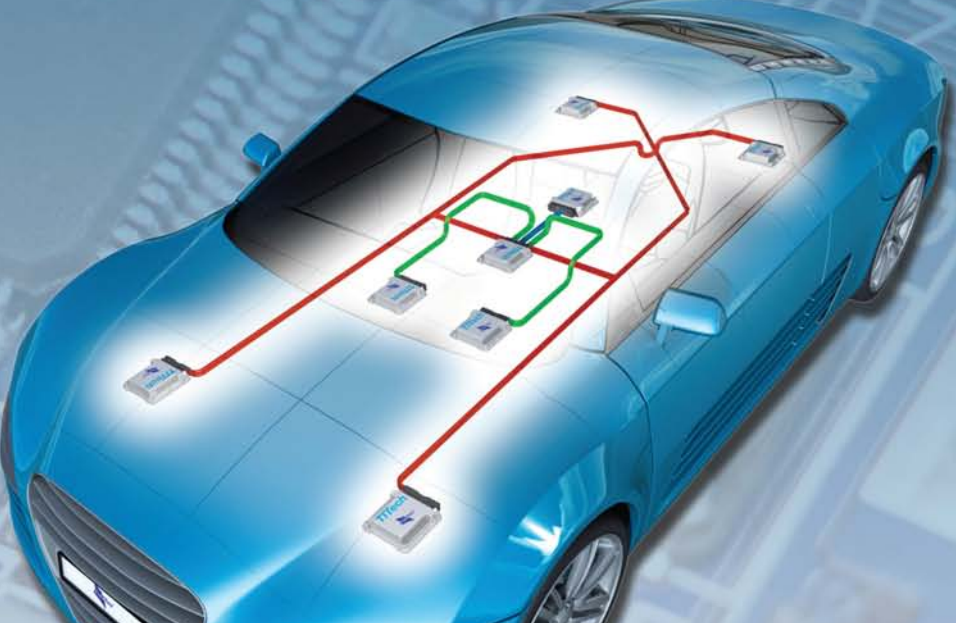
INDUSTRIAL INFORMATION TECHNOLOGY SERIES

Automotive Embedded Systems Handbook



Edited by
Nicolas Navet
Françoise Simonot-Lion

CRC Press
Taylor & Francis Group

Automotive Embedded Systems Handbook

INDUSTRIAL INFORMATION TECHNOLOGY SERIES

Automotive Embedded Systems Handbook

Edited by
Nicolas Navet
Françoise Simonot-Lion

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

International Standard Book Number-13: 978-0-8493-8026-6 (Hardcover)

Library of Congress Cataloging-in-Publication Data

Automotive embedded systems handbook / edited by Nicolas Navet and
Francoise Simonot-Lion.
p. cm. -- (Industrial information technology ; 5)
Includes bibliographical references and index.
ISBN-13: 978-0-8493-8026-6
ISBN-10: 0-8493-8026-X
1. Automotive computers. 2. Automobiles--Electronic equipment. 3. Automobiles--Automatic control--Equipment and supplies. 4. Embedded computer systems. I. Navet, Nicolas. II. Simonot-Lion, Francoise. III. Title. IV. Series.

TL272.53.A9868 2009
629.2--dc22 2008024406

Contents

Preface

The objective of the *Automotive Embedded Systems Handbook* is to provide a comprehensive overview about existing and future automotive electronic systems. The distinctive features of the automotive world in terms of requirements, technologies, and business models are highlighted and state-of-the-art methodological and technical solutions are presented in the following areas:

- In-vehicle architectures
- Multipartner development processes (subsystem integration, product line management, etc.)
- Software engineering methods
- Embedded communications
- Safety and dependability assessment: validation, verification, and testing

The book is aimed primarily at automotive engineering professionals, as it can serve as a reference for technical matters outside their field of expertise and at practicing or studying engineers, in general. On the other hand, it also targets research scientists, PhD students, and MSc students from the academia as it provides them with a comprehensive introduction to the field and to the main scientific challenges in this domain.

Over the last 10 years, there has been an exponential increase in the number of computer-based functions embedded in vehicles. Development processes, techniques, and tools have changed to accommodate that evolution. A whole range of electronic functions, such as navigation, adaptive control, traffic information, traction control, stabilization control, and active safety systems, are implemented in today's vehicles. Many of these new functions are not stand-alone in the sense that they need to exchange information—and sometimes with stringent time constraints—with other functions. For example, the vehicle speed estimated by the engine controller or by wheel rotation sensors needs to be known in order to adapt the steering effort, to control the suspension, or simply to choose the right wiper speed. The complexity of the embedded architecture is continually increasing. Today, up to 2500 signals (i.e., elementary information such as the speed of the vehicle) are exchanged through up to 70 electronic control units (ECUs) on five different types of networks.

One of the main challenges of the automotive industry is to come up with methods and tools to facilitate the integration of different electronic subsystems coming from various suppliers into the vehicle's global electronic architecture. In the last 10 years, several industry-wide projects have been undertaken in that direction (AEE*, EAST, AUTOSAR, OSEK/VDX, etc.) and significant results have already been achieved (e.g., standard components such as operating systems, networks and middleware, "good practices," etc.). The next step is to build an accepted open software architecture, as well as the associated development processes and tools, which should allow for easily integrating the different functions and ECUs provided by carmakers and third-part suppliers. This is ongoing work in the context of the AUTOSAR project.

As all the functions embedded in cars do not have the same performance or safety needs, different qualities of service are expected from the different subsystems. Typically, an in-car embedded system is divided into several functional domains that correspond to different features and constraints. Two of them are concerned specifically with real-time control and safety in the vehicle's behavior: the "power train" (i.e., control of engine and transmission) and the "chassis" (i.e., control of suspension, steering, and braking) domains. For these safety-critical domains, the technical solutions must ensure that the system is dependable (i.e., able to deliver a service that can be justifiably trusted) while being cost-effective at the same time.

These technical problems are very challenging, in particular due to the introduction of X-by-wire functions, which replace the mechanical or hydraulic systems, such as braking or steering, with electronic systems. Design paradigms (time-triggered, "safety by construction"), communication networks (FlexRay, TTP/C), and middleware layers (AUTOSAR COM) are currently being actively developed in order to address these needs for dependability.

The principal players in the automotive industry can be divided into:

- Vehicle manufacturers
- Automotive third-part suppliers
- Tool and embedded software suppliers

The relationships between them are very complex. For instance, suppliers providing key technologies are sometimes in a very strong position and may impose their technical approach on carmakers. Since the competition is fierce among carmakers and suppliers, keeping the company's know-how confidential is crucial. This has strong implications in the technical field. For instance, the validation of the system (i.e., verifying that the system meets its constraints) may have to be carried out

* Architecture Electronique Embarquée (AEE, 1997–2000) is a French project supported by the Ministry of Industry with PSA and Renault, Sagem, Siemens, and Valeo as main industrial partners. Its main objective was to find solutions for easing the portability of applicative level software. Embedded Electronic Architecture (EAST-EEA, 2001–2004, see http://www.east-eea.net/) is an European ITEA project involving most major European carmakers, automotive third-part suppliers, tools and middleware suppliers, and research institutes. Automotive Open Architecture (AUTOSAR, 2004–2007, see http://www.autosar.org) is an ongoing follow-up to EAST-EAA aimed at establishing open standards for automotive embedded architecture. Open systems and the corresponding interfaces for automotive electronics (OSEK, see http://www.osek-vdx.org) is a German automotive industry project defining standards for software components used for communication, network management, and operating systems. Some of the outcomes of OSEK (e.g., OSEK/OS) are already widely used in production cars.

with techniques that do not require full knowledge of the design rationales and implementation details.

Shortening the time to market puts on added pressure because carmakers must be able to propose their innovations—that usually rely heavily on electronic systems—within a time frame that allows for these innovations to be really considered as innovative. The players involved strive to reduce the development time while the system's overall complexity increases, demanding even more time. This explains why, despite the economic competition, they have agreed to work together to define standard components and reference architecture that will help cut overall development time.

This book contains 15 contributions, written by leading experts from industry and academia directly involved in the engineering and research activities treated in this book. Many of the contributions are from industry or industrial research establishments at the forefront of the automotive domain: Siemens (Germany), ETAS (Germany), Volvo (Sweden), Elektrobit (Finland), Carmeq (Germany), The MathWorks Inc. (United States), and Audi (Germany). The contributions from academia and research organizations are presented by renowned institutions such as Technical University of Berlin (Germany), LORIA-Nancy University (France), INRIA (France), IRCCyN Nantes University (France), KTH (Sweden), Mälardalen University (Sweden), Kettering University (United States), University of Aveiro (Portugal), and Ulm University (Germany).

Organization

Automotive Architectures

This part provides a broad introduction to automotive embedded systems, their design constraints, and AUTOSAR as the emerging de facto standard. Chapter 1, "Vehicle Functional Domains and Their Requirements," introduces the main functions embedded in a car and how these functions are divided into functional domains (chassis, power train, body, multimedia, safety, and human–machine interfaces). Some introductory words describe the specificities of the development process as well as the requirements in terms of safety, comfort, performance, and cost that need to be taken into account.

In Chapter 2, "Application of the AUTOSAR Standard," the authors tackle the problem of the standardization of in-vehicle embedded electronic architectures. They analyze the current status of software in the automotive industry and present the specifications elaborated within the AUTOSAR consortium in terms of standardization. Particular attention has to be paid to AUTOSAR because it is becoming a standard that everyone has to understand and deal with.

Finally, Chapter 3, "Intelligent Vehicle Technologies," presents the key technologies that have been developed to meet today's, and tomorrow's, automotive challenges in terms of safety, better use of energy, and better use of space, especially in cities. These technologies, such as sophisticated sensors (radar, stereo-vision, etc.), wireless networks, or intelligent driving assistance, will facilitate the conception of partially or fully autonomous vehicles that will reshape the transport landscape and commuters' travel experience in the twenty-first century.

Embedded Communications

The increasing complexity of electronic architectures embedded in a vehicle, and locality constraints for sensors and actuators, has led the automotive industry to adopt a distributed approach for implementing the set of functions. In this context, networks and protocols are of primary importance. They are the key support for integrating functions, reducing the cost and complexity of wiring, and furnishing a means for fault tolerance. Their impact in terms of performance and dependability is crucial as a large amount of data is made available to the embedded functions through the networks. This part includes three chapters dedicated to networks and protocols.

Chapter 4, "A Review of Embedded Automotive Protocols," outlines the main protocols used in automotive systems; it presents the features and functioning schemes of CAN, J1850, FlexRay, TTCAN, and the basic concepts of sensor/actuator networks (LIN, TTP/A) and multimedia networks (MOST, IDB1394). The identification of the communication-related services commonly offered by a middleware layer and an overview of the AUTOSAR proposals conclude the chapter.

CAN is at present the network that is the most widely implemented in vehicles. Nevertheless, despite its efficiency and performance, CAN does not possess all the features that are required for safety-critical applications. The purpose of the chapter, "Dependable Automotive CANs," is to point out CAN's limitations, which reduce dependability, and to present technical solutions to overcome or minimize these limitations. In particular, the authors describe techniques, protocols, and architectures based on CAN that improve the dependability of the original protocol in some aspects while still maintaining a high level of flexibility, namely (Re)CANcentrate, CANELy, FTT-CAN, and FlexCAN.

With the development of technology, there has been an increasing number of functions with strong needs in terms of data bandwidth. In addition, safety requirements have become more and more stringent. To answer to both of these constraints, in 2000, the automotive industry began to develop a new protocol—FlexRay. Chapter 5 "FlexRay Protocol," explains the rationale of FlexRay and gives a comprehensive overview of its features and functioning scheme. Finally, an evaluation of the impact of FlexRay on the development process concludes the chapter.

Embedded Software and Development Processes

The design process of an electronic-embedded system relies on a tight cooperation between car manufacturers and suppliers under a specific concurrent engineering approach. Typically, carmakers provide the specification of the subsystems to suppliers, who are then in charge of the design and realization of these subsystems, including the software and hardware components, and possibly the mechanical or hydraulic parts. The results are furnished to the carmakers, who in turn integrate them into the car and test them. Then comes the "calibration" phase, which consists of tuning control and regulation parameters in order to meet the required performances of the controlled systems. Any error detected during the integration phase leads to costly corrections in the specification or design steps. For this reason, in order to improve the effectiveness of the development process, new design methodologies are emerging, in particular, the concept of a virtual platform, which is now gaining acceptance in the area of the electronic automotive systems design.

The virtual platform concept requires modeling techniques that are suited to the design and validation activities at each step of the development process. In this context, model-based development (MBD) has been extensively studied by both car manufacturers and suppliers. How to adapt this approach to the automotive industry is discussed in Chapter 10, "Model-Based Development of Automotive Embedded Systems." This chapter identifies the benefits of model-based development, explores the state of practice, and looks into the major challenges for the automotive industry.

One of the main issues in automotive systems is to reduce the time to market. The reuse of components, or of subsystems, is one way to achieve this objective. In Chapter 8, "Reuse of Software in Automotive Electronics," the authors give an overview of the challenges faced when reusing software in the automotive industry, the different viewpoints on the reuse issue of manufacturers and suppliers, and the impact of the multipartner development approach.

Sharing the same modeling language between the different parties involved in development is an effective means to ease the cooperative development process. The main purpose of such a language is, on the one hand, to support the description of the system at the different steps of its development (requirement specification, functional specification, design, implementation, tuning, etc.) according to the different points of view and, on the other hand, to ensure a consistency between these different views. Another important aspect is its ability to reflect the structure of the embedded systems as an architecture of components (hardware components, functional components, software components). The ideas and principles brought by architecture description languages (ADLs) are well suited to these objectives. What is an ADL? Why are ADLs needed? What are the main existing ADLs and their associated tools? What are the main ongoing projects in the automotive context? Answers to these questions can be found in Chapter 9 "Automotive Architecture Description Languages."

The introduction and management of product lines is of primary importance for the automotive industry. These product lines are linked to mechanical system variations, and certain customer-visible variations, offered in a new car. The purpose of Chapter 7, "Product Lines in Automotive Electronics" is to present the systematic planning and continuous management of variability throughout the development process. This chapter provides some techniques on how to model the variability as well as traceability guidelines for the different phases of development.

Verification, Testing, and Timing Analysis

Some functions in a car are critical from the safety point of view, such as, for example, certain functions in the chassis or the power train domain. Thus, validation and verification are of primary importance.

Testing is probably the most commonly used verification technique in the automotive industry. A general view on testing approaches is given in Chapter 11 "Testing Automotive Control Software." In particular, this chapter describes current practices and several methods that are involved in the testing activities, such as the classification-tree method, test scenario selection approaches, and black-box/white-box testing processes. As already mentioned, communication networks and protocols are key factors for the dependability and performance of an embedded system. Hence, certain properties on communication architectures have to be verified. Chapter 12, "Testing and Monitoring of FlexRay-Based Applications," deals with the application of testing techniques to the FlexRay protocol. The authors review the constraints in the validation step in the development process of automotive applications and explain how fault-injection and monitoring techniques can be used for testing FlexRay.

As CAN is the most popular network embedded in cars, its evaluation has been the subject of a long line of research. Chapter 13, "Timing Analysis of CAN-Based Automotive Communication Systems," summarizes the main results that have been obtained over the last 15 years in the field of timing analysis on CAN. In particular, it is explained how to calculate bounds on the delays that frames experience before arriving at the receiver end (i.e., the response times of the frames). Accounting for the occurrence of transmission errors, for instance due to electromagnetic interferences, is also covered in this chapter. Due to its medium access control protocol based on the priorities of the frames, CAN possesses good real-time characteristics. However, a shortcoming that becomes increasingly problematic is its limited bandwidth. One solution that is being investigated by car manufacturers is to schedule the messages with offsets, which leads to a desynchronization of the message streams. As shown in Chapter 14, "Scheduling Messages with Offsets on Controller Area Network: A Major Performance Boost," this "traffic shaping" strategy is very beneficial in terms of worst-case response times. The experimental results suggest that sound offset strategies may extend the life span of CAN further, and may defer the introduction of FlexRay and additional CANs.

Chapter 15 "Formal Methods in the Automotive Domain: The Case of TTA," describes the formal verification research done in the context of time-triggered architecture (TTA), and more specifically the work that concerns time-triggered protocol (TTP/C), which is the core underlying communication network of the TTA. These formal verification efforts have focused on crucial algorithms in distributed systems: clock synchronization, group membership algorithm, or the startup algorithm, and have led to strong results in terms of dependability guarantees. To the best of our knowledge, TTA is no longer being considered or implemented in cars. Nevertheless, the experience gained over the years with the formal validation of the TTA will certainly prove to be extremely valuable for other automotive communication protocols such as FlexRay, especially in the perspective that certification procedures will be enforced for automotive systems, as they are now for avionic systems.

We would like to express our gratitude to all of the authors for the time and energy they have devoted to presenting their topic. We are also very grateful to Dr. Richard Zurawski, editor of the Industrial Information Technology Series, for his continuous support and encouragements. Finally, we would like to thank CRC Press for having agreed to publish this book and for their assistance during the editorial process.

We hope that you, the readers of this book, will find it an interesting source of inspiration for your own research or applications, and that it will serve as a reliable, complete, and well-documented source of information for automotive-embedded systems.

Nicolas Navet

Françoise Simonot-Lion

Editors

Nicolas Navet has been a researcher at the Grand Est Research Centre at the National Institute for Research in Computer Science and Control (INRIA), Nancy, France, since 2000. His research interests include real-time scheduling, the design of communication protocols for real-time and fault-tolerant data transmission, and dependability evaluation when transient faults may occur (e.g., EMI). He has authored more than 70 refereed publications and has received the CAN in Automation International Users and Manufacturers Group research award in 1997 as well as five other distinctions (e.g., best paper awards). Since 1996, he has worked on numerous contracts and projects with automotive manufacturers and suppliers. He is the founder and chief scientific officer of RealTime-at-Work, a company dedicated to providing services and software tools that help optimize the hardware resource utilization and verify that dependability constraints are met. He holds a BS in computer science from the University of Berlin, Berlin, Germany and a PhD in computer science from the Institut National Polytechnique de Lorraine, Nancy, France.

Françoise Simonot-Lion is a professor of computer science at University of Nancy, Nancy, France. She has been the scientific leader of the Real Time and InterOperability (TRIO) research team since 1997, which is an INRIA project at the Lorraine Laboratory of Computer Science Research and Applications (LORIA) in Nancy, France. From 2001 to 2004, she was responsible for CARAMELS, a joint research team with PSA Peugeot Citroën funded by the French Ministry for Research and Technology. She has participated in the French Embedded Electronic Architecture project (AEE, 1999–2001), and in the European project ITEA EAST-EEA (2001–2004). The purpose of ITEA EAST was to define an industry-wide layered software architecture, including a communication middleware, and a common architecture description language supporting a formal description of in-vehicle embedded systems (EAST-ADL). She is also an associate editor of *IEEE Transactions on Industrial Informatics*.

Contributors

Luis Almeida
Department of Electronics
Telecommunication and Informatics
University of Aveiro
Aveiro, Portugal

Jakob Axelsson
Volvo Car Corporation
Gothenburg, Sweden

and

Department of Computer Engineering
Mälardalen University
Västeras, Sweden

Manuel Barranco
Department of Mathematics and Informatics
University of the Balearic Islands
Palma, Spain

Patrice Bodu
Informatics, Mathematics and Automation for La Route Automatisée
National Institute for Research in Computer Science and Control (INRIA)
Rocquencourt, France

DeJiu Chen
Department of Machine Design
Royal Institute of
Technology
Stockholm, Sweden

Mirko Conrad
The MathWorks, Inc.
Natick, Massachusetts

Joaquim Ferreira
Department of Information Technologies Engineering
Polytechnic Institute of Castelo Branco
Castelo Branco, Portugal

Ines Fey
Safety and Modeling Consultants
Berlin, Germany

Ulrich Freund
ETAS
Stuttgart, Germany

Thomas M. Galla
Elektrobit Corporation
Vienna, Austria

Michael Golm
Siemens AG
Princeton, New Jersey

Michael Gonschorek
Elektrobit Corporation
Munich, Germany

Mathieu Grenier
Lorraine Laboratory of Computer Science Research and Applications
Nancy, France

and

University of Nancy
Nancy, France

Hans A. Hansson
Mälardalen Real-Time Research Centre
Mälardalen University
Västeras, Sweden

Bernd Hardung
AUDI AG
Ingolstadt, Germany

Lionel Havet
National Institute for Research in Computer Science and Control (INRIA)
Nancy, France

and

RealTime-at-Work
Nancy, France

Thorsten Kölzow
AUDI AG
Ingolstadt, Germany

Andreas Krüger
AUDI AG
Ingolstadt, Germany

Christian Kühnel
Faculty of Informatics
Technical University of Munich
Garching, Germany

Henrik Lönn
Volvo Technology Corporation
Gothenburg, Sweden

Diana Malvius
Department of Machine Design Royal Institute of Technology
Stockholm, Sweden

Nicolas Navet
National Institute for Research in Computer Science and Control (INRIA)
Nancy, France

and

RealTime-at-Work
Nancy, France

Mikael Nolin
Mälardalen Real-Time Research Centre
Mälardalen University
Västeras, Sweden

Thomas Nolte
Mälardalen Real-Time Research Centre
Mälardalen University
Västeras, Sweden

Roman Pallierer
Elektrobit Corporation
Vienna, Austria

Michel Parent
Informatics, Mathematics and Automation for La Route Automatisée
National Institute for Research in Computer Science and Control (INRIA)
Rocquencourt, France

Holger Pfeifer
Institute of Artificial Intelligence
Ulm University
Ulm, Germany

Juan Pimentel
Electrical and Computer Engineering Department
Kettering University
Flint, Michigan

Julian Proenza
Department of Mathematics and Informatics
University of the Balearic Islands
Palma, Spain

Sasikumar Punnekkat
Mälardalen Real-Time Research Centre
Mälardalen University
Västeras, Sweden

Mark-Oliver Reiser
Software Engineering Group
Technical University of Berlin
Berlin, Germany

Guillermo Rodriguez-Navas
Department of Mathematics and Informatics
University of the Balearic Islands
Palma, Spain

Bernard Sanchez
Continental Automotive GmbH
Toulouse, France

Bernhard Schätz
Faculty of Informatics
Technical University of Munich
Garching, Germany

Françoise Simonot-Lion
Lorraine Laboratory of Computer Science Research and Applications
Nancy, France

and

University of Nancy
Nancy, France

Friedhelm Stappert
Continental Automotive GmbH
Regensburg, Germany

Martin Törngren
Department of Machine Design
Royal Institute of Technology
Stockholm, Sweden

Yvon Trinquet
Institute of Communications Research and Cybernetics of Nantes (IRCCyN)
Nantes, France

and

University of Nantes
Nantes, France

Stefan Voget
Continental Automotive GmbH
Regensburg, Germany

Matthias Weber
Carmeq GmbH
Berlin, Germany

I

Automotive Architectures

1 Vehicle Functional Domains and Their Requirements

Françoise Simonot-Lion
Lorraine Laboratory of Computer Science Research and Applications

Yvon Trinquet
Institute of Communications Research and Cybernetics of Nantes

1.1 General Context

The automotive industry is today the sixth largest economy in the world, producing around 70 million cars every year and making an important contribution to government revenues all around the world [1]. As for other industries, significant improvements in functionalities, performance, comfort, safety, etc. are provided by electronic and software technologies. Indeed, since 1990, the sector of embedded electronics, and more precisely embedded software, has been increasing at an annual rate of 10%. In 2006, the cost of an electronic-embedded system represented at least 25% of the total cost of a car and more than 35% for a high-end model [2]. This cost is equally shared between electronic and software components. These general trends have led to currently embedding up to 500 MB on more than 70 microprocessors [3] connected on communication networks. The following are some of the various examples. Figure 1.1 shows an electronic architecture embedded in a Laguna (source: Renault French carmaker) illustrating several computers interconnected and controlling the engine,

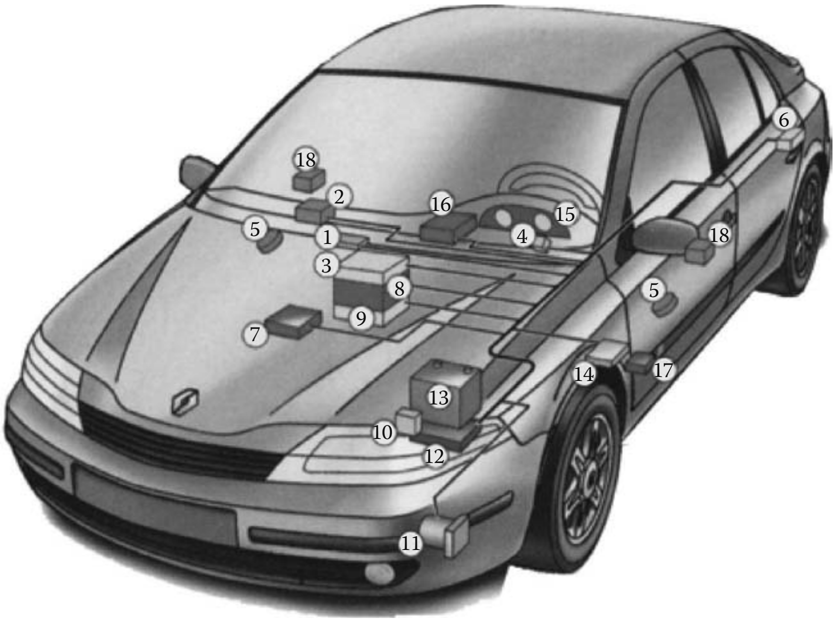


FIGURE 1.1 A part of the embedded electronic architecture of a Renault Laguna. (Courtesy of Renault Automobile. With permission.)

the wipers, the lights, the doors, and the suspension or providing a support for interaction with the driver or the passengers. In 2004, the embedded electronic system of a Volkswagen Phaeton was composed of more than 10,000 electrical devices, 61 microprocessors, three controller area networks (CAN) that support the exchanges of 2500 pieces of data, several subnetworks, and one multimedia bus [4]. In the Volvo S70, two networks support the communication between the microprocessors controlling the mirrors, those controlling the doors and those controlling the transmission system and, for example, the position of the mirrors is automatically controlled according to the sense the vehicle is going and the volume of the radio is adjusted to the vehicle speed, information provided, among others, by the antilock braking system (ABS) controller. In a recent Cadillac, when an accident causes an airbag to inflate, its microcontroller emits a signal to the embedded global positioning system (GPS) receiver that then communicates with the cell phone, making it possible to give the vehicle's position to the rescue service. The software code size of the Peugeot CX model (source: PSA Peugeot Citroen French carmarker) was 1.1 KB in 1980, and 2 MB for the 607 model in 2000. These are just a few examples, but there are many more that could illustrate this very large growth of embedded electronic systems in modern vehicles.

The automotive industry has evolved rapidly and will evolve even more rapidly under the influence of several factors such as pressure from state legislation, pressure from customers, and technological progress (hardware and software aspects). Indeed, a great surge for the development of electronic control systems came through the regulation concerning air pollution. But we must also consider the pressure from consumers for more performance (at lower fuel consumption), comfort, and safety. Add to all this the fact that satisfying these needs and obligations is only possible because of technological progress.

Electronic technology has made great strides and nowadays the quality of electronic components—performance, robustness, and reliability—enables using them even for critical systems. At the same time, the decreasing cost of electronic technology allows them to be used to support any function in a car. Furthermore, in the last decade, several automotive-embedded networks such as local interconnect networks (LIN), CAN, TTP/C, FlexRay, MOST, and IDB-1394 were developed. This has led to the concept of multiplexing, whose principal advantage is a significant reduction in the wiring cost as well as the flexibility it gives to designers; data (e.g., vehicle speed) sampled by one microcontroller becomes available to distant functions that need them with no additional sensors or links.

Another technological reason for the increase of automotive embedded systems is the fact that these new hardware and software technologies facilitate the introduction of functions whose development would be costly or not even feasible if using only mechanical or hydraulic technology. Consequently, they allow to satisfy the end user requirements in terms of safety, comfort, and even costs. Well-known examples are electronic engine control, ABS, electronic stability program (ESP), active suspension, etc. In short, thanks to these technologies, customers can buy a safe, efficient, and personalized vehicle, while carmakers are able to master the differentiation between product variations and innovation (analysts have stated that more than 80% of innovation, and therefore of added value, will be obtained thanks to electronic systems [5]). Furthermore, it also has to be noted that some functions can only be achieved through digital systems. The following are some examples: (1) the mastering of air pollution can only be achieved by controlling the engine with complex control laws; (2) new engine concepts could not be implemented without an electronic control; (3) modern stability control systems (e.g., ESP), which are based on close interaction between the engine, steering, and braking controllers, can be efficiently implemented using an embedded network.

Last, multimedia and telematic applications in cars are increasing rapidly due to consumer pressure; a vehicle currently includes electronic equipment like hand-free phones, audio/radio devices, and navigation systems. For the passengers, a lot of entertainment devices, such as video equipment and communication with the outside world are also available. These kinds of applications have little to do with the vehicle's operation itself; nevertheless they increase significantly as part of the software included in a car.

In short, it seems that electronic systems enable limitless progress. But are electronics free from any outside pressure? No. Unfortunately, the greatest pressure on electronics is cost!

Keeping in mind that the primary function of a car is to provide a safe and efficient means of transport, we can observe that this continuously evolving "electronic revolution" has two primary positive consequences. The first is for the customer/consumer, who requires an increase in performance, comfort, assistance for mobility efficiency (navigation), and safety on the one hand, while on the other hand, is seeking reduced fuel consumption and cost. The second positive consequence is for the stakeholders, carmakers, and suppliers, because software-based technology reduces marketing time, development cost, production, and maintenance cost. Additionally, these innovations have a strong impact on our society because reduced fuel consumption and exhaust emissions improve the protection of our natural resources and the environment, while the introduction of vision systems, driver assistance, onboard diagnosis, etc., targets a "zero death" rate, as has been stated in Australia, New Zealand, Sweden, and the United Kingdom.

However, all these advantages are faced with an engineering challenge; there have been an increasing number of breakdowns due to failure in electric/electronic systems. For example, Ref. [6] indicates that, for 2003, 49.2% of car breakdowns were due to such problems in Germany. The quality of a product obviously depends on the quality of its development, and the increasing complexity of in-vehicle embedded systems raises the problem of mastering their development. The design process is based on a strong cooperation between different players, in particular Tier 1 suppliers and carmakers, which involves a specific concurrent engineering approach. For example, in Europe or Japan, carmakers provide the specification for the subsystems to suppliers, who, in turn, compete to find a solution for these carmakers. The chosen suppliers are then in charge of the design and realization of these subsystems, including the software and hardware components, and possibly the mechanical or hydraulic parts as well. The results are furnished to the carmakers, or original equipment manufacturer (OEM), who install them into the car and test them. The last step consists of calibration activities where the control and regulation parameters are tuned to meet the required performance of the controlled systems. This activity is closely related to the testing activities. In the United States, this process is slightly different since the suppliers cannot really be considered as independent from the carmakers.

Not all electronic systems have to meet the same level of dependability as the previous examples. While with a multimedia system customers require a certain quality and performance, with a chassis control system, safety assessment is the predominant concern. So, the design method for each subsystem depends on different techniques. Nevertheless, they all have common distributed characteristics and they must all be at the level of quality fixed by the market, as well as meeting the safety requirements and the cost requirements. As there has been a significant increase in computer-based and distributed controllers for the core critical functions of a vehicle (power train, steering or braking systems, "X-by-wire" systems, etc.) for several years now, a standardization process is emerging for the safety assessment and certification of automotive-embedded systems, as has already been done for avionics and the nuclear industry, among others. Therefore, their development and their production need to be based on a suitable methodology, including their modeling, a priori evaluation and validation, and testing. Moreover, due to competition between carmakers or between suppliers to launch new products under cost, performance, reliability, and safety constraints, the design process has to cope with a complex optimization problem.

In-vehicle embedded systems are usually classified according to domains that correspond to different functionalities, constraints, and models [7–9]. They can be divided among "vehicle-centric" functional domains, such as power train control, chassis control, and active or passive safety systems and "passenger centric" functional domains where multimedia/telematics, body/comfort, and human–machine interface (HMI) can be identified.

1.2 Functional Domains

Carmakers distinguish several domains for embedded electronics in a car, even though sometimes the membership of only one domain for a given compartment is not easy to justify. According to the glossary of the European ITEA EAST-EEA project [10], a domain is defined as "a sphere of knowledge, influence, and activity in which one or more systems are to be dealt with (e.g., are to be built)." The term domain can be used as a means to group mechanical and electronic systems.

Historically, five domains were identified: power train, chassis, body, HMI, and telematics. The power train domain is related to the systems that participate in the longitudinal propulsion of the vehicle, including engine, transmission, and all subsidiary components. The chassis domain refers to the four wheels and their relative position and movement; in this domain, the systems are mainly steering and braking. According to the EAST-EEA definition, the body domain includes the entities that do not belong to the vehicle dynamics, thus being those that support the car's user, such as airbag, wiper, lighting, window lifter, air conditioning, seat equipment, etc. The HMI domain includes the equipment allowing information exchange between electronic systems and the driver (displays and switches). Finally, the telematic domain is related to components allowing information exchange between the vehicle and the outside world (radio, navigation system, Internet access, payment).

From one domain to another, the electronic systems often have very different features. For example, the power train and chassis domains both exhibit hard real-time constraints and a need for high computation power. However, the hardware architecture in the chassis domain is more widely distributed in the vehicle. The telematic domain presents requirements for high data throughput. From this standpoint, the technological solutions used are very different, for example, for the communication networks, but also for the design techniques and verification of the embedded software.

1.2.1 Power Train Domain

As mentioned previously, this domain represents the system that controls the engine according to requests from the driver (e.g., speeding up, slowing down as transmitted by the throttle position sensor or the brake pedal, etc.) and requirements from other parts of the embedded system such as climate control or ESP; the controller acts according to natural factors such as air current temperature, oxygen level, etc. on the one hand, and to environmental annoyances such as exhaust pollution, noise, etc. on the other. It is designed to optimize certain parameters like driving facilities, driving comfort, fuel consumption, etc. One parameter that could be controlled by such a system is the quantity of fuel that has to be injected into each cylinder at each engine cycle according to the engine's revolutions per minute (rpm) and the position of the gas pedal. Another is the ignition timing, and even the so-called variable valve timing (VVT) that controls the time in the engine cycle at which the valves open. There are still others such as the optimal flow of air into the cylinder, the exhaust emission, and the list goes on.

Some information, such as the current rpm, the vehicle speed, etc., are transmitted by this system to another one whose role is to present them to the driver on a dashboard; this last component is actually part of the HMI domain.

The main characteristics for the embedded systems of the power train domain are

- From a functional point of view: The power train control takes into account the different working modes of the motor (slow running, partial load, full load, etc.); this corresponds to various and complex control laws (multivariables) with different sampling periods. Classical sampling periods for signals provided by other systems are 1, 2, or 5 ms, while the sampling of signals on the motor itself is in phase with the motor times (from 0.1 to 5 ms).
- From a hardware point of view: This domain requires sensors whose specification has to consider the minimization of the cost/resolution criteria. When it is economically possible for the targeted vehicle, there are also microcontrollers that provide high computation power, thanks to their multiprocessor architecture and dedicated coprocessors (floating point computations), and high storage capacity. Furthermore, the electronic components that are installed into the hardware platform have to be robust to interferences and heat emitted by the engine itself.
- From an implementation point of view: The specified functions are implemented as several tasks with different activation rules according to the sampling rules, with stringent time constraints imposed on task scheduling, mastering safe communications with other systems, and with local sensors/actuators.

Continuous, sampled, and discrete systems are all found in this domain. The control laws contain many calibration parameters (about 2000). Their specification and validation are supported by tools such as Matlab/Simulink [11]. Their deployment and their implementation are the source of a lot of technical problems. For example, underlying control models are generally based on floating point values. If, for economical reasons, the implementation has to be done on a microcontroller without a floating point coprocessor, the programmer has to pay attention to the accuracy of the values in order to be sure to meet the precision required at the specification level of the control laws [12,13]. Another major challenge, as mentioned previously, is to efficiently schedule cyclic activities, because some of them have constant periods, while others have variable periods, according to the motor cycles. This means that scheduling them depends on different logical clocks [14]. Currently, the validation of the control laws is mainly done by simulation and, for their integration, by emulation methods and/or testing. Since the power train domain is subject to hard real-time constraints, performance evaluation and timing analysis activities have to be performed on their implementation models first.

1.2.2 Chassis Domain

The chassis domain is composed of systems whose aim is to control the interaction of the vehicle with the road (wheel, suspension, etc.). Controllers take into account the requests emitted by the driver (steering, braking, or speed up orders), the road profile, and the environmental conditions, like wind, for example. They have to ensure the comfort of the driver and the passengers (suspension) as well as their safety. This domain includes systems like ABS, ESP, automatic stability control (ASC), and four-wheel drive (4WD). The chassis domain is of the utmost importance for the safety of the passengers and of the vehicle itself. Therefore, its development has to be of high quality, as for any critical system.

The characteristics of the chassis domain and the underlying models are similar to those presented for the power train domain: multivariable control laws, different sampling periods, and stringent time constraints (around 10 ms). As for the power train domain, the systems controlling the chassis components are fully distributed onto a networked microcontroller and they communicate with other systems. For example, an ESP system corrects the trajectory of the vehicle by controlling the braking system. Its role is to automatically correct the trajectory of the vehicle as soon as there is understeering or oversteering. To do this, it has to compare the steering request of the driver to the vehicle's response. This is done via several sensors distributed in the vehicle (lateral acceleration, rotation, individual wheel speeds), taking samples 25 times per second. As soon as a correction needs to be applied, it will brake individual front or rear wheels and/or command a reduction of engine power to the power train systems. This system cooperates online with various others such as ABS, electronic damper control (EDC) [15], etc., in order to ensure the safety of the vehicle.

Furthermore, X-by-wire technology, currently applied in avionic systems, is emerging in the automotive industry. X-by-wire is a generic term used when mechanical and/or hydraulic systems are replaced by electronic ones (intelligent devices, networks, computers supporting software components that implement filtering, control, diagnosis, and functionalities). The purpose of such a technology is to assist the driver in different situations in a more flexible way and to decrease production and maintenance cost for braking or steering systems. Nowadays, vehicles equipped with X-by-wire systems have kept traditional mechanical technologies as a backup in case the electronic ones fail. The suppression of this backup presents a major challenge in embedded system design. Conventional mechanical and hydraulic systems have stood the test of time and have proved themselves to be reliable. Therefore, a pure X-by-wire system has to reach at least the same level of safety assessment, with redundancy, replica, functional determinism, and fault tolerance being some of the key underlying words. X-by-wire systems have been used in the avionic industry for some time and so some lessons can be learned from this experience. Nevertheless, due to economical reasons as well as space and weight constraints, the solutions used in an avionics context cannot be compared to that of automotives (in particular, it is impossible to have the same level of hardware redundancy). So, specific fault-tolerant solutions need to be developed. Note that this domain will be mainly concerned by the emerging standard ISO 26262 (committee draft put to the ballot in 2008) on the safety of in-vehicle embedded systems and the certification process that will be required (Section 1.4). It should be noted that, for this domain, the time-triggered software technologies [16,17] bring well-suited solutions despite their lack of flexibility. The Flexray network, the OSEKtime operating system (Offene Systeme und deren schnittstellen für die Elektronik im Kraft-fahrzeug) time operating system and the associated Fault-Tolerant communication (FTCom), or the basic software of AUTomotive Open Standard ARchitecture AUTOSAR (Chapter 2) are good candidates for the implementation of such systems.

1.2.3 Body Domain

The body domain contains functions embedded in a vehicle that are not related to the control of its dynamics. Nowadays, wipers, lights, doors, windows, seats, and mirrors are controlled more and more by software-based systems. In general, they are not subject to stringent performance constraints and, from a safety standpoint, they do not represent a critical part of the system. However, there are certain functions, like an advanced system whose aim is to control access to the vehicle for security, that have to respect hard real-time constraints. It has to be noted that the body functions often involve many communications between each other and consequently have a complex distributed architecture. In this domain emerges the notion of subsystem or subcluster based on low cost sensor–actuator level networks, for example, LIN, which connects modules constructed as integrated mechatronic systems. For example, several functions can be associated to a door: lock/unlock control according to a signal transmitted by a wireless network, window control according to passenger or driver request, as well as mirror control and seat position control. One possible deployment of these functions could be that one main electronic control unit (ECU) supports the reception of the requests (lock/unlock, window up/down, seat up/down, etc.) while the controllers for the motors realizing the requested actions on the physical device (mirror, window, seat) are supported by three other ECUs (Figure 1.2). These four ECUs are connected on a LIN. As some requests concern several doors (e.g., the lock/unlock request), the main ECUs of each door are also connected, for example, on a low-speed CAN. Finally, in order to present the status of the doors to the driver (doors open/close, windows open/close), the information is transmitted by the main ECUs to the dashboard ECU via the CAN low-speed network.

On the other hand, the body domain also contains a central subsystem, termed the central body electronic, whose main functionality is to ensure message transfers between different systems or domains. This system is recognized to be a critical central entity. The body domain functions are related mainly to discrete event applications and their design and validation rely on state machines such as SDL, statecharts, UML state transition diagrams, and synchronous models. These models validate a functional specification, by simulation and, when possible, by model checking. Their implementation, as mentioned before, implies a distribution of this functional specification over hierarchically distributed hardware architecture. High computation power is needed for the central body electronic entity, and, as with the two previous domains, fault tolerance and reliability properties are obligatory for body systems. Although timing constraints are not so stringent as those for the power train and chassis systems, the end-to-end response time between stimuli and response must be evaluated,

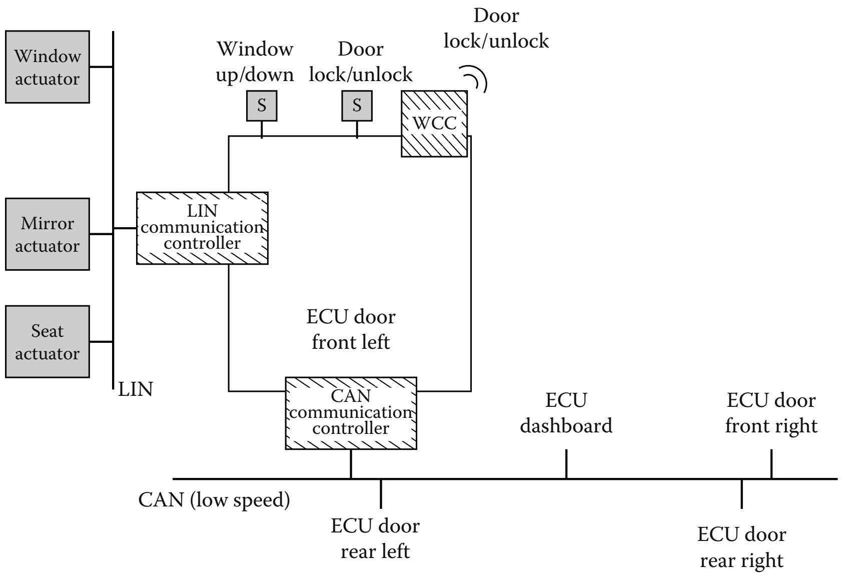


FIGURE 1.2 Example of doors control and of its deployment.

taking into account the performances of the hardware platform, the scheduling policies for each microcontroller, and the network protocol. In fact, the designer has to prove that these response times are always acceptable and therefore that the responses of each stimulus are done in a bounded interval. One challenge in this context is, first, to be able to develop an exhaustive analysis of state transition diagrams and, second, to ensure that the implementation respects the fault tolerance and safety constraints. The problem here is to achieve a balance between the time-triggered approach and flexibility.

1.2.4 Multimedia, Telematic, and HMI

Telematics in vehicles includes systems that support information exchanges between vehicles or between vehicle and road infrastructures. For example, such systems are already used for collecting road tolls; in the near future, telematics will make it possible to optimize road usage through traffic management and congestion avoidance (Chapter 3), to automatically signal road collisions, to provide remote diagnostics (Section 1.2.6), or even to provide access to on-demand navigation, on-demand audio-video entertainment, Web surfing, sending or receiving e-mails, voice calls, short message services (SMSs), etc.

HMI systems support, in a general sense, the interaction between the driver and the passengers with numerous functions embedded in the car. Their main functionalities are, on the one hand, presenting information about the status of the car (e.g., the vehicle speed, the oil level, the status of a door, the status of lights, etc.), the status of a multimedia device (e.g., current frequency for a radio device, etc.), or the result of a request (e.g., visualization of a map provided by a navigation system) and, on the other hand, receiving requests for multimedia equipment (command for radio, navigation systems, etc.). The next generation of multimedia devices will provide new sophisticated HMIs related mainly to entertainment activities. A challenge for HMI system development is thus to take into account, not only the quality, performance, and comfort of the system, but also the impact of this technology on safety [18]. In fact, using HMI must be simple and intuitive, and should not disturb the driver. One way to control the multimedia systems is to avoid too many buttons. The commands should be grouped in a way that minimizes the movements of the driver. For example, the most common solution is to group them on the steering hand wheel—there are as many as 12 buttons on a steering wheel for a high-end model, causing potential confusion between them. Where and how to present information to the driver is also a major problem. The information needs to be clear and should not distract the driver's attention from the road. For example, in the new Citroën C6 a head-up display (HUD) allows key driving information (the vehicle speed, etc.) to be shown on the windscreen in the driver's direct line of vision. Thanks to such a system, the driver can read the information without looking away from the road, as is now done with a traditional dashboard.

Multimedia and telematic devices will be upgradeable in the future and, for this domain, a plug-and-play approach is preferable. These applications need to be portable and the services furnished by the platform (operating system and/or middleware) should offer generic interfaces and downloading facilities. The main challenge here is to preserve the security of the information from, to, or within the vehicle. Sizing and validation do not rely on the same methods as those for the other domains. Here, we shift from considering messages, tasks, and deadline constraints toward fluid data streams, bandwidth sharing, and multimedia quality of service, and from safety and hard real-time constraints toward security for information and soft real-time constraints. Nevertheless, the optimal sizing of these systems can be difficult to determine. For example, a telematics and multimedia platform integrated into a high-end car can be composed of two processors on which about 250 threads running on a Java Machine or on a multitask operating systems will be scheduled. These threads are in charge of handling the data stream and their schedule must give the quality of service required by the user. This kind of system is recognized as "soft" or "firm" real time because it is admissible for some instances of threads, depending on the current load of the processor, to be rejected without significantly reducing the quality of service.

According to experts of this domain, communication between a car and its environment (vehicle-to-vehicle [V2V] or vehicle-to-infrastructure [V2I]) will become more and more important in future years and will bring with it various services with strong added value. The future technologies in this domain begin with efficient voice recognition systems, line-of-sight operated switches, virtual keyboards, etc. but will evolve to include new systems that monitor the status of the vehicle and consequently manage the workload of the driver by avoiding, for example, the display of useless information.

1.2.5 Active/Passive Safety

Demands for vehicles ensuring the safety of driver and passengers are increasing, and are both customer-driven as well as regulatory-based. As mentioned in Section 1.1, the challenge to the automotive industry is to design cars whose embedded systems are able to reach the required safety level at minimal costs. In fact, automotive embedded safety systems target two objectives: "active safety" and "passive safety," the former letting off a warning before a crash and the latter acting after a crash. Seat belts and airbags are examples of systems that help to reduce the effects of an accident, and so they contribute to passive safety. Nowadays, the passive safety domain has reached a good maturity level. An airbag is controlled by a complex algorithm embedded on an ECU and consumes information provided by other systems. Alerted by signals coming from various sensors (deceleration, vehicle speed), this algorithm regulates the right moment to deploy the airbags. The device has to work within a fraction of a second from the time a crash is detected by the sensor to its activating the airbag. As far back as 1984, the U.S. government required cars being produced after April 1, 1989 to have airbags on the driver's side (U.S. Department of Transportation) and in 1998, dual front airbags also became mandatory. Active safety refers to avoiding or minimizing an accident and systems such as braking systems, ABS, ESP, lane keeping, etc., have been specified and marketed for this purpose. The most advanced technological solutions (Chapter 3) are adaptive cruise control and collision warning/avoidance/mitigation systems that contribute to the concept of advanced driver assistance. In general, active safety systems interpret signals provided by various sensors and other systems to assist the driver in controlling the vehicle and interact strongly with almost all the systems embedded in the car.

1.2.6 Diagnostic

As shown in the examples presented in the previous sections, nowadays the complexity of electronic architectures embedded in a car infers functions deployed on several microcontrollers to intensively interact between themselves. Therefore, diagnosis has become a vital function throughout the lifetime of a vehicle. So, any system that can help to access and relate information about a car is obviously very important and should be designed simultaneously with the original design of the car. In particular, specifying a system that is able to collect information and establish onboard diagnostics (OBD) is advantageous for the vehicle's owner as well as for a repair technician. The generic term used for this function is "onboard diagnostics" or OBD. More precisely, this concept refers to self-diagnosis and reporting facilities, which were made possible with the introduction of computer-based systems that could memorize large amounts of information. While the role of early diagnostic functions was limited to a light switching on as soon as a specific problem was detected, recent OBD systems are based on standardized communication means—a standardization of monitored data and a standardized coding and reporting of a list of specific failures, termed diagnostic trouble codes (DTC). Thanks to this standardization effort, the memorized values of parameters can be analyzed through a single compliant device. The underlying intent to this standardization effort was a regulatory constraint on exhaust emission control systems throughout the useful lifetime of a vehicle. The OBD-II specification, mandatory for all cars sold in the United States as of 1996, precisely defines the diagnostic connector, the electrical signaling protocol, the messaging format, as well as the vehicle parameters that can be monitored. In 2001, the European Emission Standards Directive 98/69/EC [19] established the requirement of EOBD, a variant of OBD-II, for all petrol vehicles sold in the European Union as of January 2001. Several standards have been successively provided: ISO 9141-2 concerns a low-speed protocol close to that of RS-232 [20], ISO 14230 introduced the protocol KWP2000 (Keyword Protocol 2000) that enables larger messages [21], and ISO 15765 proposes a diagnostic, termed Diag-on-CAN, which uses a CAN [22]. The next step forecast will enable reporting emissions violations by the means of a radio transmitter.

1.3 Standardized Components, Models, and Processes

As pointed out in Section 1.1, the design of new in-vehicle embedded systems is based on a cooperative development process. Therefore, it must ensure the interoperability between components developed by different partners and ease their portability onto various platforms in order to increase the system's flexibility. On the one hand, a means to reach these objectives is furnished by the standardization of services sharing hardware resources between application processes, in particular, networks and their protocols and operating systems. On the other hand, portability is achieved through the specification of a common middleware. Notice that such a middleware also has to deal with the interoperability properties. Finally, a standardized and common support for modeling and documenting systems all along their development eases the design process itself and, more specifically, the exchanges between the different partners at each step of the development cycle. In the following, we introduce some of the standardized components or models aiming to support this cooperative development process.

1.3.1 In-Vehicle Networks and Protocols

Specific communication protocol and networks have been developed to fulfill the needs of automotive-embedded systems. In 1993, the SAE Vehicle Network for Multiplexing and Data Communications Standards Committee identified three kinds of communication protocols for in-vehicle embedded systems based on network speed and functions [23]; they are called, respectively, "class A," "class B," and "class C." The same committee also published a requirement list concerning safety critical applications. In particular, the communication protocol for X-by-wire systems must respect requirements for "dependability and fault-tolerance" as defined for class C [24]. Networks compliant to class A provide a bit rate below 10 kbps and are dedicated to sensor and actuator networks; the LIN bus and TTP/A bus are among the most important protocols in this class. Class B specifies a medium speed (10–500 kbps) and is thus convenient for transferring information in vehicle-centric domains and the body's electronics systems. CAN-B is a widely used class B protocol. Class C has been defined for safety-relevant systems in power train or chassis domains. The data rates here are lower than 1 Mbps. CAN-C (high-speed CAN), TTP/C, and FlexRay fall into this category. They have to provide highly reliable and fault tolerant communication. Obviously, class C networks will be required in future X-by-wire applications for steering and braking. For further information on automotive-embedded networks, the reader can refer to Chapter 4 as well as to Refs. [25,26].

1.3.2 Operating Systems

OSEK/VDX [27] is a multitask operating system that is becoming a standard in the European automotive industry. This standard is divided in four parts: OSEK/VDX OS is the specification of the kernel itself; OSEK/VDX COM concerns the communication between tasks (internal or external to an ECU); OSEK/VDX NM addresses network management; and finally, OSEK/VDX OIL is the language that supports the description of all the components of an application. Certain OSEK-targeted applications are subject to hard real-time constraints, so the application objects supported by OSEK have to be configured statically.

OSEK/VDX OS provides services on objects like tasks ("basic tasks," without blocking point, and "extended tasks," that can include blocking points), events, resources, and alarms. It proposes a fixed priority (FP) scheduling policy that is applied to tasks that can be preemptive or non-preemptive, and combined with a reduced version of the priority ceiling protocol (PCP) [28,29] in order to avoid priority inversion or deadlock due to exclusive resource access. Intertask synchronization is achieved through private events and alarms. The implementation of an OSEK/VDX specification has to be compliant to one of the four conformance classes—BCC1, BCC2, ECCI, ECC2—that are specified according to the supported tasks (basic only or basic and extended), the number of tasks on each priority level (only one or possibly several), and the constraints of the reactivation counter (only one or possibly several). BCC1 defines a restricted implementation that aims to minimize the size of the corresponding memory footprint, the size of the data structures, and the complexity of the management algorithms. ECC2 specifies the implementation of all the services. The MODISTARC project (Methods and tools for the validation of OSEK/VDX based DISTributed ARChitectures) [30] provided the relevant test methods and tools to assess the compliance of OSEK/VDX implementations.

In order to describe an application configuration, the OSEK consortium provided a specific language, called OSEK/VDX OIL (OSEK Implementation Language). This language allows, for one ECU, the description of several application configurations, called application modes. For example, the application configurations can be specified for a normal operation mode, for a diagnosis mode, and for a download mode.

The dependability purpose and fault tolerance for critical applications is usually achieved by a time-triggered approach [17]. So, the time-triggered operating system OSEKtime [27] was defined. It supports static and time-triggered scheduling, and offers interrupt handling, dispatching, system time and clock synchronization, local message handling, and error detection mechanisms. Thanks to these services, an application running on OSEKtime can be predictable. OSEKtime is compatible to OSEK/VDX and is completed by FTCom layer for communication services.

It should be noted that the specification of the basic software for AUTOSAR (Chapter 2) is based on services from OSEK and OSEKtime. Commercial implementations of OSEK/VDX standard are available [27] and open-source versions as well [31,32].

Rubus is another operating system tailored for the automotive industry and used by Volvo Construction Equipment. It was developed by Arcticus systems [33]. Rubus OS is composed of three parts: the Red Kernel, which manages the execution of off-line scheduled time-triggered tasks; the Blue Kernel, which is dedicated to the execution of event-triggered tasks; and the Green Kernel, which is in charge of external interrupts. As for OSEK/VDX OS, the configuration of the tasks has to be defined statically off-line.

For multimedia and telematics applications, the operating systems are generic ones, such as VxWorks (from WindRiver) or even a Java machine. "Microsoft Windows Automotive 5.0" extends the classical operating system Windows CE with telematic-oriented features and was, for example, installed among others in certain Citroën Xsara and the BMW 7 series.

1.3.3 Middleware

Flexibility and portability of applicative components require two main features. On the one hand, an application embedded on a distributed platform is based on the description of elements, the semantics of the interaction types among the elements, and, consequently, the semantics of their composition. Note that these interactions must be specified disregarding the allocation of components on an ECU. On the other hand, the properties required at the application level, mainly timing and dependability properties, must be met when components are allocated onto a technical platform (operating systems, communication drivers and protocol, input/output [I/O] drivers, etc.). Traditionally, these features are achieved through the specification of a middleware. Firstly, the structure of the middleware, that is, the elementary software components allocated on each ECU and the way they interact, has to be formally identified and, secondly, the interface services that furnish a way for applicative components to use the middleware services independently of their allocation have to be furnished. During the last decade, several projects focused on this purpose (see, e.g., the German Titus project [34] started by DaimlerChrysler in 1994). The purpose of this project was to develop an interface-based approach similar to the ROOM methodology [35], but differing considerably in certain details, mainly in making an "actor-oriented" approach that was suitable for ECU software. The French EEA project [36] identified the classes of software components implemented on an ECU. Then the European ITEA EAST EEA project refined these classes and proposed a more advanced architectural view of an ECU [10]. The mission of the DECOS project, supported by the sixth EU Framework Program, was to develop an architecture-based design methodology, to identify and specify the associated components off-the-shelf (COTS) hardware and software components, and to provide certified development tools and advanced hybrid control technologies. This project targeted control systems concerning the dependability of software-intensive systems, in particular, in avionics (airbus) and automotive industries. After that, the Volcano project concentrated on just the communication services and provided a way (both middleware components and interface services) for supporting the signal exchanges between distant applicative components by hiding the underlying protocol. Volcano targeted the timing properties imposed on signal exchanges [37,38].

Finally, the AUTOSAR consortium (see Chapter 2 for more details) standardized a common software infrastructure for automotive systems [39]. Once put into practice, it will bring meaningful progress when designing an embedded electronic architecture because (1) it will allow the portability of the functions on the architecture and their reuse, (2) it will allow the use of hardware COTS, and (3) on a same ECU it will be able to integrate functions from different suppliers. During the lifetime of the car, this standard will facilitate updating the embedded software as this technology evolves, as well as the maintenance for the computers.

1.3.4 Architecture Description Languages for Automotive Applications

Sharing the same modeling language between the different partners involved in the design of these in-vehicle embedded systems is a means to support an efficient collaborative development process. In this context, such a language will have to allow for describing a system at different steps of its development (requirement specification, functional specification, design, implementation, tuning, etc.) by taking into consideration the different viewpoints of the actors as well as ensuring a consistency between these different views. It will also need to reflect the structure of the embedded systems as an architecture of components (hardware components, functional components, software components). The concept of architecture description languages (ADLs), developed for large software applications [40], is well suited to these objectives. ADLs are used to describe the structure of a system by means of the components that are interconnected in a way to form configurations. These descriptions are free of implementation details, one of the objectives being the mastery of the structure of complex systems. Thus the composition (associated to hierarchy) used to specify the assembly of the elements constitutes the fundamental construction. For critical systems, as is the case in automotive electronics, an ADL must support not only the specification of the functional aspects of the system, but also those that are extra-functional (timing properties, dependability properties, safety properties), and other transformation and verification facilities between design and implementation, while maintaining a consistency between the different models. In 1991, Honeywell Labs specified an ADL, MetaH [41], that was dedicated to avionics systems. This language was chosen in 2001 to be the core of an avionics ADL (AADL) standard under the SAE authority [42]. For the specific automotive domain, several languages were proposed (Chapter 9). For example, the language EAST-ADL [43], which is tightly related to the generic reference architecture mentioned in the previous section, was specified in the European ITEA EAST-EEA project [10] and extended in the ATESST project [44]. The purpose of EAST-ADL is to provide support for the nonambiguous description of in-car embedded electronic systems at each level of their development. It provides a framework for modeling such systems through five abstraction levels, divided into seven layers (also called artifacts), as shown in Figure 1.3. Some of these layers are mainly concerned with software development while others are linked to the execution

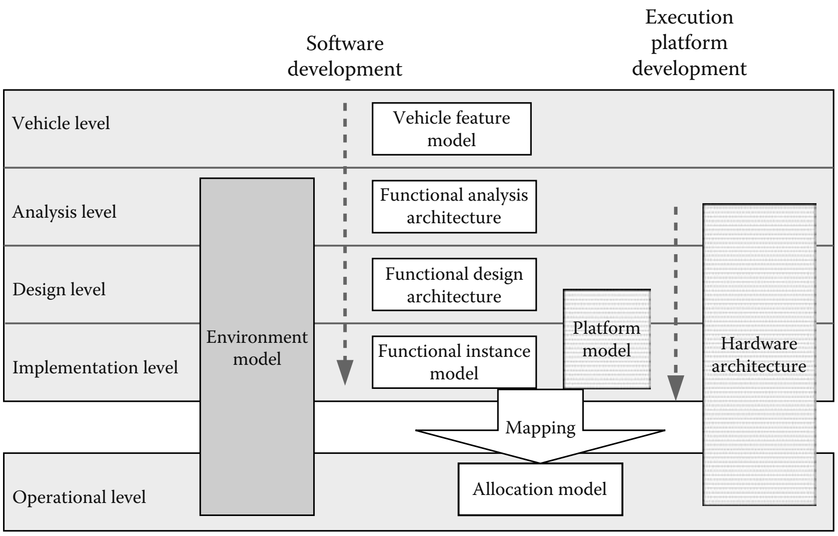


FIGURE 1.3 The abstraction levels and the system views in EAST-ADL.

platform (ECUs, networks, operating systems, I/O drivers, middleware, etc.). All these layers are tightly linked, allowing traceability among the different entities that are implicated in the development process. Besides the structural decomposition, which is typical for any software development or modeling approach, the EAST-ADL also has means for modeling cross-cutting concerns such as requirements, behavioral description and validation, and verification activities. At vehicle level, the vehicle feature model describes the set of user-visible features. Examples of such features are antilock braking or windscreen wipers. The functional analysis architecture, at the analysis level, is an artifact that represents the functions that realize the features, their behavior, and their cooperation. There is an n-to-n mapping between vehicle feature model and functional analysis architecture entities, that is, one or several functions may realize one or several features. The functional design architecture (design level) models a decomposition or refinement of the functions described at analysis level in order to meet constraints regarding allocation, efficiency, reuse, supplier concerns, etc. Again, there is an n-to-n mapping between the entities for functional design architecture and the corresponding ones in functional analysis architecture. At the implementation level, the role of the function instance model is to prepare the allocation of software components and exchanged signals to OS tasks and frames. It is, in fact, a flat software structure where the functional design architecture entities have been instantiated. It provides an abstraction of the software components to implement. In order to model the implementation of a system, EAST-ADL furnishes, on the one hand, a way to describe the hardware platforms and their available services (operating system, protocol, middleware) and, on the other hand, a support for the specification of how a function instance model is distributed onto a platform. This is done thanks to three other artifacts. The hardware architecture includes the description of the ECUs and, more precisely, those for the microcontroller used, the sensors and actuators, the communication links (serial links, networks), and their connections. The platform model defines the operating system and/or middleware application programming interface (API) and, in particular, the services provided (schedulers, frame packing, memory management, I/O drivers, diagnosis software, download software, etc.). Finally, the allocation model is used at the operational level. It models the tasks that are managed by the operating systems and frames, which are in turn managed by the protocol. This is the result of the function instance model entities being mapped onto the platform model. Note that the specification of a hardware architecture and a platform model is done simultaneously with function and software specification and can even be achieved during the definition of an allocation model. At this lowest abstraction level, all of the implementation details are captured. The EAST-ADL language provides consistency within and between the artifacts belonging to the different levels from a syntactic and semantic point of view. This makes an EAST-ADL-based model a strong and nonambiguous support, not only for the realization of software components, but also for building, automatically, models that are suited for format validation and verification activities [45,46].

1.4 Certification Issue of Safety-Critical In-Vehicle Embedded Systems

Several domains are recognized as critical, for example, nuclear plants, railways, avionics. They are subject to strong regulations and must prove that they meet rigorous safety requirements. Therefore, the manner of specification and the management of the dependability/safety properties represent an important issue, as well as the certification process. This problem has become of primary importance for the automotive industry due to the increasing number of computer-based systems such as critical functions like steering and braking. Consequently, several proposals have been under study. The existing certification standards [47], ARP 4754 [48], RTCA/DO-178B [49] (used in avionics), or EN 50128 [50] (applied in the railway industry), provide stringent guidelines for the development of a safety-critical embedded system. However, these standards are hardly transposable for in-vehicle software-based systems: partitioning of software (critical/noncritical), multiple versions, dissimilar software components, use of active redundancy, and hardware redundancy. In the automotive sector, the Motor Industry Software Reliability Association (MISRA), a consortium of the major actors for automotive products in the United Kingdom, proposes a loose model for the safety-directed development of vehicles with onboard software [51]. Also, the generic standard IEC 61508 [52], used for electrical/electronic/programmable electronic systems appears to be a good candidate for supporting a certification process in the automotive industry. Finally, an upcoming standard is being developed, derived from that for the IEC, which serves automotive-specific needs.

The ISO international draft standard ISO WD 26262, planned for 2008, is currently under progress in cooperation with the EU, the United States, and Japan [53,54]. The next step will consist in the assessment of its usability by the members of the ISO association. The ISO WD 26262 standard is applied to functional safety, whose purpose is to minimize the danger that could be caused by a possibly faulty system. The ISO draft specifies that functional safety is ensured when "... a vehicle function does not cause any intolerable endangering states, which are resulting from specification, implementation or realization errors, failure during operation period, reasonably foreseeable operational errors [and/or] reasonably foreseeable misuse." This definition concerns, in fact, the entire life cycle of a system. Safety control has to be effective during the preliminary phase of the system design (in particular, hazard analysis and risk assessment), during development (functional safety requirement allocation for hardware and software, and system evaluation), and even during operation services and decommissioning (verification that assumptions made during safety assessment and hazard analysis are still present). Once the function of a system has been specified, the safety process dictates that it goes over an established list of driving situations and their corresponding malfunctions and, for each one of them, gives the safety functions that are specified to avoid such situations as well as how to maintain the vehicle in a safe mode. Each of these situations is characterized by the frequency of its occurrences, the severity of the damage, and the controllability of the situation by a driver. The system is characterized according to these parameters by a so-called automotive safety integrity level (ASIL). The format definition of the safety properties associated to each ASIL is not known at the present time. If we refer to the generic standard IEC 61508 [52], each SIL is defined by two kinds of safety properties: functional requirements, that is, no erroneous signals are produced by an ECU, and safety integrity attributes, that is, the probability of dangerous failure occurrences per hour has to be less than a given threshold (e.g., less than 10^{-8}). Throughout the development of the system that realizes a function, it must be verified that this system ensures all the properties required by the SIL assigned to the function. Verification activities are based, for example, on failure mode and effect analysis (FMEA), fault or event tree analysis, etc. completed by several techniques that could depend on the development process stage (formal methods and model checking, performance evaluation, schedulability and timing analysis, probability, hardware in the loop, system in the loop, etc.).

1.5 Conclusion

Nowadays, for any activity in our everyday life, we are likely to use products or services whose behavior is governed by computer-based systems, also called embedded systems. This evolution also affects the automotive industry. Several computers are embedded in today's vehicles and ensure functions that are vehicle centric, such as motor control, braking assistance, etc., as well as passenger centric such as entertainment, seat control, etc. This chapter has shown why this evolution is inescapable and has outlined the main thrusts of this development. First, state regulations, such as controlling exhaust emissions or mandatory active safety equipments (e.g., airbags), which impose embedding complex control laws that can only be achieved with computer-based systems. Second, customers are asking for more comfortable, easy-to-drive, and safe cars and carmakers are aiming to launch new innovative products; both are costly. Today's advancing software technology appears to be a good trade-off between cost and product development, and therefore facilitates the introduction of new services in cars. In order to identify the requirements applied to embedded electronic systems, we presented a classification of these systems according to well-identified functional domains. The pressure of these requirements affects the technological solutions in terms of hardware components as well as software development. Finally, the economical constraints push for the emergence of standards easing hardware/software independence, and consequently an efficient collaborative development process of embedded electronic architectures (Chapter 2) and the reuse of hardware and software entities (Chapter 8). For example, at the present time, the CAN is predominant in the interconnection of the ECUs. However, due to the increase in exchanges between ECUs, other solutions are emerging (e.g., the FlexRay network, the integration of mechatronic systems deployed on hierarchical distributed architecture, etc.). The growing complexity of the software embedded in a car reflects a well-mastered development process. Autonomous and automated road vehicles, communicating cars, and integrated traffic solutions are keywords for the vehicle of the future. These trends target controlling motorized traffic, decreasing congestion and pollution, and increasing safety and quality of lives (Chapter 3). In such a scenario, the development of a vehicle cannot be considered separately, but must be seen as part of a complex system. Furthermore, the next standard OSI 26262, and those that are already being applied for road traffic, form another strong argument for solid, structured design methods. Thanks to international initiatives, such as AUTOSAR, the concepts of model-based development (MBD), model-driven development (MDD), and component-based software engineering (CBSE) are penetrating the culture of automotive system designers. This will be possible as soon as tools supporting these concepts, and suited to the automotive industry, reach a higher level of maturity.

References

1. OICA. International Organization of Motor Vehicle Manufacturers. http://www.oica.net.
2. SAE. International Society of Automotive Engineers. http://automobile. sae.org/.
3. P. Hansen. New S-class Mercedes: Pioneering electronics. *The Hansen Report on Automotive Electronics*, 18(8), October 2005.
4. J. Leohold. Communication requirements for automotive systems. In *Slides Presented at the 5th IEEE International Workshop on Factory Communication System*, WFCS'2004, Vienna, Austria, September 2004.
5. G. Leen and D. Heffernan. Expanding automotive electronic systems. *IEEE Computer*, 35(1), January 2002, pp. 88–93.
6. E. Knippel and A. Schulz. Lessons learned from implementing configuration management within electrical/electronic development of an automotive OEM. In *International Council on Systems Engineering, INCOSE 2004*, Toulouse, France, June 2004.
7. S. Fürst. AUTOSAR for safety-related systems: Objectives, approach and status. In *Second IEE Conference on Automotive Electronics*, London, United Kingdom, March 2006.
8. A. Sangiovanni-Vincentelli. Automotive electronics: Trends and challenges. In *Convergence 2000*, Detroit, MI, October 2000.

9. F. Simonot-Lion. In-car embedded electronic architectures: How to ensure their safety. In *Fifth IFAC International Conference on Fieldbus Systems and their Applications, FeT'2003*, Aveiro, Portugal, July 2003.
10. ITEA EAST-EEA project. http://www.east-eea.net.
11. The MathWorks. MATLAB/SIMULINK. http://www.mathworks.com.
12. T. Hilaire, Ph. Chevrel, and Y. Trinquet. Designing low parametric sensitivity FWL realizations of LTI controllers/filters within the implicit state-space framework. In *44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*, Séville, Spain, December 2005.
13. T. Hilaire, Ph. Chevrel, and J. Whidborne. A unifying framework for finite wordlength realizations. *IEEE Transactions on Circuits and Systems-I, Fundamental Theory and Applications*, 54(8): 1765–1774, 2007.
14. C. André, F. Mallet, and M.-A. Peraldi Frati. A multiform time approach to real-time system modelling. In *IEEE Second International Symposium on Industrial Embedded Systems—SIES'2007*, Lisbon, Portugal, July 2007.
15. A. Schedl. Goals and architecture of FlexRay at BMW. In *Slides Presented at the Vector FlexRay Symposium*, March 2007.
16. TTTech Computertechnik AG. http://www.tttech.com.
17. H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*, volume 395 of *The Springer International Series in Engineering and Computer Science*. Springer, 1997.
18. D. McClure. The HMI challenge—balancing safety with functionality. Technical Report 37, SBD, September 2006.
19. EOBD. Directive 98/69/EC of the European parliament and of the council. *Official Journal of the European Communities*, October 1998.
20. ISO 9141. *Road Vehicles—Diagnostic Systems*. International Organization for Standardization, 1989.
21. ISO 14230. *Road Vehicles—Diagnostic Systems—Keyword Protocol 2000*. International Organization for Standardization, 1999.
22. ISO 15765. *Road Vehicles—Diagnostics on Controller Area Networks (CAN)*. International Organization for Standardization, 2004.
23. Society of Automotive Engineers. J2056/1 Class C Applications Requirements Classifications. In *SAE Handbook*, Vol. 1, 1994.
24. B. Hedenetz and R. Belschner. Brake-by-wire without mechanical backup by using a TTP-Communication Network. Technical report, SAE—Society of Automotive Engineers, Detroit, MI, 1998.
25. N. Navet, Y.-Q. Song, F. Simonot-Lion, and C. Wilwert. Trends in automotive communication systems, special issue on industrial communications systems. *Proceedings of the IEEE*, 93(6):1204–1223, 2005.
26. Society of Automotive Engineers. J2056/2 Survey of Known Protocols. In *SAE Handbook*, Vol. 2, 1994.
27. OSEK-VDX. http://www.osek-vdx.org.
28. J. Liu. *Real-Time Systems*. Prentice Hall, Englewood Cliffs, NJ, 1997.
29. L. Sha, R. Rajkumar, and J. Lehoczky. Priority inheritance protocols: An approach to real-time synchronisation. *IEEE Transactions on Computer*, 39(9):1175–1185, 1990.
30. MODISTARC. http://www.osek-vdx.org.

31. OpenOSEK. http://www.openosek.org/.
32. J.L. Bechennec, M. Briday, S. Faucou, and Y. Trinquet. Trampoline: An open source implementation of the OSEK/VDX RTOS. In *11th IEEE International Conference on Emerging Technologies and Factory Automation—ETFA'06*, Prague, September 2006.
33. RUBUS. Arcticus Systems AB. http://www.arcticus.se.
34. U. Freund and A. Burst. Model-based design of ECU software: A component-based approach. In *OMER LNI, Lecture Notes of Informatics, GI Series*, October 2002.
35. B. Selic, G. Gullekson, and P.T. Ward. *ROOM: Real-Time Object Oriented Modeling*. John Wiley, New York, 1994.
36. F. Simonot-Lion and J.P. Elloy. An architecture description language for in-vehicle embedded system development. In *15th Triennial World Congress of the International Federation of Automatic Control—B'02*, Barcelona, Spain, July 2002.
37. A. Rajnak. *Volcano—Enabling Correctness by Design*. CRC Press, Taylor & Francis, Boca Raton, FL, 2005.
38. A. Rajnak, K. Tindell, and L. Casparsson. Volcano communications concept—Technical report. Technical report, Volcano Communications Technologies AB, 1998.
39. H. Fennel, S. Bunzel, H. Heinecke, J. Bielefeld, S. Fürstand, K.P. Schnelle, W. Grote, N. Maldenerand, T. Weber, F. Wohlgemuth, J. Ruh, L. Lundh, T. Sandén, P. Heitkämper, R. Rimkus, J. Leflour, A. Gilberg, U. Virnich, S. Voget, K. Nishikawa, K. Kajio, K. Lange, T. Scharnhorst, and B. Kunkel. Achievements and exploitation of the AUTOSAR development partnership. In *Convergence 2006*, Detroit, MI, October 2006.
40. N. Medvidovic and R. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26(1):70–93, 2000.
41. S. Vestal. Metah support for real-time multi-processor avionics. In *Workshop on Parallel and Distributed Real-Time Systems, WPDRTS '97*, Geneva, Swiss, April 1997. IEEE Computer Society.
42. P.H. Feiler, B. Lewis, and S. Vestal. The SAE avionics architecture description language (AADL) standard. In *9th IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2003*, Washington, DC, May 2003.
43. U. Freund, O. Gurrieri, J. Küster, H. Lönn, J. Migge, M.O. Reiser, T. Wierczoch, and M. Weber. An architecture description language for developing automotive ECU-software. In *International Conference on Systems Engineering, INCOSE 2004*, Toulouse, France, June 2004.
44. ATESST. Advancing traffic efficiency and safety through software technology. EC IST FP6 project, http://www.atesst.org.
45. V. Debruyne, F. Simonot Lion, and Y. Trinquet. EAST-ADL—an architecture description language—validation and verification aspects. In P. Dissaux, M. Filali, P. Michel, and F. Vernadat, editors, *Architecture Description Language*. Kluwer Academic Publishers, 2004, pp. 181–196.
46. V. Debruyne, F. Simonot-Lion, and Y. Trinquet. EAST ADL, an architecture description language—validation and verification aspects. In *IFIP World Computer Congress 2004—Workshop on Architecture Description Languages*, Vol. 176 of *IFIP Book Series*, Toulouse, France, September 2004. Springer, pp. 181–196.

47. Y. Papadopoulos and J.A. McDermid. The potential for a generic approach to certification of safety-critical systems in the transportation sector. *Journal of Reliability Engineering and System Safety*, 63:47–66, 1999.
48. SAE International. Certification Considerations for Highly-Integrated or Complex Aircraft Systems. International Standard, November 1996.
49. RTCA DO-178B. Software Considerations in Airbone Systems and Equipment Certification. International Standard. Radio Technical Commission for Aeronautics, 1994.
50. EN50128. *Railway Applications—Software for Railway Control and Protection Systems*. International Standard, CENELEC, 2001.
51. P.H. Jesty, K.M. Hobley, R. Evans, and I. Kendall. Safety Analysis of Vehicle-Based Systems. In *Eighth Safety-Critical Systems Symposium*, Southampton, United Kingdom, 2000. Springer.
52. IEC. IEC 61508-1, Functional Safety of Electrical/Electronic/ Programmable Safety-related Systems—Part 1: General Requirements, IEC/SC65A. International Standard, 1998.
53. ISO WD 26262. Automotive Standards Committee of the German Institute for Standardization: Road Vehicles—Functional Safety. Preparatory Working Draft.
54. M. Findeis and I. Pabst. Functional safety in the automotive industry, process and methods. In *VDA Alternative Refrigerant Winter Meeting*, Saalfelden, Austria, February 2006.

2

Application of the AUTOSAR Standard

Stefan Voget
Continental Automotive GmbH

Michael Golm
Siemens AG

Bernard Sanchez
Continental Automotive GmbH

Friedhelm Stappert
Continental Automotive GmbH

2.1 Motivation

Today, development of electronic control units (ECUs) is characterized by several driving factors:

- Demands for more services, security, economy, and comfort
- Increasing complexity due to more ECUs and the growth in sharing software and functionality [1–5]
- More diversity of ECU hardware and networks (controller area network [CAN], local interconnect network [LIN], FlexRay, MOST, etc.)

As a result of these driving factors, communication between ECUs is increasing. Unfortunately, the ECU networks are oriented for a distribution of automotive functions which, despite the fact that it has been evolving for some time, is still not structured with respect to the newest technologies. The ECUs are grouped into several subdomains (Chapter 1), for example, power train, body, telematic, chassis, etc. Before AUTomotive Open System ARchitecture (AUTOSAR), the networks of ECUs were neither standardized in accordance with their interfaces across these subdomain borders nor developed with respect to the interrelationships between the nodes of the network.

Comparable statements can be made for the development processes. The software development processes for different ECUs evolved according to the individual history of the subdomains, and were quite divergent for a long time. In the automotive industry, most of the widespread system development processes assign functional requirements to software and hardware components on a one-to-one basis.

2.1.1 Shortcomings in Former Software Structures

The increasing total share of software resulted in high complexity and high costs. This became more critical with nonstandardized development processes and inadequate networks. In addition, the incorporation of third-party software made the collaboration between companies even more complex.

An appropriate level of abstraction in the software architecture modeling and appropriate integration concepts were still missing. The architectures did not reflect the effects of quality requirements. As a consequence, these often remained vague and unexplored. The architectures evolving with a single solution development strategy did not represent long-term solutions.

To further complicate matters, a lot of the functionalities are distributed over several ECUs, for example, the software that controls the lights of the indicator functionality is distributed over up to eight ECUs in high-end vehicles. Moreover, some of the future functionalities are not realizable with a loose side-by-side of the ECUs, for example, drive-by-wire will need a very close and safe interlocking of ECUs across different domains [6]. The traditional split of automotive functions will require more and more interconnectivity with the new upcoming functionalities.

2.1.2 Setting up AUTOSAR

With respect to this background, the leading automobile companies and their first-tier suppliers formed a partnership in 2003. This partnership has established an industry-wide standard for the automobile electronic, AUTOSAR, which is headed by the following 10 "core partners": BMW Group, Bosch, Continental, DaimlerChrysler, Ford Motor Company, General Motors, PSA Peugeot Citroën, Siemens VDO, Toyota Motor Corporation, and Volkswagen. The first phase of AUTOSAR started in 2003 and ended in 2006. During this phase, 52 "premium members," companies of the suppliers, and software and semiconductor industries joined the development of this standard and made major contributions in the consortium. In addition to these premium members, there are "associate members," "development members," and "attendees," whose roles in AUTOSAR varied with respect to their contribution and exploitation (www.autosar.org).

The first phase of AUTOSAR finished at the end of 2006, and the first AUTOSAR products were made available on the market.

The members of AUTOSAR agree that AUTOSAR makes it possible to control the complexity of the electrical and electronic components, together with an increase in quality and profitability. The future of automotive engineering is in these modular and scalable AUTOSAR software architectures.

2.1.3 Main Objectives of AUTOSAR

The main principle of AUTOSAR is "cooperate on standards, compete on implementation." Thus, the members established a set of main objectives, which also needed to be standardized because they were not considered primary factors for competitiveness [7–10].

- Consideration of availability and safety requirements
- Redundancy activation
- Scalability to different vehicle and platform variants
- Implementation and standardization of basic functions as an industry-wide "standard core" solution
- Transferability of functions from one ECU to another within the network of ECUs
- Integration of functional modules from multiple suppliers
- Maintainability throughout the whole product life cycle
- Increased use of commercial-off-the-shelf (COTS) hardware
- Software updates and upgrades over vehicle lifetime

On all levels of modeling, an object-oriented eXtensible Markup Language (XML) class model, specified in UML 2.0, is used. Starting with the explanation of the description language you have to go down (via the metamodel-level) to a concrete user model, which can be realized as a concrete XML file.

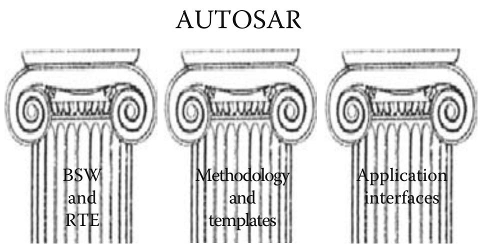


FIGURE 2.1 The three pillars of AUTOSAR.

The most important consequence of the stringent component-based approach of AUTOSAR, concerning the development process, is a separation of application development from the lower levels of the integration development (basic software [BSW]). The separator between these two parts is the AUTOSAR runtime environment (RTE), which concretely realizes the concept of a virtual functional bus (VFB) as an abstracting communication principle. The idea of this concept is that an application does not need to know the concrete paths from data and signals below RTE when two applications communicate together.

This simplifies matters for the application developer. He or she, de facto, needs no further knowledge about concrete architectures, even if a deeper knowledge about the interfaces, made available to an application on top of the RTE, is still indispensable (Figure 2.1).

2.1.4 Working Methods in AUTOSAR

AUTOSAR was set up as a partnership to define an industry-wide standard. The consortium tries to use as many existing solutions as possible, trying not to invent everything newly. In most cases, standardization means choosing one option over several alternatives. Of course, as different solutions are already implemented by the companies, this means agreeing on compromises. If possible, existing standards are taken as they are, for example, CAN, LIN, OSEK, etc. In other cases, if it is not possible to agree directly on an existing solution, cooperation with other standardization groups is established, for example, with FlexRay consortium, ASAM-FIBEX,[*] MOST, etc.

To be able to standardize one needs a minimal stability and some common understanding of the issue one is dealing with. Therefore, AUTOSAR was not started without any preparation. Several research projects were carried out in advance, in particular, the projects ITEA-EAST/EEA[†] and AEE,[‡] which can be mentioned here.

[*] Field bus exchange format from Association for Standardization of Automation and Measuring Systems and Measuring Systems.

[†] An European-funded project for embedded electronic architecture (EEA).

[‡] A French funding project; a predecessor of ITEA-EST/EEA.

2.2 Mainstay of AUTOSAR: AUTOSAR Architecture

2.2.1 AUTOSAR Concept

To fulfill the requirements discussed in the previous chapter and in Ref. [11], the AUTOSAR consortium defined a new development methodology for automotive software and software architecture. The development methodology is focused on a model-driven development style. The software architecture, as well as the ECU hardware and the network topology, are modeled in a formal way, which is defined in a metamodel that supports the software development process from architecture up to integration. All available modeling elements are specified by the "AUTOSAR metamodel" [12]. The metamodel is defined according to the rules of the OMG Meta Object Facility [13].

The envisioned development methodology starts by defining the software architecture. An exemplary software architecture can be seen in Figure 2.2.

The boxes represent software "components." At the perimeter of the boxes the communication "ports" of the software components are shown. A port with an inward pointing triangle is a "required port." A port with an outward pointing triangle is a "provided port." Required ports are the data receivers in a data flow-oriented communication, whereas provided ports are the senders. A provided port can be connected with one or more required ports of other software components. To be able to connect ports, the interfaces of the two ports must be compatible.

There are two types of interfaces, "sender/receiver interfaces" and "client/server interfaces." A sender/receiver interface supports message-based communication, while a client/server interface supports a remote-procedure-call style of communication.

A sender/receiver interface consists of a list of "data elements." Each data element has a name and a data type.

The client/server interface consists of a list of "operations." Each operation has a signature, consisting of a name and a list of "parameters." Each parameter is described

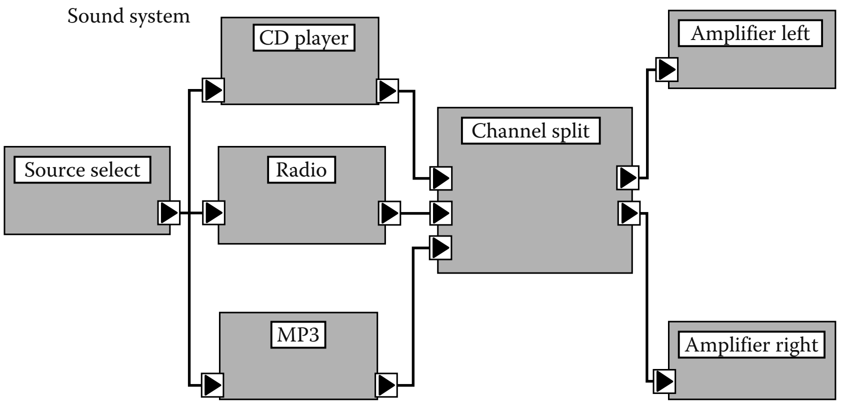


FIGURE 2.2 Example for software components and connectors.

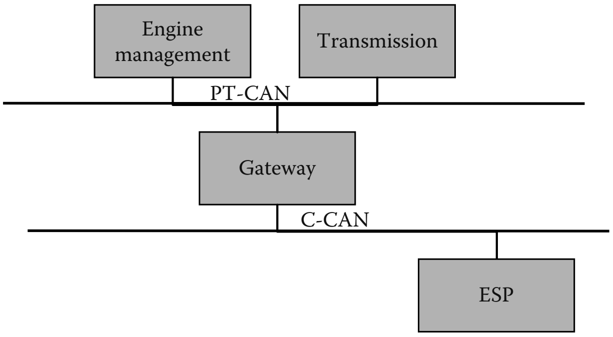


FIGURE 2.3 Examplary network topology.

by a name, a type, and a direction, which can be either in, out, or in–out. The details of all software components related to modeling elements are described in Ref. [14].

The software architecture is defined without consideration of the hardware on which the software components will run on later. This means that two software components might run on the same ECU or on different ECUs. The communication between the components is then either an intra-ECU communication or an inter-ECU communication. To abstract from this difference, AUTOSAR introduces the VFB. The VFB can be seen as a software bus to which all components are attached. The VFB software bus is based on ideas similar to common object request broker architecture (CORBA) [15].

The hardware architecture is modeled in parallel to the definition of the software architecture. AUTOSAR allows for modeling the topology of a vehicle network as well as the hardware of an ECU. An example of this topology can be seen in Figure 2.3.

The example shows two ECUs connected to a power train CAN (PT-CAN) and one ECU connected to a chassis CAN (C-CAN). The two CAN busses are connected through a gateway.

Once the software architecture and the network topology are defined, the software entities can be mapped to the hardware entities. The software component template standardizes the format for describing the software entities and is a very important part of the AUTOSAR metamodel. It defines how the software architecture is specified.

2.2.2 Layered Software Architecture

AUTOSAR defines a software architecture for ECUs. This architecture is defined in a layered style. The lowest layer of the architecture, the microcontroller abstraction layer (MCAL), is responsible for providing abstractions of typical devices. The MCAL modules could be considered as device drivers. There are four groups of MCAL modules: microcontroller drivers, memory drivers, communication drivers, and input/output (I/O) drivers. The communication drivers include drivers for CAN, LIN, and FlexRay. The I/O drivers include drivers for pulse width modulation (PWM), analog-to-digital converter (ADC), and digital I/O (DIO). Above the MCAL there is