnecessary redundant sensors can be avoided. In the context of the automatic network configuration a correct transmission with respect to timing requirements can also be verified. Hence, the approach establishes a *Single-System Illusion* in the sense that the developer does not have to take care whether the data is produced locally or on a remote ECU.

To structure the network and to minimize the required bandwidth, a concentrator architecture is suggested. This architecture forms sub networks in which high amount of data can be shared in a fast way. Hence control loops with high requirements regarding response time can be implemented locally. The data visible beyond sub network structures is restricted by a so called 'concentrator', which makes all necessary information visible to the outside world and hides the rest. High-level functions can be realized at upper levels. Here, the control response times might be more relaxed.

In order to keep the complexity of the network as simple as possible the number of different communication technologies shall be as small as possible. Buses which are able to handle event- and time-triggered communication are best suited. Their properties allow to give tighter bounds on the maximum latency for the communication and to ensure by construction that a faulty component cannot disturb the whole communication on a bus. Remaining bandwidth can additionally be used for event-triggered communication, which normally constitutes best effort services. When an underlying time-triggered bus is additionally synchronized with the above running application, the response time of functions can further be improved.

Naturally the software architecture needs to support fault tolerance. The redundancy management shall be done semi automatically and be based on a separation of fault-tolerance mechanisms and application functionality. The fault-tolerance mechanism can thus be implemented within the run-time system and development tools can select and generate appropriate mechanisms based on a mechanism specification done by the developer. To keep the number of additionally needed ECUs low, the integration of different functions on the same ECU shall be supported by the software architecture. High reliability and safety can be ensured by executing the functions in a time and space separated way. Figure 3 illustrates a possible setup for redundancy.

Through providing a helpful integrated continuous tool chain the development of car software is speeded up. Thereby it is useful to use model driven software development (MDSD) tools with code generation to automotive standards, like AUTOSAR [1].

V. Stage of Development

In this section, we will describe the current development stage of the project. The current version of the eCar consists of all necessary components to realize a human controlled driving via sidestick and touch screen. Therefore all mechanical and electrical components had to be put together and brought into service. The current architecture is shown in figure 5. It consists of distributed heterogeneous hardware components, which have to communicate with each other to realize the functionality of the car. Basic components are Luminary Micro LM3S8962 boards [8] with an ARM Cortex M3 operating at a clock

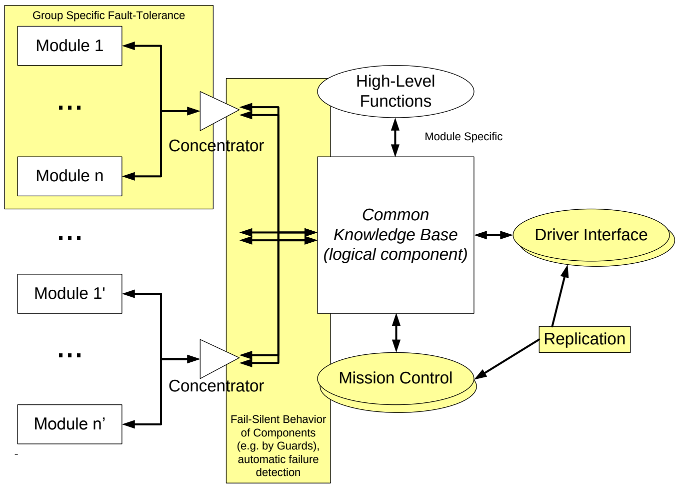
Group Specific Fault-Tolerance
Module 1
...
Module n
Concentrator
...
Module 1'
...
Module n'
Concentrator
Fail-Silent Behavior of Components (e.g. by Guards), automatic failure detection
High-Level Functions
Module Specific
Common Knowledge Base (logical component)
Driver Interface
Replication
Mission Control

Fig. 4. Redundancy Management

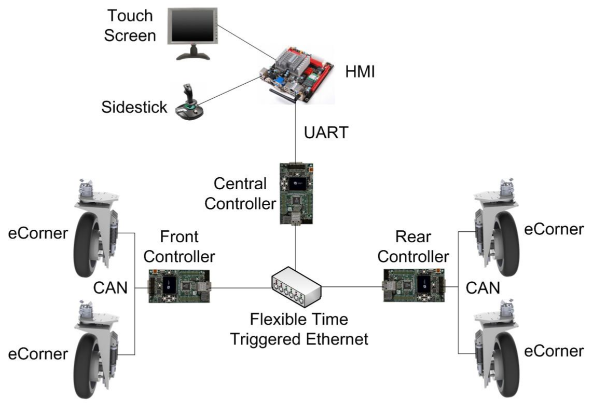

Fig. 5. Current Architecture of the eCar

rate of 50 Mhz. FreeRTOS [2] is used as real-time operating system on the Luminary Micro boards. For communication, an own real-time Ethernet protocol, the Flexible Time-Triggered Ethernet (FTTE) protocol, based on the standard IEEE 1588 [5] was developed. It supports similar to FlexRay time- and event-triggered communication, but can be extended to allow also dynamic slot allocation (similar to FlexRay v3.0). Furthermore, it operates on a higher speed (currently 100 Mb/s).

Two controllers, one at the rear and one at the front axis, are used to implement the smart sensor / actuator layer with respect to the eCorners. Since no power electronics with FlexRay or Ethernet interface were available, a CAN bus is used to control the individual controllers. A third controller based on an Intel Atom Dual Core CPU is used to implement the smart sensor / actuator layer with respect to the HMI. The control/decision layer is implemented on a Luminary controller.

Currently the experimental platform is able to respond to the driver input. Thereby the driver can select between four-wheel steering, vector, parking, turn on place, hand brake and emergency brake mode. According to the actual drive mode the user input is handled in different ways. For example the turn on place mode only reacts to left and side movements of the sidestick and translates them in the related clockwise direction.

VI. Future Work

After successfully bringing the experimental platform eCar into service, there are a lot of open points for future research. The most important point is to establish fault tolerance mechanisms to ensure a reliable and safe operation of the car even when errors occur. An effective error handling strategy has to include a redundancy management to cover blackout of various ECUs during operation. To relax the effort of developing new or alternative modules for the experimental platform a model driven development tool shall be developed, which enables the programmer to specify the system requirements of modules on a high-level of abstraction. The models can then be used to preconfigure the system according to the needs of the modules. They can also be used to generate most of the code for the run-time system. By creating a model driven development tool the creation of new function can be simplified. Also a useful improvement is a diagnosis and configuration tool, which helps to flash the new versions of the system over the network to the corresponding ECU and is also able to query liveness information during system operation from the ECU in the distributed system.

Also the hardware setup needs some further improvements. One important requirement is related to the battery system. Currently lead-acid batteries are used to store the required energy for the operation of the car. Due to the limited charging capabilities, the braking force is currently limited. A switch to lithium-ion batteries will solve this problem. Additional sensors can be integrated to inspect the battery system during operation for early recognition of critical battery states.

Furthermore additional sensors are going to be integrated to implement established driver assistance functions and automotive safety functions. Intended features are for example automatic parking. The driver interface still constitutes an open research topic. It is currently not clear if a sidestick is the best choice to control the vehicle in all situations. On the other side a normal steering wheel cannot be used for control in all the different implemented driving modes. A mixture of both concepts (side stick and steering wheel) can be an interesting solution. Finally, the platform will also be used to investigate on different concepts for car-to-x communication.

Acknowledgment

The authors would like to thank the group Embedded Systems and Robotics at Technische Universität München for their support during the mechanical construction of the platform.

References

[1] AUTomotive Open System ARchitecture (AUTOSAR) Release 4.0.
[2] *FreeRTOS, www.freertos.org*.
[3] ISO 26262: Road vehicles - Functional safety.
[4] Siemens VDO Making a Case for In-Wheel Systems: the eCorner Project. *Green Car Congress*, Sep 2006.
[5] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages 1–269, 2008.
[6] C. Buckl. *Model-based development of fault-tolerant real-time systems*. PhD thesis, Technische Universität München, 2008.
[7] R. N. Charette. This Car Runs on Code. *IEEE Spectrum*, Green Tech, Advanced Cars, Feb 2009.
[8] Luminary Micro. *LM3S8962 Microcontroller*, Mar 2008.

Gerd Kainz received his master degree (Diplom-Informatiker (Univ.)) in computer science major and electrical engineering minor from the Technische Universität München, Germany, in 2009. Since April 2009 he is a PhD student at fortiss – Munich Software and Systems Institute. His research interests are model-driven tools, middleware, communication networks and embedded systems.

Dr. Christian Buckl received his master degree (Diplom-Informatiker (Univ.)) in computer science major in 2004, and his Ph.D. degree in 2009 both from the Technische Universität München, Germany. Since March 2009 he is leading the group Cyber-Physical Systems in the newly founded research institute fortiss. His research include fault-tolerant system design, real-time capable communication protocols and model-driven development tools for embedded systems.

Prof. Dr. Alois Knoll is a full professor at the Computer Science Department of the Technische Universität München (TUM) since 2001. His group Robotics and Embedded Systems at TUM investigates on cognitive, medical and sensor-based robotics, as well as embedded systems. In these fields he has published over 200 technical papers and guest-edited international journals. Since 2009, he is director of the research institute fortiss.