VECTOR

Technical Papers
on the Development of Embedded Electronics

7th Edition | English

Vector – Automotive. Embedded. Engineering.

Dear Reader,

There's no better medium for having your challenges taken up than qualified articles in professional journals. Presenting current problems and their solutions is just as important an issue to us as discussing technical principles and standards. Experts at Vector address your questions and concerns and develop real-world solutions from them.

In the seventh edition collection of Vector technical articles, we cover current issues from AUTOSAR Adaptive, the tense field of security and safety and new developments to automated driving and much more in 27 new articles.

We hope you're looking forward to some very inspiring articles!

Stuttgart, July 2018 Dr. Thomas Beck

Status: July 2018
Responsible for content: Vector Informatik GmbH, Stuttgart

All mentioned product names are either registered or nonregistered brand names of their respective owners.
Continual global availability of all products or services is not guaranteed.
Errors and omissions reserved.

Reprint only permitted with written approval of Vector Informatik GmbH, Stuttgart.

Please note: Some of the web links given in this article may be obsolete.
They were valid at the time of first publication of the technical article.

Illustration & Design: Cirek Mediendesign, Stuttgart, Germany

Contents

Company Profile

Vector – The Company

Serial Bus Systems

Data Communication in the Automobil Part 1 1/0
Architecture, Tasks, and Advantages of Serial Bus Systems

Serial Bus Systems – CAN

Data Communication in the Automobil Part 2 – CAN 1/6

CAN Gets Even Better 1/12
Ways to Transition from Classic CAN to the Improved CAN FD

CAN FD: Fast Measurement and Reprogramming 1/16

▶ **Eye Diagram Analysis for CAN FD** 1/20
Fast identification of poor signal quality

▶ **CAN Tools – Then and Now** 1/24

Serial Bus Systems – LIN

Data Communication in the Automobil Part 3 – LIN 1/28

Serial Bus Systems – FlexRay

Data Communication in the Automobil Part 4 – FlexRay 1/34

Serial Bus Systems – Automotive Ethernet

Wireless Analysis in a Multi-Protocol CAN Environment 1/40

Acquiring Bus Data Wirelessly from Multiple Vehicles 1/44
Top-notch Precision Despite the "Air Interface"

IP and Ethernet in Motor Vehicles 1/50
Challenges for the Development Tool, Illustrated by Today's Applications

Challenge of Ethernet Use in the Automobile 1/56
Flexible Interfaces and Software Tools Simplify ECU Development

New Perspectives on Remaining Bus Simulation for Networks with SOME/IP 1/62
Validating Automotive IP Network Elements

AUTOSAR Learns Ethernet 1/66

New Communication Paradigms in Automotive Networking 1/70
Ethernet and CAN FD are the New Trailblazers

Full Transparency with Automotive Ethernet 1/76
Finally Seeing what is Really Happening

▶ **Time Synchronization in Automotive Ethernet Networks** 1/82
Balancing Act Between AUTOSAR, IEEE, and TSN

▶ **The Future with SOA, POSIX, TSN** 1/88
Automotive Ethernet: Trends and Challenges

▶ **From Signal to Service** 1/94
Challenges for the Development of AUTOSAR Adaptive Applications

Serial Bus Systems – ITS-G5

Testing Car2x Applications 1/100
Requirements for Test Tools Based on Example of the Road Works Warning

Car2x – From Research to Product Development 1/104
How Automotive OEMs and Suppliers are Successfully Completing Production Car2x Projects

Serial Bus Systems – K-Line

K-Line: Flexible Solutions for a Classic Protocol 1/110
From Precise Monitoring to Data Manipulation for Generic Byte Protocols

Serial Bus Systems – MOST

Data Communication in the Automobil Part 5 – MOST 1/114

Development of distributed Systems

▶ **Controlling Complexity in the Vehicle Wiring System** 2/0
Model-based Wiring Harness Development

▶ **Eye On the Whole System** 2/4
Why Consistent Implementation of the AUTOSAR System View is Worth it

▶ **Design and Optimization of E/E Architectures** 2/10
PREEvision as Model-Based Tool for the Next Generation

▶ **E/E Development for Future Vehicle Innovations** 2/16
An Integrated, Model-based Approach

▶ **Integrated Development of a Complete E/E Architecture** 2/26
Reaching Goals Reliably Using a Model-Based Approach

▶ **Service-oriented Architectures and Ethernet in Vehicles** 2/30
Towards Data Centers on Wheels with Model-based Methods

▶ **Model-based Solutions for Medical Systems** 2/36
An Integrated Approach for the E/E Development Process

▶ **Integrated Requirement Management** 2/42
The Next Stage in the Evolution of Requirement Engineering

Testing

▶ **Current Approaches for ECU Testing** 3/0
What Really Matters

Quickly Converting Test Benches Worldwide in Record Time 3/4
Record-Breaking

▶ **Network Tests for Everyone** 3/8
With Minimum Configuration Effort for Maximum Testing Depth

Case Study: Fraunhofer 3/13

Tips and Tricks for the Use of CAPL 3/14
Part 1: CAPL Basics

Tips and Tricks for the Use of CAPL 3/16
Part 2: Effectively Apply CAPL

Tips and Tricks for the Use of CAPL 3/18
Part 3: CAPL for Advanced Users

Quick Paths to a Comprehensive Remaining Bus Simulation 3/20
Creative Virtual Networks Without Programming Expertise

Hardware Simulation for the Challenging Unimog Tire Pressure Control System 3/24
Time Savings and New Options by ECU Tests on the Model

ECU Testing with XCP Support 3/28
A Look Behind the Scenes

Case Study: GETRAG 3/31

Flexible Test Systems for Efficient ECU Testing 3/32
Functional Testing with Error Simulation at the Developer's Bench

▶ **Functional GUI Testing of In-vehicle Infotainment Systems in Virtual and Real Environments** 3/36

▶ **What Kinds of Interfaces are Available?** 3/40
Testing in Heterogeneous Tool Landscapes

Case Study: Moore Industries 3/44

Case Study: SsangYong Motor Company 3/45

Case Study: Autec 3/46

Easy Access to Bus Analysis 3/48

▶ These technical articles have been added since the last Pressbook edition.

Contents

Vehicle Diagnostics

Automatic Validation of Diagnostic Services by Use of a Diagnostic Integration and Validation Assistant at Opel 4/0
Automated Testing of Diagnostic Implementations Based on the Example of the Opel Insignia.

Automated Data-driven Validation of the Diagnostic Implementation 4/8

▶ **Automatic Diagnostic Validation is not Rocket Science** 4/12
... Just a Matter of Consequent Exploitation of Existing Possibilities

The Standard Mix Does It 4/16
Part 1: Diagnostics with AUTOSAR

The Standard Mix Does It 4/20
Part 2: ODX in the AUTOSAR Development Process

Diagnostic Tools for WWH-OBD 4/24
Implementation of the New Requirements for OEMs and Suppliers

Case Study: KTM 4/27

From Diagnostic Requirements to Communication 4/28
Standardization is the Trend in the Development of Automotive Electronics

Diagnostics from a Distance 4/32
Interactive Diagnostics for Vehicles Worldwide

ECU Calibration

▶ **Data Recording for ADAS Development** 5/0
Scalable Recording of Sensor and ECU Data

▶ **All Puzzle Parts for OBD Documentation** 5/6
Integrated Process for Generating the Legally Required OBD Documentation Faster and Without Errors

Calibrating ECUs 5/10
Trends and Effects on Development Methods and Tools

Analyze Large Quantities of Measurement Data Rationally and Flexibly 5/14

Verification of Driver Assistance Systems in the Vehicle and in the Laboratory 5/18

Calibration Data Management: A Puzzle Game No More 5/22

Case Study: BMW 5/27

Riding on the Razor's Edge 5/28
Optimal Parameterization of an Engine Controller for Drag Racing

Case Study: HOERBIGER 5/31

AUTOSAR

AUTOSAR ECU Development Process Using DaVinci and MICROSAR from Vector 6/0

Case Study: vSBC 6/7

AUTOSAR Methodology in Practice 6/8

On-Board Diagnostics Meets AUTOSAR 6/12

Case Study: AUTOSAR 6/13

AUTOSAR in Heavy-Duty Vehicles 6/14
Integration of J1939 in AUTOSAR

Case Study: PATAC 6/17

AUTOSAR – Equipped for Everything? 6/18

User-friendly Configuration of AUTOSAR ECUs with Specialized Software Tools 6/22

AUTOSAR Goes Multi-Core – the Safe Way 6/28

AUTOSAR-Compliant Development of an In-car Radio Product Using MICROSAR for the European Market 6/32

High-Rate Task Scheduling within AUTOSAR 6/36

New Opportunities With AUTOSAR 6/40

Functional Safety

SilentBSW – Silent AUTOSAR Basic Software for Safety-Related ECUs 7/0
Coexistence of Safety-Related and Non-Safety-Related Software in one ECU by Protecting Memory Areas

Practical Implementation of Mixed-ASIL Systems 7/4
A Certified Operating System Simplifies the Development of Safety-Related Software

Seamless Implementation of ECU Software Based on ISO 26262 7/8

Is this what the Future Will Look Like? 7/14
Implementing Fault Tolerant System Architectures with AUTOSAR Basic Software

▶ **Safety and Performance with ASIL D AUTOSAR Basic Software** 7/20

▶ **Has Functional Safety Ceased to Be a Topic of Interest?** 7/24
Statement of Dr. Heling, Leader of the "Software ISO 26262" Working Group within ZVEI

Automotive Cyber Security

Secure Communication for CAN FD 8/0

Cyber Security – Challenges and Practical Guidance 8/4
Success Factors for Security Engineering

Cyber Security for the Automotive Industry 8/12
Practical Experiences on the Application of Cyber Security

▶ **Risk-Oriented Automotive Cybersecurity** 8/18
Industry Experiences and Case Study

E-Mobility

▶ **E-Mobility Ready for the Mass Market?** 9/0
Trends, Progress and Experiences

▶ **Commercial Vehicles and Electromobility** 9/4
Charging Ahead

▶ **Inductive Charging** 9/8
From Evaluation to Standardized E-Mobility

Inductive Charging Gives Trigger to Future of E-Mobility 9/12
ISO/IEC-15118 Standardization of Wireless Power Transfer

Smart Testing of Smart Charging 9/16
Consistent Test Case Coverage for Electric Mobility

Case Study: ZSW 9/21

Smart Charging 9/22
A Key to Successful Electric Mobility

ECU Testing for Electric and Hybrid Vehicles 9/26
Intelligent Measuring the Dynamic Power Consumption

New Bus Interfaces for Electric/Hybrid Vehicle Development 9/30
Realtime Performance Enables Flexible Use in the EV/HEV Development Field

▶ These technical articles have been added since the last Pressbook edition.

Contents

Open Networks – SAE J1939

Networking Heavy-Duty Vehicles Based on SAE J1939 10/0
From Parameter Group to Plug-and-Play Application

Quo Vadis SAE J1939 Standardization 10/6

Integration of J1939 Systems in Practice 10/12

CAN and J1939 Under Extreme Duty Conditions 10/16
Vehicle Electronics Guarantees Functionality in Cold, Ice and Snow

Current Trends in Network Development for Heavy-Duty Vehicles 10/22
Factors of Success in Electronic Development

Open Networks – ISOBUS

Automatic Interoperability Tests for ISO11783 Systems 10/26
Universal Test Solution Assures Compatibility

Forging New Pathways in Testing ISOBUS Task Controllers 10/30
Simulations Replace Inflexible and Time-Intensive Test Methods

Better Test Quality by Automation 10/36
Automated HIL Test System Ensures ISOBUS Functionality of Agricultural Machines

Development of a Cooperative Tractor-Implement Combination 10/42

Open Networks – CANopen

Prototyping and Testing CANopen Systems 10/48
Reaching Goals Rapidly with Minimal Effort

Automatic Testing of CANopen Devices 10/52

Consulting

► **Connectivity** 11/0
Practical Experiences with Smart Systems and Services

Lean Requirements Engineering 11/6

Improving Engineering Efficiency with PLM/ALM 11/10

► These technical articles have been added since the last Pressbook edition.

Vector – the Company

Information at a Glance

Vector Informatik is the leading manufacturer of software tools and embedded components for the development of electronic systems and their networking with many different systems from CAN to Automotive Ethernet.
Vector has been a partner of automotive manufacturers and suppliers and related industries since 1988. Vector tools and services provide engineers with the decisive advantage to make a challenging and highly complex subject area as simple and manageable as possible. Vector employees work on electronic innovations for the automotive industry every day. Worldwide customers in the automotive, commercial vehicles, aerospace, transportation, and control technology industries rely on the solutions and products of the independent Vector Group for the development of technologies for future mobility.

Reliable Partner with Quality

Our goal is excellence in all areas! Vector processes are worldwide regularly assessed and certified, and they comply with current standards:
> Quality Management - ISO 9001
> Automotive SPICE
> Functional Safety - ISO 26262

Employees
> 2,000
worldwide

Subsidiaries
24 locations in
12 countries

Customers
> 7,500
companies
in 72 countries

Associations
participation in
15 committees

The Vector Portfolio

Development of Distributed Systems
Tools and services to support you in designing and developing networks and networked ECUs - especially for simulation, analysis and testing of network communication.
Main product: PREEvision

Testing
Tools and services that provide a scalable and re-usable solution from pure SIL simulations to HIL testing with functional acceptance tests.
Main products: CANoe, CANalyzer, vTESTstudio, VT System, Data Logger, VectorCAST

Diagnostics
Tools and services to describe, implement, validate and test the diagnostic functionalities that are required to run diagnostic services on an ECU.
Main products: CANdelaStudio, Indigo, vFlash, DiVa

ECU Calibration
Tools to access the ECU at run-time for acquisition and modification of measurement data and parameters to optimize ECU algorithms.
Main products: CANape, VX1000, vCDM, vSignalyzer, vADASdeveloper

Embedded Software and Systems
Software components for the ECU communication and for re-programming (flashing) via CAN, LIN, FlexRay, MOST and Ethernet, real-time operating systems, AUTOSAR basic software, diagnostic software, project work and services.
Main products: MICROSAR, Flash Bootloader, VC ECU, customer projects

Consulting
Vector offers consulting for the optimization of your technical product development, the associated business processes and tools, as well as for the successful implementation of change.
Our offer: consulting services, engineering services

Data Communication in the Automobile – Part 1:

Architecture, Tasks, and Advantages of Serial Bus Systems

History

The first vehicles powered by gasoline engines already had electrical components, such as ignition coils, contact breakers, and spark plugs. These were quickly followed by other electrical devices such as headlights, brake lights, direction signals, interior lights, and windshield wipers. The introduction of components for entertainment and information, such as radios and record players and more recently cassette and CD players, meant that automobiles soon contained electronic components as well. Since the 1970s, electronics came along that improved or enhanced the functions of the vehicle itself or made driving easier. The next stage in the 1980s was directed at driving safety, including ABS and airbags, and expansion of comfort features such as air conditioning, mirror dimming and adjusting, power windows, and cruise control.

Growing competition brought about by increasing globalization ensured more and more innovation. Automobile manufacturers met this multifaceted challenge with electronics. The development of electronic components grew so rapidly in the 1990s that it is beyond the scope of this article to list all of the individual stages. Nowadays, the focus is on environmental friendliness and on the ability of automobiles to participate safely in road traffic as in self-driving cars, also knows as autonomous. Vehicles are also starting to be connected with one another and with road infrastructure devices as well as with the Internet via WLAN.

However, none of these functions would be possible in the automobile without data exchange between electronic components. And it is exactly this need for data exchange that has proven to be and still remains the real challenge. Initially, a dedicated cable was used for the transmission of each separate information message **(Figure 1)**. As the amount of information grew, however, the cable harnesses became so large that their weight and multitude of connectors was problematic, not to mention the logistic challenges that arose during manufacturing, maintenance, and further development.

Figure 1: Point-to-Point-Connections

What is a "Bus"?

A pioneering solution for all of these problems was the introduction of the so-called bus. The word "bus" comes from the Latin word "omnibus", which simply means "for all". The many individual cables were thus replaced by a single cable that is shared by all information of all electronic control units (ECUs) **(Figure 2)**.

However, it was then necessary to find ways to organize the timely transmission of this multitude of information over common wires. Different technologies arose, which we refer to as serial bus systems.

Before we delve into the specific characteristics of the individual bus systems in the subsequent articles in this series, we will start by explaining the technical fundamentals of the serial bus systems that are used in modern motor vehicles and then compare their underlying concepts.

A common characteristic of all buses is that each connected ECU shares a single input and output and, unlike in networks, information does not have to pass through the ECUs. Thus, when one ECU sends information on a bus, all other ECUs receive the information at almost the same time. The condition "almost" is necessary solely on account of the signal transit time on copper, which is approximately 5 ns/m.

Figure 2: Bus Networking

Addresses Just Like the Post Office

For smooth information exchange, the data to be sent must be clearly allocated to its senders and receivers. We call this addressing. A general distinction is made between sender-selective and receiver-selective addressing. We are familiar with both of these types of addressing from our mailbox. With sender-selective addressing, the sender defines the desired receiver using a unique destination address. This corresponds to a standard letter with a destination address and return address.

In contrast to this, receiver-selective addressing identifies the information to be sent and not the ECUs. For this reason we talk about identifiers here and not addresses. We recognize deliveries like this in our mailboxes from the name (identifier) of the sending supermarket or home improvement store chain. Each receiver decides whether or not he uses the received information.

The sender pairs information and address respectively identifier thus allowing the receiver to recognize the unit. In transmission technology we refer to this unit as a "frame" because the addresses and information are framed by a start identification and end identification, which mainly serve to ensure error-free transmission between the sender and receiver. It's also referred to as a message or packet.

The Protection of Data

The most important tasks of a serial bus system include timely data transmission at fast enough rates for the particular technical application and, above all, the protection of data during transmission. The use of a serial bus system in the automobile depends in large part on the degree of protection needed and the amount of data per unit of time that must be transmitted.

The protection of a serial bus system depends on how well the system prevents errors during data transmission and how well it detects remaining data corruption. The residual error probability is one measure for this protection. This is the product of probability A that the data to be transmitted are corrupted and probability B that corrupted data remain undetected. The lower the residual error probability, the more the data transmission is protected.

There are many different causes of data corruption. The most powerful are sparks from ignition plugs and electric motors. Other galvanic, capacitive, or inductive interactions and electromagnetic fields also play a role. Even reflections at the end of the bus cable are an internal cause of data errors on a serial bus. The more effective a bus is at eliminating or preventing these causes, the better the data transmission is protected.

A few important data protection measures are sufficient. These include shielding the transmission medium (cable or wire) and all electrical and electronic components. Alternatively the principle of differential signal transmission **(Figure 3)**

Figure 3: Differential transmission on a twisted pair

achieves a high degree of transmission protection without the need for real physical shielding. Shielding would be too expensive and does not adequately meet the requirements for flexibility and heat resistance. Using differential signal transmission, information is serially transmitted over unshielded twisted pair with the help of voltage levels that are exactly symmetrically opposed. If the voltage on one wire increases by X volts, the voltage on the other wire decreases by X volts. As a result, the electric fields in the cables are exactly opposed and the resulting magnetic fields around the wires cancel each other almost completely. Such a system hardly emits any noise. The voltage difference between the two wires represents the signal from which external noise is subtracted.

Sufficient clearance between the data transmission and power supply cables and between electrical and electronic components is helpful. It is also essential to limit the data transmission speed as well as the number and steepness of the data signal edges and to terminate the two bus ends with the characteristic impedance of the transmission medium to prevent reflections.

Nevertheless, transmission errors can never be fully eliminated, which is why error detection measures are needed. The checksum calculation method is most commonly used. With this method, the sender uses a defined algorithm to calculate a checksum from the data to be transmitted and includes it at the end of the message. A receiver can compare this checksum with the one that it has calculated from the received data. If the two checksums do not match, there is an error. The more sophisticated the algorithm and the longer the checksum, the better the data error detection capability. A checksum must not be too long, however, because every message in turn becomes longer and less data can be transmitted on the bus.

If an error is detected, the question arises as to how to correct it. Errors can be corrected based on the checksum contained in a message. However, this requires longer checksums and, in particular, immense computing capacity in the receiving ECUs. This correction method is not used in automobiles. Instead, the faulty message is discarded and a new transmission is requested.

Figure 4: Controlled and random bus access

Is Information Transmitted Fast Enough?

A bus system is regarded as capable of transmission in real-time [1], or real-time capable, if it can guarantee sufficiently fast transmission of all data that accumulates for an application. The essential factors for this are the number and size of messages, the available transmission speed (also referred to as bandwidth), and the bus access method of the ECUs. For the latter, a distinction is made between controlled bus access and random bus access **(Figure 4)**.

With controlled bus access, the bus access right of an ECU is already defined before its bus access. Such systems are called deterministic systems because it can be exactly determined or calculated when a particular ECU transmits which data. Deterministic behavior is an important precondition for achievement of real-time. Because the entire communication sequence runs according to schedule and can hardly be influenced, bus systems with controlled bus access are characterized by a poor dynamic response.

Bus systems with uncontrolled bus access avoid this disadvantage. Each ECU has the right to transmit data at any time. This results in a very fast bus access but also poses the following risk: Depending on the density and length of the messages and the available bandwidth, there is a greater or lesser acute risk of collision. This is not an ideal basis for real-time.

If all ECUs monitor the bus continuously and send information only when the bus is available, the risk of collision is significantly reduced, but not fully reduced. This risk is eliminated altogether by introducing priorities for information, which can be recognized on the CAN bus with help of the identifier. However, even these bus access methods cannot guarantee timeliness (real-time). The reason is that, as a result of prioritization, there is a risk that messages with lower priority will be subject to long delays and no longer be received in a timely manner.

Figure 5: Architecture of serial bus systems

Architecture of Serial Bus Systems

Based on the reference model for data communication specified by the ISO (International Standardization Organization), the serial bus interface of an ECU in the automobile is generally distributed among two (communication) layers. The physical layer implements the physical bus connection including the physical signal transmission. Above that is the data link layer with its tasks including addressing, framing, bus access, synchronization, and error detection and correction **(Figure 5)**.

The physical bus connection is made with help of a transceiver. A communication controller takes over the tasks of the data link layer. If all ECUs on the bus use the same type of transceivers and communication controllers, the basic preconditions exist for smooth data exchange.

During serial communication, the sender's application delivers the data block to be sent together with addresses or identifiers to the communication controller. Check and synchronization information as well as start and end identification are added, so that a frame results. The transceiver now sends the frame on the bus.

Most buses in automobiles are in the form of a cable to which the ECUs are connected via short spur lines. This is called a line topology or bus topology. On the receiver side, the transceiver passes the frame to the communication controller, which checks the information, and if data is received correctly, forwards the data block to the application. For some tasks such as bus management, that is, the concerted putting to sleep and waking up of all ECUs, and the diagnosing and configuring of ECUs, the functions of the physical layer and the data link layers are not sufficient. The higher layers of the ISO reference model for higher communication protocols are then used in order to achieve the required communication functionality.

Cross-Vendor Bus Technologies

Intensified competition is ensuring more and more safety and comfort functions in automobiles. As a result, there is not only a continual increase in the number of electronic components in vehicles but also a significantly higher degree of networking and a rapid rise in data volumes since most new automobile functions rely on data exchange. The challenge arising from this is to keep the increasing complexity under control to ensure the continued quality and reliability of functions. For this purpose, the automobile industry has developed standards, such as "AUTOSAR" (AUTomotive Open System ARchitecture, **Figure 6**), which provide the necessary transparency on a system or function level. Cross-vendor communication standards such as the serial bus systems CAN [2], LIN [3], FlexRay [4], MOST [5], and Ethernet [6] ensure more transparency on the lower communication levels.

Figure 6: AUTomotive Open System ARchitecture

CAN (Controller Area Network) is used mainly in the drive and chassis areas and in the operation of the vehicle. LIN (Local Interconnect Network) is used to control simple comfort functions, such as the air conditioning, seats, mirrors, and windows. MOST (Media Oriented System Transport) has long served the infotainment area with transmission of video and audio signals that require large bandwidths. Also Ethernet is increasingly being used for this. Currently, Ethernet is mainly used for diagnostics including flashing. In the future, however, its main use will be in the driver assistance area, including the park assist and autonomous driving sub-areas. Finally, FlexRay should enable the most demanding communication in safety-critical applications such as electronically controlled steering and braking. However, lawmakers along with the automakers are finding it difficult to push forward here, by trying to avoid the risks in these areas that are highly critical to safety.

CAN was developed in the early 1980s by Robert Bosch GmbH and became an international standard (ISO 11898) in 1994. The founders of Vector played a central role in this development. LIN (ISO 17987), FlexRay (ISO 17458), and MOST came from cross-vendor organizations, namely the LIN Consortium, FlexRay Consortium, and MOST Cooperation. Vector [7] supports automobile manufacturers and suppliers in networking using CAN, LIN, FlexRay, MOST, and Ethernet with a consistent set of design and development tools as well as with software components and basic software for AUTOSAR ECUs. Advice, consulting services and tools for process management supplement the areas of application. A comprehensive training program that includes basic seminars for CAN, LIN, FlexRay, and Ethernet rounds out the services. Parts 2 to 5 of this series cover the details of the serial bus systems CAN, LIN, FlexRay, and MOST.

Closing Remarks About the Term "Real-Time"

The term "real-time" is often used loosely or imprecisely, because it is not very easy to grasp. Because I had to use it at the outset, I would like to clarify a few facts about this subject. Whoever wants to know more can draw on other sources. [1]

Part 9 of DIN 44300 (Information Processing), which has been replaced by DIN ISO/IEC 2382, defines real-time as follows: "Real-time refers to the operation of a computing system on which data processing programs are always operable such that processing results are available within a specified time span. The data may occur randomly or at predefined times, depending on the application."

For a real-time capable system, it is thus not enough to deliver a measurement or calculation result with the correct value. Rather, this also has to occur within a specified response time. If this is not the case, the system has failed. According to the theory of real-time systems, the required response time must be calculated for an application running on the system.

People often speak carelessly about "real-time" if a program runs without a noticeable delay. However this definition is not very accurate.

It is wrong to use "real-time" as a synonym for "very fast", because real-time systems even have to schedule no-load operations in order to also meet real-time requirements under high load.

A distinction is made between hard real-time and soft real-time. The distinguishing criterion is the different consequences of a violation of the real-time requirements.

> Hard real-time requirements: If the system exceeds the required response time, it has failed. Real-time systems must always supply the correct result within the required response time, and the user of a hard real-time system must be able to rely on this. (For example, engine control: if the requirements are violated, the engine sputters and even damage may occur.)

> Soft real-time requirements: Such systems typically meet the required response time but not always. Thus, for example, the required response time reaches only an average or satisfies a different statistical criterion. The time requirements are not absolutely strict, but rather are viewed as guidelines. Exceeding the required response time is not regarded as a failure. It can be exceeded frequently as long as it still falls within a tolerance range. Or, the response time can far exceed the required response time occasionally. (For example, a video conference: When response time requirements are violated, the image "jerks" but work can continue.)

Literature References:

[1] de.wikipedia.org/wiki/Echtzeit
[2] de.wikipedia.org/wiki/Controller_Area_Network
[3] de.wikipedia.org/wiki/Local_Interconnect_Network
[4] de.wikipedia.org/wiki/FlexRay
[5] www.mostcooperation.com
[6] de.wikipedia.org/wiki/BroadR-Reach
[7] www.vector.com

Ernst Christmann, Physicist, Mathematician
is a Senior Technical Trainer in the area of software tools for ECU testing and development and has worked for Vector Informatik GmbH in Stuttgart since 2004.

Data Communication in the Automobile – Part 2:

Reliable Data Exchange with CAN

As presented in part 1 of our article series, the increasingly complex electronic systems in automobiles are calling for a higher level of data exchange between the ECUs. In order to ensure this with sufficient reliability and speed, the CAN bus was developed. CAN stands for Controller Area Network. As the name implies, the CAN bus can link a larger area and reach a length up to multiple kilometers.

The CAN bus was developed by Bosch [1] and became a standard in 1993. It is currently available as ISO 11898 **(Figure 1)**. The standard is divided into multiple parts. The first part specifies the CAN protocol and covers all aspects of the data link layer (framing, addressing, bus access, data integrity) and the physical signal coding as part of the physical layer of the ISO 7498 reference model – the so-called OSI layer model (OSI: Open Systems Interconnection). The CAN controller was developed for handling the CAN protocol.

Parts 2 and 3 of ISO 11898 describe two versions of the physical layer, namely CAN high-speed and CAN low-speed. The latter is also often called fault-tolerant CAN because it continues functioning if one of its two wires breaks, although with diminished reliability. Parts 2 and 3 also cover the physical layer of the ISO reference model, including the physical bus connection, data rates, and the voltage levels on both CAN wires.

CAN uses differential signal transmission, which reduces noise sensitivity and requires two communication wires (CAN high and CAN low) that are terminated at both ends with characteristic impedance R_T of 120 Ω.

CAN high-speed is mainly used in drive and chassis applications. It is primarily implemented by the CAN high-speed transceiver, which supports a maximum data rate of 1 Mbit/s. The CAN low-speed physical layer has been used mainly in the convenience area. It is placed in doors, seats, and sunroofs where it is subject to bending that can cause wire breaks. Here the fault-tolerant CAN low-speed transceiver with a maximum data rate of 125 kbit/s is used. It can also be operated using a single wire. Although this type of CAN bus is rarely used.

Figure 1: CAN Standard

The CAN interface consists of a CAN controller and a CAN transceiver **(Figure 2)**.

The CAN controller handles the CAN protocol. The CAN transceiver connects the CAN controller physically to both of the CAN wires and measures or generates the voltage levels on these two wires.

What Has to Go from Where to Where?

CAN uses the receiver-selective form of addressing. The identifier (ID) indicates the content of the transmitted data and not the destination. A message can thus be received and evaluated by all ECUs on the bus (message distribution). The application of a receiving ECU decides whether it evaluates a message. It can even set an acceptance filter in its CAN controller when starting, which hides unneeded CAN messages in the message stream based on their identifiers. CAN offers two sizes of identifiers: 11 bit and 29 bit. The smaller identifier (standard format) is used in passenger cars. It provides 2048 different messages, while the larger identifier offers 536 870 912 messages. The latter is mainly required in commercial vehicles for CAN-based software protocols, such as SAE J1939, but is now found in passenger cars as well.

Receiver-selective addressing offers the following advantages:

> Cost savings through shared use of sensors by all ECUs on the bus.
> Easy implementation of distributed functions
> It allows different configurations without adaptation of hardware or software

Figure 2: Structure of a CAN ECU

Events Trigger the Transmission of Messages

If a newsworthy event happens in everyday life, it is communicated in newscasts. In the world of serial buses, the term "event" is also used to describe an occurrence that requires information to be transmitted. For fast communication of information, the ideal situation is for the underlying event to directly trigger transmission of the respective data. This is referred to as event-driven transmission.

The alternative would be to transmit information according to a prescribed schedule or time pattern. But, if information is now produced that requires transmission and it's not the ECU's turn to send, the transmission must wait.

To avoid this wait, CAN was developed as an event-driven bus system. Every CAN node is authorized to access the CAN bus immediately after occurrence of an event and to send data that has been created. The only exception is if another ECU is already transmitting data. Courtesy dictates that another transmission is not to be interrupted.

The important thing here is that the other transmission does not last too long. CAN limits the message length to a maximum of 130 bits (for 11 bit identifiers). With the usual data transmission rate of 500 kbit/s in passenger cars, this leads to a transmission duration of approximately 0.25 milliseconds. The bus is available again after that. This is an important precondition for data transmission that must be sufficiently fast for applications like drive and chassis.

However, there is still a risk of collisions, namely when multiple ECUs want to start transmitting messages simultaneously after the bus becomes available again after a transmission. This danger rises with increasing bus load. If messages were to be destroyed as a result of a collision, this would cause the bus load to increase and initiate a vicious cycle that would place the availability of sufficient data transmission speed in doubt. To avoid this, the CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) bus access method is used in the CAN network.

Priorities Instead of Collisions

When an ECU wants to send, it must check whether the bus is free (Carrier Sense – CS). If the bus is busy, the ECU must wait. When the bus becomes available again, there is a possibility that other ECUs have been waiting for it too. In this case, all ECUs start sending messages (Multiple Access – MA). To avoid the impending damage from this collision (Collision Avoidance – CA or Collision Resolution – CR), a process now occurs that is referred to as bitwise arbitration.

All ECUs with a transmission request simultaneously send the identifier of their respective CAN message to be transmitted, bitwise from the most significant to least significant bit.

A bit with significance 0 is dominant on the CAN bus. This means if two ECUs simultaneously transmit different bit

values, the 0 value prevails over the 1 value on the bus. This is called "Wired AND bus logic" (0 = dominant, 1 = recessive). For each ID bit that is sent, each ECU compares the value on the bus with the value it sent. This is called bit monitoring. The rules of the arbitration logic determine whether an ECU may continue sending or must stop **(Figure 3)**.

One bit after the transmission of the identifier the arbitration ends. All ECUs that transferred a logical 1 when sending their identifier but found a 0 on the bus, had to stop sending. Only one ECU did not have to stop and can now transmit its entire message uncontested.

Every loser of an arbitration changes to receive mode and waits until the winner has finished transmitting its message. As soon as the bus is free again, the ECUs access it for another send attempt. In this way, the bus logic and arbitration logic not only prevent collisions (Collision Avoidance) but also ensure a priority-controlled bus access, since the smaller the numerical value of an identifier, the higher the priority of the related CAN message. Assuming the bus is free, the message with the smallest identifier (ID = 0) is therefore transmitted without delay. In the case of collisions, messages with numerically higher ID values have a risk of delay as a result of losing the arbitration.

With a lower bus load, the probability of collisions is low. This type of random, nondestructive, and priority-controlled bus access ensures fair and very fast bus access. When the bus load increases, collisions and delays of, in particular, low priority messages accumulate. In the worst case, information arrives too late at their receivers or not at all. For this reason, it is necessary to ensure when planning and developing a CAN bus that the bus load will not be too high.

The bitwise arbitration has yet another small obvious disadvantage: A bit must have a long enough duration so that delays in sending do not pose problems. An electrical signal has a propagation speed of approximately 200 000 km/s or 0.2 m/ns in copper. Additionally, there are delays through the CAN controller and transceiver. A bit level must last long enough on the bus that the ECUs most distant from the sender can still detect and evaluate it. As a result, there is a reciprocal relationship between the bus length and transmission rate.

Figure 3: Dominance of Zero Bit and Arbitration

Ordering and Delivering Data

Road traffic includes public transit buses that run according to schedule and charter buses that run only when ordered. We find both models in CAN, too. Messages can be sent cyclically at regular intervals or only on request.

CAN buses in vehicles mainly use cyclic transmissions. CAN buses are shorter than 40 m in passenger cars and shorter than 200 m in commercial vehicles. As a result, high transmission rates are possible that allow cyclic sending without the bus load becoming too high.

On long CAN buses (over a kilometer), found in building complexes or industrial plants, only low transmission rates are possible. Cyclic sending of messages makes sense only if the cycles are correspondingly long. However, data can be sent on request.

There is the Data Frame for transmitting data **(Figure 4)** and the Remote Frame for requesting data. Because an identifier indicates the content of the data, it can be used by either, the Remote Frame and the Data Frame for transmitting the requested data. The Remote Transmission Request (RTR) bit, included at the end of the identifier, is also used for arbitration. This prevents problems if a remote frame and the associated data frame collide during arbitration. If RTR = 1, a Remote Frame is present. Otherwise, a Data Frame is present.

A Bit for the Clock Comparison

Synchronization between sender and receivers is a basic precondition for the transmission of Data Frames and Remote Frames. For reasons of cost and effort, CAN has no clock line. Synchronization is realized with signal edges and a defined mechanism. If nothing is being sent on CAN (bus idle), the bus level is recessive, as would be the case if only bits with value 1 were being sent. For this reason, a message begins with the transmission of a dominant synchronization bit (Start of Frame – SOF). All receivers adjust their clocks at the falling signal edge of this bit.

Figure 4: CAN Data Frame

Each receiver ensures synchronization throughout the transmission by evaluating each falling signal edge and adjusting its own time pattern (Bit Timing), if necessary. Long series of homogeneous bits do not exhibit any signal edges. To break these series, CAN generates additional signal edges by inserting a complimentary bit after five of equal value. This is called bit stuffing. In this way, a falling edge is forced after 10 bits at the latest. After synchronization, the receivers discard these stuff bits.

The SOF (Start of Frame) is followed by either the 11 bit ID or 29 bit ID. Its length is determined in the IDE bit (Identifier Extension). IDE = 0 means short IDs. IDE = 1 means long IDs. CAN allows a maximum of eight byte payload in the data field. The exact number of payload bytes is indicated using a DLC (Data Length Code). The data field is followed by the 15 bit check field, also called the checksum or cyclic redundancy check – CRC.

The sender calculates this checksum from all bits to be transmitted. In mathematical terms, it divides the bit string by the 15th degree polynomial 0xC599 or $1x^{15} + 1x^{14} + 1x^{10} + 1x^{8} + 1x^{7} + 1x^{4} + 1x^{3} + 1x^{0}$ on the binary field, thus $x \in \{0,1\}$.

Each receiver does the same thing with the arriving bits. It then compares both sequences and rates the received message in the acknowledge slot (ACK slot) following the recessive CRC delimiter. 0 stands for good. 1 stands for error. A CAN message is concluded after the recessive ACK delimiter with the 7 bit recessive EOF (End of Frame). Because no more stuff bits are inserted after the CRC, the EOF is a one-to-one identifier for the end of the message. The EOF is followed by three recessive bits, but these no longer belong to the message. They are called an Intermission (ITM) or Interframe Space (IFS). Only after this is another message permitted to be sent.

Error Detection and the Bull in the China Shop

The probability that errors in CAN messages remain undetected is very low at $4.7 * 10^{-11}$ or less [2]. The CAN protocol has defined error detection mechanisms. On the receiver side, these include not only the CRC but also the check of the format (Form Check) and the check according to the bit stuffing rule (Stuff Check). The sender uses bit monitoring to check whether a bit level on the bus corresponds to the send request, and it evaluates the ACK slot.

Figure 5: CAN Error Frame

For example, if statistics indicate that every thousandth message on a CAN bus exhibits an error and the CAN bus operates 1000 hours per year at a data rate of 500 kbit/s, an average bus load of 25 % and an average message length of 80 bits, statistically a CAN message with an undetected error would only occur every 4000 years.

As soon as an ECU notices an error, it aborts the message transmission by sending six dominant bits. This so-called Error Flag is the proverbial bull in the china shop of the CAN protocol. Wherever it appears, it deliberately violates applicable rules, i.e. in most cases the bit stuffing rule. This impairs the affected CAN transmission to such an extent that all ECUs on the CAN bus can also detect only locally detectable errors. A second Error Flag follows because all other ECUs now see the error generated by the first Error Flag. Then eight recessive bits are awaited. This is called the Error Delimiter. The two Error Flags and the Error Delimiter together are called the Error Frame **(Figure 5)**. It is essential for network-wide data consistency.

Errors must be corrected! The sender of the corrupted message does this by retrying as soon as the CAN bus is free again, thus, after the Error Frame and ITM. However, there is no guarantee that the retry takes place immediately because every ECU is allowed to start sending after each ITM and which message is passed through is determined by the arbitration process. The error recovery time thus depends on the message priority and the bus load.

What would happen if an ECU signals an error where none existed? Active message transmissions would be aborted needlessly and the communication would be disrupted. This is prevented through self-monitoring of the CAN controller using an error counter. When an ECU detects an error first, i.e. sends the first of the two Error Flags, it must increase its counter by 8 points. As the sender of the second Error Flag, it only has to increment its counter by 1. If a message is received without error, the counter is reduced by 1. If the ECU detects the error when sending, it increases the Transmission Error Counter (TEC). Otherwise, it increases the Receive Error Counter (REC). If one of these counters exceeds the value 127, the CAN controller changes over to Error Passive mode. Consequently it can no longer send Error Flags consisting of dominant bits. Generating Error Flags with recessive level, it no longer interrupts or disturbs other ECUs, disturbing only itself while sending. The TEC continues to be incremented when errors occur until it forces the shutdown of the ECU when value 255 is reached **(Figure 6)**.

CAN Messages Are Graded

Every message transmission is graded by all receivers simultaneously. They all return an evaluation exactly within the ACK bit during the transmission of the sender. This is what is called an "In Transmission Response" or an "In

Frame Acknowledgment". A dominant level corresponds to a Good grade and a recessive level to a Poor grade. Because the sender sends the ACK slot recessively, already one Good grade is enough to confirm the correctness of the message transmission. Poor grades of other ECUs would thus be overwritten and remain unheard initially. However, these ECUs send an Error Flag after the ACK delimiter.
If there is no Good evaluation at all and the ACK slot remains recessive, the sender specifies an ACK error and cancels its message transmission with an Error Flag.

The Limits of CAN

CAN is still the most commonly used bus technology in automobiles. However, CAN has systemic limits. Due to the principle of arbitration, CAN itself is not deterministic even during cyclic sending of messages. Using CAN, extremely time-critical applications do not function with sufficient reliability. In addition, only a maximum of one million bits per second can be transmitted with CAN. Another is the high degree of reliability of CAN that makes it too expensive for simple applications that can manage without it.
Driver assistance or autonomous driving as well as comfort applications in the audio area and especially the video area require significantly higher data transmission rates and place higher demands on timely availability of data. Bus systems such as FlexRay and networks such as MOST and Ethernet have been established in these areas. But CAN also has been further developed, and an expansion of CAN, called CAN FD, is now available that enables significantly higher transmission rates. The revised CAN standard ISO 118981:2015 containing CAN FD has been in place since December 2015.
On the other end, LIN (Local Interconnect Network) fills the gap when it comes to cost-effective networking of sensors and actuators in the comfort area, such as for control of power windows, seats, external mirrors, sunroofs, and the air conditioning.

Reliable ECU Networking

Our specialists at Vector [3] support manufacturers and suppliers in the automotive industry and other industries not only in CAN networking, but also in communication systems such as LIN, FlexRay, CAN FD, and Ethernet.
We offer consistent tool chains for planning, design, development, and maintenance as well as software components and basic software for AUTOSAR ECUs. For getting started with ECU networking, we offer seminars on the basics of CAN, LIN, FlexRay, and Ethernet. To gain the necessary skills for handling the above named tools, engineers and technicians attend workshops where they learn all about the multifaceted development tasks concerning electronics in automobiles [4].
In the first part of this series [5], we addressed serial bus systems in general. The serial bus systems LIN, FlexRay, and MOST will be addressed in the remaining articles in the series. Interested readers will find additional in-depth information on the previously published topics on the Vector E-Learning platform [6].

Literature References:

[1] de.wikipedia.org/wiki/Controller_Area_Network
[2] Unruh, J.; Mathony, H. J.; Kaiser, K.H: Error Detection Analysis of Automotive Communication Protocols. SAE International Congress 1990.
[3] www.vector.com
[4] www.vector-academy.com
[5] Christmann, E.: Data Communication in the Automobile – Part 1: Architecture, Tasks, and Advantages of Serial Bus Systems
[6] elearning.vector.com

Ernst Christmann, Physicist, Mathematician
is a Senior Technical Trainer in the area of software tools for ECU testing and development and has worked for Vector Informatik GmbH in Stuttgart since 2004.

Figure 6: States of a CAN Controller

CAN Gets Even Better

Ways to Transition from Classic CAN to the Improved CAN FD

Robert Bosch introduced the new CAN protocol CAN FD (CAN with Flexible Data rate) in March 2012. Key new features of this protocol are that it extends the useful data length from eight to 64 bytes and offers significantly higher data transmission rates. Based on its performance data, CAN FD is positioned between High Speed CAN (1 Mbit/s) and FlexRay (10 Mbit/s). It is ideal for filling the performance gap between these bus systems at low cost. This article explains, from the perspective of a tool supplier, what needs to be modified and the effects of such changes on CAN FD development and simulation. These modifications are made on various levels: from the hardware level to potential data formats and various communication layers.

CAN FD gives the automotive industry a new foundation for better networking solutions in areas where bottlenecks have been occurring for years now due to growth in data traffic in vehicle electronics. This situation has resulted in compromises and makeshift approaches such as subdividing a network into multiple buses, which drives costs higher. However, switching from CAN to the higher performance FlexRay is associated with high investment costs. For the majority of developers a migration from the event-driven CAN to the time-triggered FlexRay also requires enormous changes to familiar ways of working and thinking.

Evolution Instead of Revolution

For around 20 years now, CAN has been the dominant bus system in the automotive industry, and so a broad base of developer know-how already exists in this area. This expertise can still be applied to CAN FD projects, since CAN FD preserves existing CAN concepts such as bus arbitration, message acknowledgment, event-driven control, etc. CAN FD can be used very flexibly. Applications can benefit from either the higher bit rate, longer useful data length or both. When somewhat higher transmission rates are needed due to long bus lines, e.g. in some truck applications, the useful data length of up to 64 bytes helps to achieve noticeably higher data throughput.

Although CAN FD shares many of the same types of functionality as CAN, improved protocol extensions and adaptations to hardware and software are still required. Among other things, CAN FD introduces three new bits to the control field: the EDL (Extended Data Length), BRS (Bit Rate Switch) and ESI (Error State Indicator) bits. The EDL bit distinguishes frames in CAN FD format from those in classic CAN format: the EDL bit is recessive (High level) for CAN FD frames and dominant (Low level) for CAN frames. Similarly, a recessive BRS bit switches to the higher bit rate when sending the data field. The ESI bit is used to identify an error state of a CAN FD node. In addition, another four bits form the Data Length Code (DLC) that covers the extended useful data length; its possible values are 12, 16, 20, 24, 32, 48 and 64 bytes **(Figure 1)**.

Challenge for Tool Suppliers

For a supplier of software development tools, the challenge in introducing a new system such as CAN FD is to make suitable tools available as soon as the customer group begins to develop its first CAN FD ECUs. Integrating CAN FD in an existing test and simulation environment requires making rapid modifications to the various hardware levels up to the application level. An associated database is also needed to describe the communication network. In this context, the first question that should be asked is what is the best way to model CAN FD? Is a PDU abstraction necessary, considering the useful data length of up to 64 bytes? PDUs (Protocol Data Unit) are not only commonly used with FlexRay; they are also supported by the AUTOSAR System Description. They can be used for such purposes as additional CRC checks, or they can serve as Tx triggers.

Essentially, it makes sense to preserve as many of the practice-proven aspects of the CAN world as possible in modeling CAN FD. For example, it is possible to quickly migrate existing CAN test and simulation systems if CAN FD is not treated as a new bus system, but instead as an extension of CAN. Then the necessary configuration changes to tools are relatively minor. It is even possible to continue to use test scripts and databases. Then, OEMs and suppliers would be able to rapidly implement initial projects if they can initially limit CAN FD to eight data bytes. It is still possible to use DBC databases; they are widely used in CAN applications and represent the quasi industrial standard. In addition, DBC can already handle useful data that is 64 bytes in length for protocols such as J1939.

Flexible Hardware by FPGAs and Interchangeable Transceivers

The structure of the software tools – and their primary uses of analysis, testing, logging and simulation – reveal that CAN FD adaptations are required on nearly all layers of the automotive protocol stack. On the physical level, the tools almost always utilize a network interface for bus access. Network interfaces can be implemented in various ways, based on either FPGAs or standard controller chips. The latter represent a cost-effective solution, but they are strictly limited to a specific set of functions, and they must first be made available for CAN FD. The more flexible and quickly available alternative – at least in the initial CAN FD phase – is to use interfaces with FPGAs. This technology not only fulfils the necessary real-time requirements, it also enables driver updates to provide new functions such as 64-byte support or bug fixes. The question of whether CAN FD needs new transceivers depends on the specific use case. In vehicle networks, it is expected that a transmission rate between two and four Mbit/s will be implemented. For flashing ECUs, however, the rate should be as high as possible; eight Mbit/s or more are conceivable here, depending on the CAN transceiver that is used. Configurable hardware with piggyback transceivers, i.e. small interchangeable plug-on boards **(Figure 2)**, offers a flexible approach that will support future transceivers.

On the communications level, developers often need analysis tools that depict the frames and enable access to specific data of the frame. Here, it is absolutely essential to correctly interpret the new CAN FD bits EDL, BRS and ESI; the same applies to the extended useful data length **(Figure 3)**. It is easy to make the necessary modifications by modeling frames as events sent out by the tool. That is, it is sufficient to add the the CAN FD bits to existing CAN events and

Figure 1: CAN FD frame

Figure 2: CAN FD bus interfaces with piggyback transceivers

extend the useful data length to the longer length. Only analysis windows that display the messages need to distinguish between CAN frames and CAN FD frames.

CAN FD on Higher Layers

On higher layers, the focus is on diagnostics, network management (NM), the transport protocol (TP), interaction layer (IL) and Tx models. While applications on the transport layer interpret even more frames, the application layer generally no longer works with frames, but with signals instead. Network management, the interaction layer and Tx models are all dependent on specific automotive OEM requirements. The extended useful data length of CAN FD affects both the interaction layer and the transport layer, and so it requires modifications. The advantage of modular tools with suitable interfaces is that not only are the OEM-specific features easy to implement or interchange, but the extensions required for CAN FD are easy to implement as well. On the application level, developers are in a good starting situation if they have designed their applications so that they are independent of a specific protocol and they are systematically based on signals. They can continue to use their simulation models, test scripts and graphic user interfaces nearly unmodified.

The Cards Have Been Reshuffled: CAN (FD), LIN, FlexRay

Of course, the process of introducing a new bus system has some effects on current network protocols. Hardly any changes are expected in the convenience area of LIN, since CAN FD does not offer any advantages there. It must be realized, however, that it is impossible to implement simple TP Routing (known as raw TP) with useful data greater than eight bytes by simply copying the data of a CAN frame to a LIN frame. The situation is different with FlexRay, since CAN FD is a more economical alternative that is always the preferred solution when event-driven applications are being used. The use of FlexRay, on the other hand, is more limited to real-time critical and deterministic applications. Some migrations will be made in existing CAN systems. Networks with high bus loads of about 50 percent are candidates for CAN FD. The higher bandwidth of CAN FD avoids splitting networks with high bus loads, and it is then possible to track multiple networks back to a single one. This saves on costs by not requiring as many gateways, and a beneficial side effect is that it eliminates undesirable gateway latencies.

Currently, there is no consensus on whether there will be mixed CAN/CAN FD networks; this depends primarily on decisions by the vehicle manufacturer. Certain actions would definitely need to be taken to implement mixed operation of CAN and CAN FD. For example, in FD operation CAN nodes need to be passively disconnected, because they would otherwise detect a form error due to the EDL bit and would send an Error Frame.

Figure 3: CAN FD simulation and test tool

Migration from CAN to CAN FD

An advanced analysis, test and simulation tool can offer valuable assistance in nearly all situations related to the development of CAN FD ECUs and networks. The possibilities range from analyzing an individual network node to fully simulating entire networks. Such a tool would give developers the ability to predict future bus loads and make optimal decisions based on reliable simulation data, e.g. to determine whether or not it is necessary to make a split into several networks. The tool is used to perform remaining bus simulations over the course of development, which substitute for a missing subnetwork and its components. Simulated nodes are then be replaced by real ECUs in a step by step manner. The simulation can also serve as a gateway, and it can connect to a new CAN FD network with a classic CAN branch. It is not necessary to make any modifications to existing CAN ECUs here.

Conclusion

With flexible FPGA-based network interfaces and high-performance analysis, test and simulation tools, automotive OEMs and suppliers have all of the components they need for successful CAN FD developments. The described system supports different database formats and enables a variety of abstractions on the signal and PDU level. Open interfaces make it possible to implement OEM-specific properties. Network simulations and remaining bus simulations ensure optimal test conditions in all situations that occur during the development process.

Translation of a German publication in Elektronik automotive, issue 04/2013

Image rights:
Vector Informatik GmbH

Literature:

CAN FD specification V1.0, Robert Bosch GmbH

Link:

Vector CAN Solutions: www.vector.com/can_fd

Peter Decker
has been employed at Vector Informatik GmbH in Stuttgart since 2002 and is currently Product Manager in the area of Multi-Bus Development for CANoe/CANalyzer.

CAN FD: Measuring and reprogramming

The complexity of the CAN FD technology is equivalent to the regular CAN network but it offers a significantly increased bandwidth. It is therefore an alternative to Flexray or Ethernet networks.

Author

Armin Happel

Erik Sparrer

Peter Decker

Vector Informatik GmbH
Ingersheimer Str. 24
DE-70499 Stuttgart
Tel.: +49-711-80670-0
Fax: +49-711-80670-111

Links
www.vector.com

Literature:
CAN-FD specification V1.0, Robert Bosch GmbH

CAN with a flexible data rate (CAN FD) is a technological evolution of the CAN network. It provides more bandwidth than CAN with less complexity than Flexray. The network specialists at Vector investigated two typical applications, measurement of ECU internal signals via XCP and ECU reprogramming, using the CAN FD system.

ECU measurement with XCP on CAN FD

In ECU development, the measurement and calibration of multiple signals and parameters for open and closed loop control algorithms represents an important calibration use case. ECU developers prefer to use the XCP (Universal Measurement and Calibration Protocol) measurement and calibration protocol that has been standardized by ASAM e.V. In the current protocol version 1.2, CAN FD is introduced as a new XCP transport layer. XCP enables the utilization of measurement and calibration tools such as Vector's CANape (Figure 1) to modify parameters during real-time operation and measure the altered behavior of the ECU. Considering a CAN system the bandwidth of the transmission medium may quickly become exhausted, depending on the number of signals to be monitored. XCP on CAN FD significantly extends the capabilities with up to 64 bytes of payload and data rates of at least 5 Mbit/s in the data phase.

XCP on CAN FD data throughput

To estimate the maximum available data throughput of XCP over CAN respective to CAN FD, the frame size versus the available payload within a frame has been investigated for a measurement of multiple ECU signals. The data throughput calculations are based on the assumption of 100 % bus load. The actual size of the frame fields for CAN and CAN FD are shown in Table 1 and Table 2. However, frame sizes cannot be predicted precisely for either CAN or CAN FD. To assure synchronization of bus nodes to signal edges, in dependence of its content an a-priori unknown amount of additional stuff bits is in- ▷

Figure 1: Measurement over XCP on CAN FD with CANape

serted into the frame. To model the stuff bit dependent frame size variation, a best and worst case scenario has been analyzed.

The results of data throughput calculations are graphically represented as a sector (Figure 2, Table 3), where a frame may reside in dependent of its actual contents. To verify the theoretical calculation, a realistic measurement reflecting a practical measurement use case was processed based on a simulation environment. At the laboratory setup – which consists of CANape measurement and calibration software, suitable interface hardware and a PC-based ECU emulation – the time of flight between the in- and output of the CAN/CAN FD driver was measured in both directions. The experimentally measured values greatly meet the mathematical prediction (Figure 2, Table 5) and hence verify the modeling of the available data throughput. Comparing the acquired measurement data needed for a transmission using CAN versus CAN FD, the data throughput of CAN FD has been found to be increased by factor 1.3 up to 5.4 depending on the system's configuration (Table 4).

Above its already improved bandwidth, XCP over CAN FD possesses further potential for improvement. Due to the equivalent physical communication basis of CAN and CAN FD, it is likely that the communication paths of existing ECU software will still be limited to an eight-byte data transmission after migrating to CAN FD. In this case XCP can only benefit from the higher data transmission rate but cannot utilize the full 64 bytes of payload available in CAN FD frames. To optimize the data transmission rate, the payload of small XCP packets could be concatenated as a large CAN FD frame (Figure 3). Vector is currently working on a proposal that enables packet concatenation for XCP over CAN FD in a future XCP specification.

Table 1: Structure of a CAN frame

Name	Size [Bit]
Start of Frame	1
Arbitration Field	12
Control Field	6
Data Field	≤ 64
CRC Field	15
Acknowledge Field	2
End of Frame	10

Table 2: Structure of a CAN FD frame

Name	Size [Bit]
Start of Frame	1
Arbitration Field	12
Control Filed (1st part)	4
Control Field (2nd part)	5
Data Field*	≤ 512
CRC Field*	18 / 22
Ackknowledge Field	2
End of Frame	10

Flash programming

(Re-) programming of flash memory is the second use case in which significant improvements are expected through the utilization of fast network protocols. In the three flash phases "delete", "download/program" and "verify", the download time is a key factor in conventional CAN systems, that can be accelerated by faster bus systems such as Flexray, Ethernet and CAN FD.

Regardless of the transmission protocol, it makes sense to use additional optimization strategies for downloading, such as data compression and pipelined programming. Although compression by an LZSS (Lempel-Ziv-Storer-Szymanski) algorithm reduces the volume of data to be transmitted, its efficiency is highly dependent on the data structure, and data extraction in the ECU generates additional CPU load that need to be taken into account. Pipelined programming, on the other hand, represents a type of parallelization: while a data segment is still being written in the ECU, transmission of the next segment is already started. Therefore, the potential performance gain from this method is the greatest when programming times are shorter than data transmission times.

Flexray offers a transmission rate of 10 Mbit/s, but it is not fully available for (re-) programming. In the periodic communication sequence of the time-triggered protocol, all PDUs (Protocol Data Unit) are predefined in fixed slots. If many slots are reserved for diagnostic service requests such as for download, this reduces bandwidth for the useful data. Realistic configurations provide for 4 PDUs to 8 PDUs with 42 bytes to 255 bytes each per cycle for diagnose services. Vector engineers have measured download rates of 40 to 60 kB/s when pipelined programming is used. ▷

Figure 2: Measured and calculated CAN FD data throughput in ECU measurement

Table 3: Calculated data throughputs of data measurement with XCP on CAN FD (fA=500 kbit/s)

f_D [kBit/s]	$\overline{f_T}$ [kBit/s] Best Case	$\overline{f_T}$ [kBit/s] Worst Case
500	407	341
1000	753	635
4000	2130	1825

Table 4: Comparison of measured data throughputs of data measurement with XCP on CAN and CAN FD

	Measured CAN	Measured CAN FD	Factor
Min.	294 kBit/s	380 kBit/s	1.3
Max.	318 kBit/s	1712 kBit/s	5.4

Table 5: Measured data throughputs of a data measurement with XCP on CAN FD (fA=500 kbit/s).

f_D [kBit/s]	f_T [kBit/s]
500	401 ± 21
1000	724 ± 46
4000	1884 ± 172

Ethernet with Diagnostics over IP (DoIP) per ISO 13400-2 is also well-suited for fast reprogramming of ECUs. In testing 100 Mbit Ethernet and a typical microcontroller with a pure flash write rate of 180 kB/s, results were largely a function of the buffer size of the Transfer-Data service. A 16 KiB buffer enables throughput of around 150 kB/s, which is already near the limit of the flash memory used in the test.

Reprogramming via CAN FD

Since semiconductor manufacturers do not offer any microcontrollers that provide CAN FD support yet, network specialists at Vector used a microcontroller in which the CAN FD controller was implemented in an FPGA for their CAN FD measurements. The software stack on the board consists of a standard Vector UDS bootloader. The ISO 15765-2 transport layer and CAN driver were extended for support of CAN FD. To permit a quick test setup process for download testing, the CANoe simulation and testing tool was used, because the tool already offers CAN FD support. This software uses an external DLL which provides the flash programming procedure and transport layer functions. In the future, the Vector vFlash flash tool will become available for CAN FD.

Figure 4: Comparison of download and programming times with CAN and CAN FD

With the transport layer that is used, the theoretically attainable transmission rate in flashing over CAN FD is 270 kB/s to 370 kB/s at 4 Mbit/s in the CAN FD data phase. However, real measured values lie well below this (Figure 4). Surprisingly, the compression and pipelining optimization strategies were counterproductive for CAN FD in the test environment that was used. The reason is that, in the laboratory setup used, the programming time for the internal flash memory became the limiting factor in the flashing process. So this made optimizations to the download phase ineffective. However, further tests with more powerful CPUs are needed to arrive at more general conclusions about data throughput and the effectiveness of optimizations. A key finding of the measurements is that CAN FD delivers a significantly higher data throughput than CAN (Figure 4), and the effort required for migration is negligible.

Figure 3: Faster data transmission by multiple XCP packets combined in one CAN FD frame

Summary and outlook

Overall, it is still difficult to arrive at an objective comparison of the serial bus systems CAN FD, Flexray and Ethernet due to their different microcontrollers and constraints, but certain tendencies can be clearly discerned. In the case of Flexray, high download speeds and high performance for the real time data payload are not both achievable at the same time. 100 Mbit Ethernet delivers the fastest transmission rates, but it requires complex software configuration, and its hardware costs are higher than for CAN FD. CAN FD appears to be the most balanced solution, it offers high data rates and the potential for further improvement at moderate costs. In addition, it is relatively easy to migrate to the improved CAN, because of the close similarities between CAN and CAN FD. Both protocols are based on the same physical layer, and this enables reuse of transceivers, wiring and bus topologies. Since the communication principle has not changed either, existing know-how can still be applied. The modifications to affected software layers in calibration and reprogramming that need to be made are relatively minor.

CAN FD enables significant throughput gains in both measurement and reprogramming of ECUs. In (re-) programming, this shifts the bottleneck more to the flash memory. Further development to shorten the memory access times of the MCUs that are used promise additional performance gains. Efforts by Vector to extend the XCP specifications to include packet concatenation with CAN FD also offer the potential for increasing performance of the new protocol that is still untapped. ◀

Eye diagram analysis for CAN FD

(Photo: Vector Informatik)

The eye diagram is an analysis method for evaluating the signal quality of transmission networks. Eye diagram analysis can help finding corrupting influences on CAN networks.

CAN networks can transmit information bit by bit from sender to partly distant receivers. The information transfer can, however, be easily corrupted by the network topology, cable length between the participants, line and terminating resistors as well as external electric influences. Both eye diagram analysis and serial bit mask analysis allow these influences to be identified and corrected early during the configuration phase of the CAN network.

Due to the higher and flexible bit-rate of the data phase, CAN FD is considerably more noise-sensitive than Classical CAN, which has a fixed bit-rate only for the entire frame. The transmission of CAN FD frames always begins with the lower bit-rate of the arbitration phase. A switch to the higher bit-rate of the data phase is made at the sampling point of the bit-rate switch (BRS) bit. At the sampling point of the cyclic redundancy check (CRC) delimiter bit, the bit-rate is switched back to the lower bit-rate. The shorter duration of the bits during the faster data phase has a negative effect on the signal quality, which can best be analyzed with an eye diagram. The signal quality is highly dependent on:

- Design and complexity of the bus topology e.g. line or star topology,
- Adversely selected cable routing and/or bus termination,
- Susceptibility of the system to higher transmission rates typical in vehicle networks,
- Cable properties, such as their impedances or shielding.

Creation of a classical eye diagram

During the network design, a configuration for the phase segments of the bit as well as their sampling points is created separately for the arbitration phase and the data phase. According to the CAN protocol, one bit is time-divided into four segments: a synchronization segment, a segment for compensating time delays, and two phase segments, which compensate the phase errors of the bit edges. The phase segments are often designated TSEG1 and TSEG2. The TSEG 1 summarizes the first phase segment and the compensation segment. Figure 1 shows the time division of a bit into the mentioned segments. The sampling point lies between the phase segments. TSEG2, on the other hand, corresponds to the phase segment which begins after the sampling point. Since the grid of the superimposed bits is displayed directly in the eye diagram, it is immediately apparent how well the superimposed bits fit into this grid.

The bit edges should ideally lie within the synchronization segment. The algorithm used here works like a real CAN controller. Due to the time delay of the bit edges, caused by the CAN transceiver delay and the jitter of the control units, CAN controllers must resynchronize at the transition from the recessive to the dominant bus level on the receiver side. This is a prerequisite for detecting the logic level of a bit at the set sampling point. With the adjustment of the phase segments, the robustness of the synchronization mechanism can be influenced. These settings are usually checked using an eye diagram, which allows the user to visualize whether his controller settings are practical and meaningful. Due to the fact that CAN FD has two different bit-rates, it is advisable to create separate eye diagrams for the arbitration phase and the data phase.

With the software tools CANoe and CANalyzer from Vector, the user is able to configure a CAN network and record voltage signals using the Option Scope. After measuring, the user can perform an eye diagram analysis, which analyzes all received frames bit by bit and superimposes them graphically in a fixed time window. In this example, the time window is a percentage of the bit duration (Figure 1). The displayed bit segments show the configured controller settings. Possible deviations in the individual bits can be quickly identified in this view. With a good controller setting, the rising edges of all bits lie in the synchronization segment. If the bit signals also reach their dominant and recessive voltage levels uniformly, i.e. without overshoot, a robust bus topology and a correctly selected bus termination can be assumed. The diagram has an “eye” due to the fact that all the bits on the x-axis are normalized to the theoretical bit-width (reciprocal of the bit-rate), while the voltage values of the bits are plotted on the y-axis. For the described case, the eye would be wide open (Figure 2). In the opposite case, the eye would be closed, which is an indication of errors in the network structure (Figure 3). The data phase is analyzed in Figures 2 and 3 with a sampling point set to 70 % and a data-rate of 2000 kbit/s.

Further refinement of analysis criteria

In order to narrow down possible sources of error, it is recommended to create an eye diagram from different aspects. For this purpose, various filter options are provided:

- Frame type, for example CAN or CAN FD,
- CAN channel number,
- Control unit name,
- Defined bit sequences.

An alternative approach for detecting protocol errors of a control unit is the analysis of different bit sequences of a frame. With this method, a distinction can for example be made between the bits which are transmitted during the acknowledge phase by all control units and those of a particular control unit.

Figure 1: Classical eye diagram with graphical representation of the phase segments of a CAN controller (Photo: Vector Informatik)

To further refine the criteria for evaluating the eye diagram, it is helpful to create a predefined theoretical eye in the form of a bit mask. For this purpose, the user generates a bit mask as a freely definable polygon, thus defining the "good" area, which may not be crossed by a voltage signal of any bit. It is useful to define a separate bit mask for the arbitration phase and for the data phase. Figure 4 shows an eye diagram with a bit mask, in which some bits violate the defined mask. In the background, the configured segments of the CAN controller are shown.

Serial bitmask analysis

So far, the classical eye diagram, which represents the overlapping of individual bits, has been considered. An alternative visualization is the serial bit analysis. In principle, both analysis procedures are performed identically. Only in the case of the serial bitmask analysis, the bits are displayed in the order sampled by an oscilloscope. Again, it is possible to define bit masks, which are displayed for every dominant and recessive bit. The advantage over the classic eye diagram is that bit errors are assigned directly bit by bit. It is also possible to analyze only part of the bit stream. The configuration possibilities already discussed for the eye diagram can also be applied to the serial bitmask analysis. Figure 5 shows all the bits of a defined analysis area. Each bit has a bit mask with the red mask indicating a violation in the first bit.

Automating the analysis process

Both analysis methods can be automated with the Vector product CANoe. To do this, the user must first define his test cases. For example, one test case is defined to analyze the data phase of all CAN FD frames and another test case for only the arbitration phase. For each test case, a specific bit mask can be used as a test criterion. If the bit mask is violated by the bit signal, the test case result will be negative. Each test case is automatically recorded, evaluated, and stored in a test report, so that the user can understand why a test case failed.

Figure 2: Open eye with bit mask – an indication of a good network design (Photo: Vector Informatik)

Figure 3: Closed eye with bit mask violation – an indication for problems in the network design (Photo: Vector Informatik)

Figure 4: Eye diagram with bit mask violation (Photo: Vector Informatik)

Figure 5: Serial bit mask analysis in CANoe/CANalyzer with a bit mask violation (Photo: Vector Informatik)

Conclusion

The methods described here for the analysis of a CAN/CAN FD network help to quickly identify and fix design faults or negative external influences. Eye diagram analysis with CANalyzer and CANoe is also available for other bus systems such as Flexray. With CANoe, the user also has the option to perform tests in a reproducible and automated manner via CAPL test sequences. Due to the high degree of automation and with suitable tools, such tests can be repeated with minimal effort. ◀

Author

Mirko Donatzer
Vector Informatik
mirko.donatzer@vector.com
www.vector.com

CAN tools – then and now

This article leads us from the modest beginnings of the CAN tools and the past challenges to the currently achieved state and the emerging trends for the future.

Figure 1: From the early days of CAN tools (Photo: Vector)

The possibilities of a bus system can only be fully utilized in combination with powerful software tools for development, diagnostics, and support. In parallel with the rise and advancement of the CAN network, therefore a significant infrastructure comprising tools for bus analysis, management of communication data, development, and simulation of bus nodes and much more has become established. This article leads us from the modest beginnings of the CAN tools and the past challenges to the currently achieved state and the emerging trends for the future. When the first vehicles with CAN in the drive train went into series production in 1990, even specialists were surprised by the robust and trouble-free operation demonstrated by the new bus system in actual use. Troubleshooting in the modern vehicles, however, proved to cause considerable problems for developers and repair shops. This was due, on the one hand, to the lack of know-how in the area of bus technology and networked electronics and, on the other hand, to the lack of suitable tools for bus diagnostics. In some cases, this helplessness resulted in vehicles needing to be sent to the factory for error analysis. There, the development engineers used provisional tools to try and localize errors and document the procedures. Work on bit-level and with hex representations was, at this point in time, both inefficient and without alternative.

First technically adequate CAN tool

The engineers at Vector soon recognized this problem and began developing a user-friendly CAN tool for the motor vehicle sector in 1991. The application area of this software with the catchy name “CANalyzer” covered both development as well as troubleshooting. With completion of the first prototype in the spring of 1992, Vector learned from the press of the founding of CAN in Automation (CiA), whose original roots lie in general automation. After immediately contacting Holger Zeltwanger, CiA co-founder, Vector became a very early member of the CAN user organization in April 1992. On the periphery of iNet in Sindelfingen, Germany, in June of 1992, the first CiA meeting with Vector in attendance was held. At the same time, the company presented CANalyzer to the public for the first time at this trade fair, a tool that finally allowed developers to work in a comfortable, technically adequate manner. The author of this article regularly gave lectures at CiA events and held training courses on the topic of CAN in the following years. ▷

Figure 2: CAN analysis 1993: Compaq Portable with ISA-CAN interface card /Photo: Vector)

The computer and software world of that time was significantly different from that of today. Typical PCs were IBM PC/AT compatible computers equipped with 80386-CPU, clock speeds of 25 MHz and 1 MiB of RAM, while the operating system of choice was called MS-DOS. Portable computers also existed at that time, but they were truly heavy “boxes” which looked more like a sewing machine than a computer system (Figure 2). At nearly the same time as the CANalyzer presentation, Microsoft Windows 3.1 was launched on the market in early 1992. The predecessor versions of this operating system were found to be much too slow and unreliable for technical software with real-time requirements.

Vector’s CAN tool was still based on MS-DOS, but had a self-developed and fast graphical operating system with window management that had already proven itself in calibration tools developed for Bosch. This was a decisive advantage over the text-based MS-DOS tools from other producers, as it offered much better user guidance. Installation was very simple: The user simply copied the approximately 1 MiB of content located on a diskette to a PC directory, selected the correct graphics driver (Hercules, EGA, or VGA) and started an EXE file. Today, a single digital photo is about four times larger than the content of the installation diskette of the first CANalyzer version. For bus coupling, only ISA cards from Bosch and Daimler-Benz were initially available. For this purpose, the specialist secured the sales rights, in order to supply other users of CAN tools with the right hardware as well.

Depending on one's perspective, the first version of the tool can be characterized, on the one hand, as spartan and, on the other hand, as rather progressive. For example, the functionality of the first interface boards with the so-called FullCAN controllers (Intel AN82526) was still noticeably restricted. It only allowed the reception of messages whose identifiers the developer had previously entered in the object list of the controller. For the automotive ECUs of the time, the automatic reception of CAN messages on this basis was a great advantage, as the low-performance micro-controllers were not unnecessarily burdened. Universally usable CAN tools were, however, severely impaired as a result. Progressive, on the other hand, was the ability to freely program the tool. The self-developed programming language, CAPL, enabled developers and testers to both generate as well as manipulate CAN traffic. With the help of simple scripts, it was – even at that time – possible to generate correct as well as faulty data traffic for testing and stimulating test samples.

A great improvement in convenience came just a year later in 1993 with interface cards that were specially developed for CAN tools. They worked with simpler CAN controllers and could receive all messages. This, in turn, then required “high-performance” micro-controllers on the cards. Even if it might be a matter of course today, at this time it was a massive challenge to transfer the data of two high-speed CAN networks to the PC with 1 Mbit/s and 100 % bus loading without loss of messages.

The Vector tools brought a significant improvement in operation and convenience with the implementation of the CANdb data format. It is used to store and manage all parameters, messages, and signals of pre-configured CAN networks. In particular, the system permits the use of symbolic designations for messages and signals and it supports the conversion of raw bus data into physical quantities. Temperatures, for example, can be presented directly in degrees Celsius instead of in cryptic binary values, the enumeration values of codes can be displayed and much more. CANdb now gave users the ability to work with expressive and familiar designators, making coding details and ambiguous data fields in hexadecimal numbers a thing of the past. Other tool producers quickly realized this as well. As a result, CANdb soon became the de facto industry standard for pre-configured CAN networks.

The simulation of not yet implemented ECUs’ bus traffic and even complete network topologies with CANoe (CAN open environment) in 1995 marked another milestone in the history of CAN tools. For ECU developments, these remaining bus simulations are the prerequisite for realistic tests which provide information on the behavior and maturity level of the test sample, even if the final environments do not yet exist. In the meantime, Windows had established itself as the dominant PC operating system and the significantly faster Pentium-based computers finally allowed it to be used with tools.

From specialized CAN profile to multibus tool

Through the rapid proliferation of the CAN bus in both the automotive sector and in areas such as factory automation, medical technology, railway technology, and commercial vehicle technology or aviation, numerous CAN-based higher-level protocols and profiles became established. ▷

Figure 3: Network analysis today: tablets offer mobility and performance (Photo: Vector)

About the author

With respect to the CAN network, Martin Litschel can certainly be considered a pioneer. In the 1980s, he played a key role at Robert Bosch in defining the CAN protocol and was also involved in the first semiconductor implementation; he then led the development group beginning in 1987. Just one year later, he left Bosch and, together with two friends, founded Vector Informatik. Until 2014 he was responsible there for hardware and network tool development as executive director. Today he is involved with Vector as authorized officer and is active on the Vector foundation as foundation trustee.

Vector responded to this development with a series of protocol-specific variants of its CAN tools. Among the most important were the options SDS (Honeywell), Devicenet (Allen Bradley), and CANopen (CiA) for automation technology, J1939 for the commercial vehicle sector as well as CANaero for the aviation industry.

In parallel, the automotive industry developed the specialized network systems LIN, Most, and Flexray, which were no longer based on CAN. It was now a matter of expanding the existing CAN tools accordingly and giving the users the ability to continue to perform their tests and measurements in the multibus environments of the modern vehicles using the tools with which they were familiar. The highly accurate capturing, synchronization and processing of the numerous messages and signals from the various parallel bus systems continues today to be one of the greatest challenges for a tool producer.

With CAN FD (CAN Flexible Data Rate), Bosch presented an interesting successor to the Classical CAN in 2012. The main new features are a user data length that was expanded from 8 bytes to 64 bytes as well as the ability to switch to significantly higher data transmission rates. While, in the opinion of specialists, CAN will continue to be a factor for at least ten more years, the series introduction of CAN FD is just around the corner and the next generations of vehicles from various manufacturers will be equipped with it. One big advantage of CAN FD is that the existing CAN concepts such as bus arbitration, message detection, event control, etc., are retained and the developers do not need to deal with a completely different behavior. The same holds for the integration of CAN FD in the tools.

A switch to automotive Ethernet, on the other hand, involves significant changes in the familiar ways of working and thinking. The high bandwidth of Ethernet is essential for intelligent ADAS applications (Advanced Driver Assistance Systems) as well as future autonomous driving. The systems use so-called environmental models, which continuously are to be fed with large quantities of data by onboard cameras, radar sensors, etc. Instead of using a physically connected common bus system, with Ethernet one uses a distributed logical network made up of numerous switches and electrical point-to-point connections. This necessitates entirely new approaches both by network and ECU developers as well as by vehicle manufacturers. On top of this comes the handling of the significantly greater volume of data. While the previous automotive networks are defined largely statically, in future applications – especially in combination with Ethernet – service-oriented communication will play an important role. This results in greater flexibility and reusability of function units; new platforms can thereby be quickly assembled using the modular concept. On startup, the ECUs then communicate which network nodes are interested in which information and who can supply the information.

What history teaches us

In spite of modern computers with multi-core CPUs and computing power and memory sizes that are about 1000 times greater, today's challenges differ very little from those in the early stages of CAN. The fight for performance remains and the available resources are always pushed to the limits. Today's tools (Figure 3) must be able to handle extreme real-time requirements with reaction times in the sub-microsecond range. The required data transmission rates are about 100 Mbyte/s, i.e., approx. 1000 Mbit/s. Here as well, we again encounter the factor of 1000, if one thinks back to the 1 Mbit/s of the first CAN tools. Compared to the single diskette that held the first CANalyzer, a current full-installation of CANoe/CANalyzer with all options has a data volume in excess of 3 GiB, which would be equivalent to approximately 2000 of the above mentioned installation diskettes.

Vector congratulates CiA on its 25th anniversary and is thankful for the excellent collaboration over the many years. We can only speculate on what communication systems the future will bring in the next 25 years and what the data networks will look like. CAN and, above all, CAN FD have the chance to continue to play an important role in the coming decades. ◀

Author

Martin Litschel
Vector Informatik
martin.litschel@vector.com
www.vector.com

VECTOR

First Class Solutions
for Your CAN and CAN FD Projects

Your Universal Tool Chain

Increase efficiency of your projects with the universal tool chain from Vector:

> High-professionial tools for testing, flashing and calibrating ECUs
> Flexible network interfaces
> New all-in-one network disturbance interface
> High performance oscilloscope
> Proven design tools for network architectures
> Easy to configure AUTOSAR basic software
> Worldwide engineering services and trainings

More information: **www.can-solutions.com**

More CAN power by Vector: benefit from 30 years of networking experience.

Vector Informatik GmbH | Germany · USA · Brazil · France · UK · Italy · Austria · Sweden · Japan · Korea · India · China | www.vector.com

Data Communication in the Automobile – Part 3:

Cost-Effective Data Exchange with LIN

For a long time it was customary to connect ever increasing numbers of sensors, actuators, and motors directly to a central electronic control unit (ECU). However, this trend quickly reached its limits. The cabling effort increased, accompanied by larger space requirements, increasing weight, and greater noise sensitivity. More and more different cable harness and connector variants became necessary, which in turn made production, installation, and maintenance difficult. It was soon apparent that the use of a bus system to link components would be ideal in this area as well. However, CAN provides more protection than necessary for applications such as seat, window, mirror, and air conditioning control. The costs for using CAN are too high for these types of components. Manufacturers therefore looked for cheaper solutions. Simple buses with low transmission rates were developed. However, every manufacturer developed its own solution until price pressure forced more unity. LIN (Local Interconnect Network) came into being at the end of 1999. The following article explains the triumphal progression of LIN in automobiles and explains its mode of functioning.

This technology is now used so broadly that LIN can be found in almost every vehicle.

Consortium Facilitates the Breakthrough of LIN

An essential reason for the fast establishment of LIN was the founding of the LIN Consortium [1] within which well-known automobile manufacturers and suppliers as well as semiconductor and tool manufacturers joined forces to develop a global communication standard for the sensor/actuator area. With the definition of a simple, cost-effective physical layer oriented to the K-Line (ISO 9141) as well as a simple and lean communication protocol, the LIN Consortium laid the foundation for implementing simple and cost-effective ECUs. The LIN Consortium not only focused on the actual LIN communication but also on a uniform development method, the so-called LIN Workflow. However, sometimes even the best intentions and plans do not translate into reality. The LIN Workflow is an example of this.

A uniform syntax, the LIN Configuration Language, and the LIN Description File (LDF) written in this syntax are used to describe an entire LIN cluster. The LDF describes all properties of a LIN cluster, especially the communication of ECUs. With the help of the LDF, generation tools create the software components for the LIN communication. The LDF is also the basis for analysis, measurement, and test tools and remaining bus emulators **(Figure 1)**.

Two features distinguish LIN from all other bus systems. First, there are two types of ECUs, namely a LIN Master for each LIN bus and additional LIN Slaves. Second, there is a possibility to produce slave ECUs in very high quantities and to sell them to vehicle manufacturers as ready-made mass-produced goods. These high quantities reduce the price further and the market supplies manufacturers with ECUs on which they do not have to expend any in-house development effort. For example, every vehicle has many small electric motors in seats, mirrors, windows, and sunroofs. And why should any manufacturer develop their own ECUs to control them?

Large suppliers offer such slave ECUs whose capabilities they describe in the standardized Node Capability File (NCF) using a uniform syntax, the so-called Node Capability Language. The NCF describes performance features of a LIN Slave such as the frame and signal definitions, bit rates, and diagnostic parameters and is the basis for automated creation of an LDF within the framework of the system design.

The detailed and clearly arranged specification for LIN also contributed to its success. The LIN Specification 2.2A [2], available since early 2011, defines the physical layer, the communication protocol, the LIN Workflow, the LIN API, and the diagnostics and configuration of LIN nodes. The ISO standard for LIN (ISO 17987) has been in place since January 2016. Parts 1, 2, 3, 4, and 6 will be published in the next two months. Part 7 will follow shortly thereafter.

Dispense with Everything That is Dispensable

The striving for cost savings is evident, especially in the design of the physical layer. An ordinary single wire must suffice. Voltage dividers are dispensed with, and as the voltage level for LIN bits, the supply voltage is used for a bit value of 1 and the vehicle ground for the bit value of 0. A communication controller is also not required, and there is not always a need for an expensive calibrated quartz as clock. Rather, an inexpensive on-chip resonator with a frequency tolerance of up to ±14 % will suffice. In practice, there is an increasing move away from the cheap RC clocks, and there are now also LIN controllers available.

A LIN transceiver ensures the physical bus connection. A level of less than 40 % of the supply voltage is interpreted by the receiver as logical zero. A level above 60 % of the supply voltage is interpreted as logical one. This large difference between zero and one makes the bits very immune to noise, but the edges are very steep, which always causes electromagnetic interference. To keep this low, the maximum data rate is limited to 20 kbit/s and the maximum total capacitance is limited to 2 µF. Due to the wire capacitances and the capacitances of the transceivers, a 40 m long LIN bus can connect a maximum of 16 ECUs.

Figure 1: LIN Cluster

Like in School

A cluster consists of a LIN Master and at least one LIN Slave. The communication is controlled by the master. Software in the microcontrollers performs the task of the missing communication controller. Each LIN ECU including the master has a slave task. The LIN Master additionally has a master task that it uses to control the communication **(Figure 2)**.

Like in school, there is no speaking on the LIN bus without being called on. Like a teacher, the master task poses questions that are to be answered by the respective responsible slave task. Sometimes, teachers ask rhetorical questions that they answer themselves. Likewise, the master task can pose questions that are answered by the slave task of the master.

The questions are chronologically sequenced using one or more LIN schedules that are organized into Frame Slots. At the start of each Frame Slot, the master task transmits a Frame Header that contains the frame ID. Because this ID is known by each slave, one slave recognizes the header as a request and answers **(Figure 3)**.

The slave tasks recognize their respective role from the identifier: One slave task recognizes that it must now provide the data for the requested ID in the Frame Response. Other slave tasks see that they must evaluate the antici-

Figure 2: LIN Master and LIN Slaves

Figure 3: LIN Communication Cycle

pated data. The remaining slave tasks are relieved to find out they do not have to do anything.
Together, the Frame Header and Frame Response yield a LIN frame. As in CAN, the ID contains a message addressing and every LIN ECU may read the data (broadcast).

Data Transmission Using LIN Frames

Because there is no communication controller, data is transmitted in LIN via the serial interface (Serial Communication Interface – SCI) of the microcontroller. Sending occurs bitwise in CAN and bytewise in LIN. The SCI sends every byte with the least significant bit (LSB) first and frames it with a dominant start bit (0) and a recessive stop bit (1). This is called an SCI frame. A LIN frame is therefore composed of a number of SCI frames that, because of the start bit, begin with a falling edge for synchronization **(Figure 4)**.
The header contains more than just the ID, however. Due to the inexpensive, imprecisely calibrated RC clocks, the LIN Slaves can only comprehend an ID after they have undergone a sufficiently precise calibration process.

LIN Becomes Comprehensible Only After a Calibration

For calibration, the LIN Master first sends a Sync Break in order to inform all LIN Slaves of its start. With at least 13 dominant bits and 1 stop bit, the length of the Sync Break is deliberately overexpanded in such a way that even the slowest LIN Slave must detect an SCI error. Even in the case of 12 dominant bits plus 1 stop bit, a slower LIN Slave with the allowed tolerance deviation of 14 % would still see a completely "normal" SCI frame. However, an error will be detected in the 13th dominant bit because the recessive stop bit would normally have to follow at this time. As a result of this error, the LIN Slave switches to a mode in order to measure the subsequent Sync Byte. It contains the value 0x55, thus a sequence of zeros and ones whose falling edges are measured to determine the sending speed of the master. At the end, all LIN Slaves are calibrated to the master with ±1.5 % accuracy.

Data is not as Protected as in CAN

The ID that now follows consists of six bits. Two parity bits are used for error detection in the transmitted ID. Together, the ID and the two parity bits are referred to as Protected Identifier (PID). The first 60 IDs (0 to 59) are available for payload data communication. ID=60 and ID=61 are for diagnostics. The last two identifiers, ID=62 and ID=63, are reserved.
The Frame Response consists of at least one and no more than eight data bytes, followed by an 8 bit checksum for error detection. The classic checksum was introduced in the first LIN versions. Starting in LIN Version 2.0, the extended checksum is used.
The classic checksum corresponds to the inversion of the modulo 256 sum of all data bytes. A receiver also generates the modulo 256 sum of the incoming bytes and as a last step adds the incoming checksum. If the result is not 0xFF, an error is present. The extended checksum includes the PID in the calculation.

Figure 4: LIN Frame

Ample Pauses for Slow Microcontrollers

Low-cost microcontrollers are simple and often slow. For this reason, LIN allows for a small delay before beginning the transmission of the Frame Response (Response Space) and also for send pauses (Interbyte Spaces) between the individual SCI frames. In this way, the Frame Response may be prolonged by 40 % overall. The same applies to the Frame Header. However, a LIN Master does not need any pauses, because in the vast majority of cases that ECU is also connected to a CAN bus. The CAN-side requirements do not allow the use of inexpensive microcontrollers. The master uses the 40 % more time afforded to it on a longer Sync Break: It cuts its clock pulse rate in half so that 18 dominant bits plus two recessive bits emerge. The time reserve of 40 % must be taken into account in the design of the LIN schedule.

Four Types of Frames

A rigid time pattern is specified through the LIN schedule. Each header that is sent according to the LIN schedule must be answered with a LIN response without any further condition. For this reason, these types of LIN frames are called Unconditional Frames.
However, the LIN specification allows this rigidity to be loosened and frames to be sent only as needed. The two frame types "Sporadic Frame" and "Event Triggered Frame" are available starting from LIN Version 2.0.
A Sporadic Frame is an Unconditional Frame that shares the same Frame Slot with other Unconditional Frames and is transmitted by the LIN Master as needed (header and response). Collisions are excluded because the master, as the sole ECU, is the only one that may send these types of frames. If there is no need to send the frame on the part of the master, the corresponding Frame Slot remains empty.
Event Triggered Frames communicate sporadic changes or events on the side of the LIN Slaves. They also correspond to Unconditional Frames that share a common Frame Slot. But the difference is that multiple Frame Responses of different LIN Slaves are assigned to a common Frame Header. Which slave answers the Event Triggered Frame Header with a response depends on the need of the corresponding LIN Slaves. A need exists when new data are to be transmitted. So that the receiver knows which data it receives, the PID of the underlying Unconditional Frame is transmitted in the first byte of the Frame Response of the Event Triggered Frame.
If multiple slaves send, a collision follows for which the LIN Master has responsibility for resolving. It interrupts the schedule after the collision and inserts another schedule containing all Event Triggered Frames that share the unlucky frame slot.
For diagnostics of LIN Slaves, LIN provides two reserved Unconditional Frames. You have ID=60 for the Diagnostic Request and ID=61 for the Diagnostic Response. LIN uses the ISO-TP transport protocol [3] customary in CAN as well as consistent diagnostic services [4, 5].

Status and Network Management

The LIN specification defines a status management and a network management.
For LIN Slaves as of LIN 2.0, the status management prescribes that detected transmission errors such as parity and checksum errors must be indicated to the LIN Master. For this purpose, a bit is needed in an Unconditional Frame that the slave sends to the master (Status Frame). Additional information about the nature of the error is possible but not mandatory. There are also not any instructions on how to handle errors. The developer is responsible for this.
The task of the LIN network management is primarily to put the slaves to sleep and wake them up or, in other words, to bring about the transition from normal communication mode (operational mode) to sleep mode, and vice versa **(Figure 5)**. If the LIN Slaves do not detect any bus activity for four seconds, they may switch from operational mode

Figure 5: LIN Network Management

to sleep mode. In LIN 2.1 and higher, they must have done so within 10 seconds. With a sleep command, the LIN Master can switch its slaves to sleep mode at any time.
Conversely, a LIN Slave runs through an initialization to switch from sleep mode to operational mode when it detects a wake-up signal followed by a valid header. The wake-up signal consists of a dominant pulse of at least 250 microseconds and no more than 5 milliseconds duration and can be sent not only by the master but from any LIN ECU. The LIN specification limits the initialization phase to 100 milliseconds. This means that the LIN Master must begin with a header after this time span at the latest. If it remains passive, the corresponding LIN Slave sends another wake-up signal. The intervals and number of repetitions of these signals are also specified.

Ask the Experts

Our specialists at Vector [6] support manufacturers and suppliers in the automotive industry and other industries not only in LIN networking but also in communication systems such as CAN, FlexRay, CAN FD, and Ethernet.
We offer consistent tool chains for planning, design, development, and maintenance as well as software components and basic software for AUTOSAR ECUs. For an introduction to ECU networking, we offer training seminars on the basics of CAN, LIN, FlexRay, and Ethernet. We teach the necessary skills for handling the above named tools in workshops, whose goal is to quickly familiarize engineers and technicians with all aspects of the multifaceted development tasks concerning electronics in automobiles [7].
In part 1 of this series [8], we addressed serial bus systems in general. CAN was addressed in part 2 [9]. The serial bus systems FlexRay and MOST will be addressed in the remaining articles in the series. Interested readers will find additional in-depth information on the previously published topics on the Vector E-Learning platform [10].

Literature References:

[1] www.lin-subbus.org
[2] LIN Specification Package Revision 2.2A
[3] Road vehicles – Diagnostics on Controller Area Network (CAN) – Part 2: Network layer services. International Standard ISO 15765-2.4, Issue 4, 2002-06-21.
[4] Road vehicles – Diagnostics on Controller Area Network (CAN) – Part 3: Implementation of diagnostic services. International Standard ISO 15765-3.5, Issue 5, 2002-12-12.
[5] Road vehicles – Diagnostic systems – Part 1: Diagnostic services. International Standard ISO 14229-1.6, Issue 6, 2001-02-22.
[6] www.vector.com
[7] www.vector-academy.com
[8] Christmann, E.: Data Communication in the Automobile – Part 1: Architecture, Tasks, and Advantages
[9] Christmann, E.: Data Communication in the Automobile – Part 2: Reliable Data Exchange with CAN
[10] elearning.vector.com

Ernst Christmann, Physicist, Mathematician
is a Senior Technical Trainer in the area of software tools for ECU testing and development and has worked for Vector Informatik GmbH in Stuttgart since 2004.

VECTOR

Experience With Series Projects.
Count on the Worldwide Leading Solution for LIN!

Solutions for LIN

Vector has the most comprehensive solution for every phase of your professional LIN network development:

> Design your LIN networks systematically
> Simulate, analyze and test ECUs and entire networks efficiently with CANoe and the LIN Interfaces
> Use our reliable LIN software components
> Make your ECUs production-ready with calibration, stress and logging tools from Vector
> Rely on our Vector service teams for development, training and support

Visit us at: **www.lin-solutions.com**

For successful projects, take advantage of proven tools, scalable modules and over 25 years of networking expertise at Vector!

Vector Informatik GmbH | Germany · Austria · Brazil · China · France · India · Italy · Japan · Korea · Sweden · UK · USA | www.vector.com

Serial Bus Systems in the Automobile

FlexRay for Data Exchange in Safety-critical Applications

FlexRay is going into production for the first time. It will appear on the BMW X5, which was presented to the public at the Paris Auto Salon in August 2006, and it can be purchased in Germany beginning in March of this year. Within its active chassis system, FlexRay provides for secure and reliable data transmission between the central control module and the four satellite ECUs, one located at each shock absorber. This article traces FlexRay's path into the automobile and explains the key principles of FlexRay bus technology.

According to the German Federal Statistics Office [1] driving on Germany's roads was never so safe as in the year 2005. Although vehicle registrations grew considerably, there was a nearly one percent reduction in accidents involving personal injury (336619) compared to the prior year. There were also significant reductions in the number of traffic deaths (5361, -8.2%), serious injuries (76952, -4.6%) and minor injuries (356491, -1%). This trend was continued in 2006: Between January and August, 3260 traffic participants were killed, and this represents a 7.8 percent reduction compared to the prior year. The number of injured dropped by 5.8 percent over the same time period.
Decisive in lowering the number of accidents and reducing the severity of accident outcomes are active safety systems and assistance systems that support drivers in their task of driving the vehicle. One study by a number of well-known automotive OEMs showed, for example, that ESP reduced the number of skidding accidents by up to 80% [2]. Making just as important a contribution to reducing the severity of accident outcomes are increasingly safer passenger cells and optimized restraint systems.

In light of the goal of halving traffic fatalities by the year 2010, the automotive industry is focusing on further developing existing active safety systems and driver assistance systems and developing new innovative systems. Since these systems not only provide information and instructions, but often also make corrective interventions and assume driving tasks, it is no longer possible to do without electronic interfaces to the chassis and drivetrain. The combination of brake-by-wire and steer-by-wire systems is thought to have great potential.

Requirements of Future Data Transmission in the Automobile

Implementations of ever more challenging safety and driver-assistance functions go hand in hand with the increasingly more intensive integration of electronic ECUs in the automobile. These implementations require very high data rates to transmit the increasing number of control and status signals. They are signals that not only need to be transmitted extremely quickly; their transmission also needs to be absolutely deterministic.

That is the reason for the growing importance of communication systems that guarantee fast and deterministic data transmission in the automobile. Potential use of by-wire systems further requires the design of fault-tolerant structures and mechanisms. Although by-wire systems may offer wide-ranging capabilities and the benefits of increased design freedom, simplified assembly, personalization of the vehicle, etc., data transmission requirements in the automobile are elevated considerably, because these systems belong to the class of fail-operational systems. They must continue to operate acceptably even when an error occurs. CAN cannot satisfy these requires due to its event-driven and priority-driven bus access, its limited bandwidth of 500 KBit/sec based on physical constraints in the automobile, and lack of fault-tolerant structures and mechanisms [3].

FlexRay – The Answer to Heightened Data Transmission Requirements in the Automobile

The certainty that CAN could hardly be expected to satisfy growing data transmission requirements in the automobile over the mid-term, led to the development of a number of deterministic and fault-tolerant serial bus systems with far greater data rates than CAN. Examples include: TTP (Time Triggered Protocol) [4], Byteflight [5] and TTCAN (Time Triggered CAN) [6]. Although a development partnership was created as early as 2001 between Audi and the TTP promoting company TTTech, and although Byteflight was successfully applied to BMW 7-series cars in 2001, it is FlexRay that has prevailed in the automotive industry.
An important reason for the success of FlexRay was the founding of the FlexRay Consortium [7], under whose auspices the two motor vehicle manufacturers DaimlerChrysler and BMW and the two chip producers Motorola and Philips joined forces in the year 2000. Based on Byteflight bus technology originally developed by BMW, the FlexRay Consortium created the cross-OEM, deterministic and fault-tolerant FlexRay communication standard with a data rate of 10 MBit/sec for extremely safety- and time-critical applications in the automobile.
Today the FlexRay Consortium is made up of seven "core partners": BMW, Bosch, DaimlerChrysler, Freescale, General Motors, Philips and Volkswagen. Gradually, a number of Premium Associate Members (including Vector Informatik [8]) and Associate Members joined the organization.
Making a significant contribution to the success of FlexRay was the detailed documentation of the FlexRay specification. The two most important specifications, the communication protocol and the physical layer, are currently in Version 2.1. These and other FlexRay bus technology specifications can be downloaded from the homepage of the FlexRay Consortium [7].

FlexRay Communication Architecture – Time-triggered, Fault Tolerant and Flexible

Just as in the case of data communication in a CAN cluster, data communication in a FlexRay cluster is also based on a multi-master communication structure. However, the FlexRay nodes are not allowed uncontrolled bus access in response to application-related events, as is the case in CAN. Rather they must conform to a precisely defined communication cycle that allocates a specific time slot to each FlexRay message (Time Division Multiple Access – TDMA) and thereby prescribes the send times of all FlexRay messages **(Figure 1)**.
Time-triggered communication not only ensures deterministic data communication; it also ensures that all nodes of

Figure 1: Principle of FlexRay communication.

a FlexRay cluster can be developed and tested independent of one another. In addition, removal or addition of FlexRay nodes in an existing cluster must not impact the communication process; this is consistent with the goal of re-use that is often pursued in automotive development.
Following the paradigms of time-triggered communication architectures, the underlying logic of FlexRay communication consists of triggering all system activities when specific points are reached in the time cycle. The network-wide synchronism of FlexRay nodes that is necessary here, is assured by a distributed, fault-tolerant clock synchronization mechanism: All FlexRay nodes not only continuously correct for the beginning times (offset correction) of regularly transmitted synchronization messages; they also correct for the duration (slope correction) of the communication cycles **(Figure 2)**. This increases both the bandwidth efficiency and robustness of the synchronization.
FlexRay communication can be based on either an electrical or optical physical layer. Speaking in favor of electrical signal transmission is its simplicity, which brings cost advantages. The comparatively cost-intensive optical signal transmission is characterized by substantially better electromagnetic compatibility (EMC) compared to electrical signal transmission.
FlexRay communication is not bound by a specific topology. A simple, passive bus structure is just as feasible as an active star topology or a combination of the two **(Figure 3)**. The primary advantages of the active star topology lie in possibility of disconnecting faulty communication branches or FlexRay nodes and - in designing larger clusters - the ability to terminate with ideal bus terminations when physical signal transmission is electrical.
To minimize failure risk, FlexRay offers redundant layout of the communication channel **(Figure 4)**. This redundant communication channel could, on the other hand, be used to increase the data rate to 20 Mbit/sec. The choice between fault tolerance and additional bandwidth can be made individually for each FlexRay message.
Finally, an independent control mechanism (Bus Guardian) ensures that a FlexRay node only gets access to the bus during its turn in the communication cycle. This prevents bus monopolization by a defective FlexRay node (babbling idiot).

FlexRay Communication: Deterministic and Dynamic

Each communication cycle is equal in length and is essentially organized into a static time segment and a dynamic time segment **(Figure 1)**. Of central importance here is the static segment that begins each communication cycle. It is subdivided into a user-definable number (maximum 1023) of equally long static slots.
Each static slot is assigned to a FlexRay message to be sent by a FlexRay node. Assignments of static slots, FlexRay messages and FlexRay nodes are made by slot number, message identifier (ID), and the value of the slot counter implemented on each FlexRay node. To ensure that all FlexRay messages are transmitted at the right time and in the correct sequence in each cycle, the slot counters on all FlexRay nodes are incremented synchronously at the beginning of each static slot. Because of its guaranteed equidistant and therefore deterministic data transmission, the static segment is predestined for the transmission of real-time relevant messages.
Following the static segment is an optional dynamic segment that has the same length in every communication cycle. This segment is also organized into slots, but not static slots, rather so-called minislots **(Figure 1)**. Communication in the dynamic segment (mini-slotting) is also based on allocations and synchronous incrementing of the slot counters on the FlexRay nodes.

Figure 2: Clock synchronization.

Figure 3: Combined topology of passive bus and active star.

Figure 4: Passive bus structure with two communication channels minimizes failure risk.

However, it is not mandatory to transmit the FlexRay messages associated to the minislots with each communication cycle, rather they are only sent as needed. If messages are not needed, the slot counter of a minislot is incremented after the defined time period. While a (dynamic) FlexRay message is being transmitted, incrementing of the slot counter is delayed by the message transmission time **(Figure 5)**.
The allocation of a dynamic FlexRay message to a minislot implicitly defines the priority of the FlexRay message: The lower the number of the minislot, the higher the priority of the dynamic FlexRay message, the earlier it will be transmitted, and the higher the probability of transmission given a limited dynamic time segment length. The dynamic FlexRay message assigned to the first minislot is always transmitted as necessary, provided that there is a sufficiently long dynamic time segment.
In the communication design it must be ensured that the lowest priority dynamic FlexRay message can be transmitted too – at least provided that there are no other, higher priority needs. The designer of a FlexRay cluster must also ensure that transmission of the longest dynamic FlexRay message is even possible. Otherwise, the communication design would not make any sense.

The communication cycle is completed by two additional time segments (Figure 1). The "Symbol Window" segment serves to check the functionality of the Bus Guardian, and the "Network Idle Time – NIT" time segment closes the communication cycle. During the NIT the FlexRay nodes calculate the correction factors needed to synchronize their local clocks. At the end of the NIT, an offset correction is made if necessary (the slope correction is always distributed over the entire communication cycle). There is no data transmission during the NIT.

CRC-protected Data Transmission

The signals in a FlexRay cluster are transmitted by the well-defined FlexRay message, wherein there is essentially no difference in the formats of the FlexRay messages transmitted in the static segment and those transmitted in the dynamic segment. They are each composed of a header, payload and trailer **(Figure 6)**.
The header comprises the five-bit wide status field, ID, payload length and cycle counter. The header-CRC (11 bits) protects parts of the status field, ID and payload length with a Hamming distance of 6. The ID identifies the FlexRay message and represents a slot in the static or dynamic segment. In the dynamic segment the ID corresponds to the priority of the FlexRay message. The individual bits of the status field specify the FlexRay message more precisely. For example, the "sync frame indicator bit" indicates whether the FlexRay message may be used for clock synchronization.
After the header comes the so-called payload. A total of up to 254 useful bytes may be transported by one FlexRay message. The trailer encompasses the header and payload-protecting CRC (24 bit). Given a payload of up to 248 useful bytes, the CRC guarantees a Hamming distance of 6. For a larger payload the Hamming distance is 4 [8].

Figure 5: Example of communication flow in the dynamic time segment.

Figure 6: Structure of the FlexRay message with header, payload and trailer.

In Networking Issues: Achieving Objectives Rapidly with External Expertise

In the year 2001, Vector Informatik was already offering the first product solution for the development of FlexRay systems. In the meantime, developers can now obtain a comprehensive portfolio of products [9]. These include tools for designing, developing, simulating, analyzing, testing and calibrating ECUs and distributed networks. DaVinci Network Designer FlexRay gives the developer an environment for efficiently designing network architecture and communication relationships. Simulation, analysis and testing of FlexRay systems are performed with CANoe. FlexRay, whose multibus concept enables simultaneous operation of FlexRay, CAN, LIN and MOST bus systems. For precise study of the FlexRay network's system behavior in response to errors and disturbances, FRstress generates them on a channel in the FlexRay cluster. For direct access to internal ECU variables, the developer needs a special measurement and calibration protocol: XCP on FlexRay. In the context of the development of the active chassis system on the new BMW X5, BMW engineers implemented Vector's measurement, calibration and diagnostic tool CANape. As the XCP-on-FlexRay Master, CANape measures and calibrates individual ECU parameters directly via FlexRay. Besides software, Vector also develops stacks for ECUs. FlexRay software components make it possible to interconnect applications with different bus or operating systems in an uncomplicated way. For hardware access to FlexRay buses, suitable bus interfaces connect to the USB, PCI and PCMCIA ports of a PC or notebook computer.
The Vector Academy [10] can teach the basic knowledge needed to quickly become familiar with the diverse development activities related to ECU communication in the automobile. This knowledge is shared in the context of seminars on CAN, LIN, FlexRay and MOST.

Literature and links:

[1] www.destatis.de
[2] www.bosch.com
[3] Mayer, E.: Datenkommunikation im Automobil – Teil 2: Sicherer Datenaustausch mit CAN im Automobil ["Data communication in the automobile – Part 2: Reliable data exchange with CAN in the automobile"]. In: Elektronik Automotive 8/2006, pp. 34-37
[4] www.tttech.com
[5] www.byteflight.com
[6] www.can-cia.org/can/ttcan
[7] www.vector-informatik.com
[8] www.flexray.com
[9] www.flexray-solutions.com
[10] www.vector-academy.com

Eugen Mayer
(Graduate Engineer with Technical Teaching Certificate), after completing his vocational training to become a communications technician, studied electronics at the Technical College in Ravensburg/Weingarten, Germany, and studied electrical engineering and vocational teaching at the University of Stuttgart. Since 1999 he has been working at Vector Informatik where he is employed as a Senior Trainer.

VECTOR

Experience With Series Projects.
Choose Proven FlexRay Solutions!

Solutions for FlexRay

Take advantage of our extensive experience with FlexRay series projects. Use Vector's comprehensive product portfolio for your FlexRay networking:

> Simulate, diagnose and test ECUs and networks with the sophisticated development environment CANoe and the FlexRay interfaces
> Profit from standardized ECU software. Vector software modules can be flexibly configured and easily integrated.
> Ensure advantages in both quality and time: Rely on professional support during development and training.

Get on board: **www.flexray-solutions.com**

Get FlexRay on the road – with series-proven tools, scalable modules and over 25 years of Vector networking know-how!

Vector Informatik GmbH | Germany · Austria · Brazil · China · France · India · Italy · Japan · Korea · Sweden · UK · USA | www.vector.com

Wireless analysis in a multi-protocol CAN environment

Timo Löw and Andreas Nacke (both Bomag), Holger Heit and Hans-Werner Schaal (both Vector Informatik)

In developing electronics for modern construction equipment, a large share can even be tested and simulated meaningfully on test benches. In later development stages, on the other hand, it is preferable to perform tests and trial runs under real conditions at construction sites or outdoor test sites. To avoid distracting the operator in the driver s cabin with test equipment, a wireless interface has now been realized for the first time with the CANoe and CANalyzer development and analysis tools from Vector Informatik. Electronics developers at Bomag GmbH now use this interface to log the communication on various vehicle buses at a distance and analyze it. Quality requirements in earthmoving work and highway construction are continually growing with a simultaneous rise in cost and time pressure. Soil compaction and cost-, raw material- and energy-saving methods of road preservation and reconstruction are often only possible with intensive use of high-tech electronic functions. Bomag is the global market leader in the field of compaction technology. At its lead-plant in Boppard, this company – part of the Fayat Group – produces about 30 000 machines annually for soil, asphalt and garbage compaction as well as stabilizers/recyclers. Today, a large share of the company s expertise has its foundation in electronics.

When it comes to networking technology, Bomag bases its work on the CAN bus standards of the automotive industry. Initially, the electronics concept was established on the large 10-t to 15-t machines, and it was then ported to the smaller machines. Since Bomag would like to have hardware and software components be reused as often as possible within the overall corporate group, the focus was on a modular concept. This also required standardization of development and test equipment across their business sites.

Fig. 1: The electronics on the MPH 125 support efficient implementation in soil stabilization and recycling

Nothing works without electronics

High-tech equipment and electronics can be found everywhere in the machines, from remote controls to drive-by-wire steering systems to the use of GPS. Sensors continually acquire the soil composition, and the display graphically indicates to the driver where compacting work is still needed. The GPS option enables satellite-supported, large-scale compaction monitoring with seamless documentation of all parameters. In the future, radio networks will provide for data exchange between the machines driving in a group. The new type MPH 125 stabilizer/recycler – with an operating weight of 24.5 t and a power of 440 kW – is the machine with the most extensive electronics system and the most CAN nodes. First, it is used to improve and reinforce existing soil materials by mixing in lime, fly ash or cement, and secondly for milling, pulverizing and recycling existing materials in-place and locally.

Network cluster with multiple CAN buses

All Bomag machines of a given product line have the same control system, and they acquire their specific control functionality in end-of-line parameterization. Therefore, electronic developers mapped the machine s modular product concept to a modular CAN-based network cluster. CAN 1 – as the central Body-CAN bus – is connected to the most bus nodes. Its operation is based on the CANopen protocol, which enables the use of standard ▶

Fig. 2: The electronics concept reflects the modular layout of the machines

Fig. 3: J1939-specific interpretation of data in the Trace Window is performed with CANalyzer.J1939

automation components. Besides the vehicle s main computer, the data acquisition unit of the front frame and I/O module of the rear frame, CAN 1 nodes also include control and display components in the cockpit. Conventional analog and digital sensors are interfaced to the data acquisition unit, e.g. hydraulic pressure sensors and fill level sensors. The I/O module on the rear frame is responsible for control of the variable-height rotor, lateral tilt angle and lowering cabin feature for transport purposes. It was possible to significantly reduce wiring cost and effort by interfacing controls to the bus; these include the CAN-bus-capable joysticks, LC displays and external switches. Bomag created an in-house development of a CANopen driving lever with friction brake.

The powertrain bus is defined as CAN 2, and it interconnects the vehicle s main computer, engine controller, steering and driving levers, including operator consoles on the right and left. Interesting here is that the J1939 and CANopen protocols are implemented in parallel on CAN 2. A special feature of the drive control system is load-limit control of the Deutz diesel drive, which provides for dynamic power distribution between higher milling power at low speed and lower milling power at high working speed as a function of soil composition.

Besides CAN 1 and CAN 2 there may be a third CAN 3 data bus, depending on how the MPH 125 is equipped. It incorporates an optional metering computer with auxiliary display and the necessary actuators for water injection. Similarly, CAN 3 is needed to interface to the metering system for bitumen emulsion and foamed bitumen.

Multi-protocol capable tools

In electronics development, Bomag implements a number of software tools from Vector Informatik. The Stuttgart-based networking specialist offers tailor-made tools for all electronic development tasks, such as CANoe for network development and ECU tests, CANalyzer for analysis of bus data and CANape for calibrating ECUs. At the beginning of development, CANoe simulates the behavior of individual bus nodes or entire sub-networks (rest-of-bus simulation). Over the further course of ECU development, the models serve as a foundation for analysis, testing and integration of bus systems and ECUs. The C-like programming language CAPL and user-defined interfaces simplifies the user s work. A special real-time configuration significantly improved real-time capabilities even further, first by using separate PCs for rest-of-bus simulation and test execution, and second by graphic control/evaluation; this yielded the high performance of a component HIL tester. ▶

The CANalyzer analysis tool offers numerous functions for bus analysis. They range from tracing the bus data traffic to displaying signal values, performing statistical evaluations of messages, busloads and disturbances, and finally targeted generation of specific bus disturbances.

CANape is used in the calibration of electronic ECUs. All important variables and parameter sets can be modified and visualized in real time. Helpful in conjunction with the development of GPS applications is CANape Option GPS, which supplements the system with visualization of the momentary vehicle positions on an electronic map.

The universality of Vector tools is paying off at Bomag by helping it to master the complex challenges of working with multiple buses, and in particular the J1939/CANopen multi-protocol environment. The consistently applied database concept is an important pillar for the efficiency of the development tools. All members of the tool chain access the same data set, so that it is possible to save all data consistently without unnecessary redundancies or sources of error. Fitting for the bus systems used, the relevant databases of the network description are either already integrated or automatically generated to match the network configuration.

Besides supporting CAN, the tools also support the LIN, FlexRay and MOST buses as well as the higher level protocols J1939, J1587, NMEA2000, ISO11783 and CANopen. In the case of Bomag, CANape and the CANalyzer/CANoe options for J1939 and CANopen are used. Protocol-specific extensions of the tools relieve the user of the need for detailed training in the J1939 or CANopen protocol; this avoids faulty interpretations of CAN frames. Last but not least, another important requirement in the analysis of muli-bus/multi-protocol environments is that a uniform time base must always be present as a foundation for analyses.

No need for an umbilical cord

What has been difficult for Bomag electronics developers to achieve until now is time-synchronous analysis of the measurement data during the field tests mentioned in the introduction without having to be passengers in the machine. They were only able to examine the logged data afterwards, but not during a test. For these and similar cases, there is now a technically mature WLAN solution from Vector: While previously it was absolutely necessary to maintain physical contact to the bus system being analyzed when working with the tools, the specialists from Stuttgart have now made it possible to establish contact with the DUT by an extension via a modified WLAN system. So the transceiver cable between PC and CAN bus is quasi removed and replaced by the radio pathway. Noteworthy here is the fact there are no significant compromises with regard to accuracy or measurement requirements. In implementing the extension, special attention was given to satisfying requirements for correct timing in data transmission, low latency times and time-synchronous display of the data on the PC. The CAN messages – together with their time stamps – are tunneled via a TCP/IP connection, so that the time stamps provided with the messages can serve as reference times for CANoe and CANalyzer.

Fig. 4: The extended WLAN tunnels the CAN messages, including time stamp, via a TCP/IP interface, thereby enabling time-conformant representation of the data

Drilled out WLAN solution

This solution offers some key advantages compared to the capabilities of a simple CAN/WLAN bridge.

Only a bridge header is necessary for this setup. Sufficient as a host is a WLAN-capable laptop/notebook, which maintains the connection via standard on-board resources and WLAN. The "probe" on the DUT that is responsible for converting from CAN to WLAN sends the messages in strict and chronologically-correct order by considering the time stamps originally logged on the bus, which would not be possible via a CAN-WLAN-CAN bridge. During machine operation at the construction site, the Bomag electronics developers can measure, observe and evaluate without a hard wire connection to the machine.

Summary and outlook

This example from the commercial vehicle sector shows that there are interesting and demanding challenges outside of the realm of automotive electronics for luxury cars, challenges that can only be handled by complex network clusters and high-performance development and analysis tools. The universality of Vector solutions pays off here. The tools enable analyses and simulations directly on the higher layers of J1939 and CANopen. The use of multiple buses or simultaneous use of different protocols on the same bus do not present any problems. The tools always ensure correct timing with the same time base in data acquisition and evaluation – even in the case of wireless communication. The Bomag developers already have their sights set on the next WLAN project involving automatic, multi-day durability tests. Thanks to a WLAN connection, the electronics developers are able to continually observe events on the relevant bus systems with the mentioned tools. That can be done from the office or via the Internet, e.g. from home during the weekend.

Sources: Fig.1 and 2: Bomag GmbH, Fig. 3 and 4: Vector Informatik GmbH

holger.heit@vector-informatik.de
timo.loew@bomag.com
andreas.nacke@bomag.com
hans-werner.schaal@vector-informatik.de

VECTOR

Ethernet Control Units
Develop them efficiently

... with the Automotive Ethernet solutions from Vector

Our solutions support the specific physical layer in automotive as well as the protocols SOME/IP, AVB, DoIP, etc.

> Tools for simulating, analyzing and testing of Ethernet networks and controllers – also together with other vehicle bus systems
> Interfaces for undistorted access to Ethernet networks
> Embedded Software with small footprint, for automotive use
> Universal control unit for small production runs, evaluation and development – complete with AUTOSAR basic software
> Training on Ethernet technologies in the automotive domain

More information & downloads: **www.vector.com/ae**

Benefit from our experience in automotive electronics.

Reference Chart Automotive Ethernet
Useful know-how, free-of-charge,
order now: **www.vector.com/eth-poster**

Vector Informatik GmbH | Germany · Austria · Brazil · China · France · India · Italy · Japan · Korea · Sweden · UK · USA | www.vector.com

Acquiring Bus Data Wirelessly from Multiple Vehicles

Top-notch Precision Despite the "Air Interface"

In the past, analysis of vehicle communications has generally been limited to a single vehicle. This situation is changing with the development of new driver assistance systems, and it is of primary importance to Car2x technologies. A current project – involving validation of an ACC system at Daimler AG – represents the first time that bus data from two test vehicles have been logged time-synchronously with wireless technology and displayed in real-time on the display of a stationary control station.

As specialized bus systems such as LIN, MOST and FlexRay began to supplement existing CAN networks, the need arose for development tools that could handle multibus systems. One of the key objectives – that is still relevant today – was to use one analysis tool to acquire and display time-synchronous messages from different systems and bus branches networked via gateways.

New Territory: Acquiring CAN Data from Multiple Vehicles Timesynchronously

Current developments in the area of driver assistance systems are leading to impressive growth in functionality. They range from simple convenience functions to safety-related systems such as braking assistants and intelligent adaptive cruise control (ACC). These systems utilize sensors based on technologies such as microwave radar, infrared light or ultrasound, and they still operate autonomously from the perspective of a single vehicle. Now, development and testing departments are, for the first time, being confronted with a situation where multiple vehicles are involved in events.

Even when just one vehicle is considered, validation faces many challenges [1]. It is no longer sufficient to consider each vehicle separately; rather, it is necessary to consider CAN bus information and measured values from all of the vehicles involved.

The assumption that has been made until now is of a wire-bound physical connection to the mobile computer traveling in the vehicle that is by its nature restricted to one vehicle. The "air interface" is a nearly insurmountable obstacle to causal and time-accurate representation of communication involving multiple vehicles. So far, the fulfillment of key preconditions for realizing the dream of "untethered" bus data acquisition has been lacking.

Challenge of Synchronous Measurement Despite the "Air Interface"

Measurement technology specialists at Daimler AG are tackling issues related to synchronous wireless measurement data acquisition. A traditional forerunner in the development of innovative technologies, Daimler conducts complex field tests to validate driver assistance systems. Guiding such complex systems to market maturity requires numerous trials and driving tests as an indispensible part of the development process. The assistance systems must repeatedly prove their reliability under all conceivable constraints and conditions **(Figure 1)**.

In the pursuit for improved efficiency in a current ACC project, the need arose for time-synchronous wireless logging of CAN data, both of the vehicle under test and the target vehicle involved in the traffic event. In particular, time-synchronous logging of position and speed information were of great interest in evaluating and documenting during the test drives, since they enable real-time visualization from a control center located on the perimeter of the test track. Evaluating and optimizing the control algorithms requires knowledge – as precise as possible – of relative distances, speeds, transverse accelerations, etc. Where were the vehicles exactly? What was the latency time of the assistance system's reaction? And even more important: does the system only react when it should react?

Driving Robots Assure Reproducible Maneuvers

Validation of assistance systems involves an enormous amount of technical effort. Fractions of seconds are involved in deciding the success or failure of actions and reactions of the vehicles, and driving maneuvers must be absolutely reproducible. So, it is not only out of concern for safety that Daimler is utilizing driving robots instead of human test drivers **(Figure 2)**. These robots can handle all necessary driving functions such as accelerating, braking, steering, etc., and they even execute the tightest of pass-by maneuvers and "near crashes" precisely and without any emotion.

Figure 2: Driving robots handle all necessary functions such as accelerating, braking, steering, etc., in reliable validation of driver assistance systems.

Jürgen Luther and his colleagues in Research and Advanced Development are responsible for establishing new test methods for assistance systems throughout Daimler and for providing the necessary measurement equipment for test trials. The driving robots that are used are able to drive trajectories based on a differential GPS signal with a maximum lateral deviation of less than five centimeters and a time deviation in the 100 millisecond range. In ACC tests, this makes it possible to have two vehicles with different – sometimes quite fast – approach velocities drive past one another on predetermined paths with an extremely small lateral gap of just ten centimeters. Right and left lane-changing maneuvers are also tested with different velocities and gaps. Until now, Daimler developers lacked a method for establishing a time reference between the dif-

Figure 1: Typical test scenarios with vehicle under test and target vehicle, in which precise data synchronism is required:
A – Sudden swerve into lane, e.g. for ACC scanning and ranging
B – Close passing maneuver with hard braking, e.g. in brake assistants
C – Close encounter in cross traffic as benchmark for greatest precision demand

ferent CAN buses of the vehicle under test and the target vehicle. They could not be displayed together in the Trace Window. A time deviation of just 5 milliseconds between the two sets of vehicle data at a speed of 20 m/s would still represent a spatial offset of 10 centimeters, which is the required precision of the driving maneuver.

CAN over IP over WLAN ...

Utilizing the CANoe test tool – widely used in the automotive industry – and the Option .Ethernet available for it, Daimler worked together with Vector to come up with a solution for wireless and synchronous bus data acquisition of multiple vehicles. This solution gives Daimler employees the ability to acquire, save and visualize the CAN messages of both vehicles with a common time base **(Figure 3)**. CANoe .Ethernet is an extension of the standard version of CANoe with additional Ethernet-specific functions. For a long time now, the standard version has enabled efficient study of multiple vehicle buses beyond gateway limits. It let's users display information acquired with high precision in real time during test drives or stationary tests and analyze logged data at their convenience afterwards.

The IP program variant of CANoe, on the other hand, is used in development, simulation and testing of embedded systems in the IP area. This is different than in the office communication field, embedded systems involve distributed control loops that can now be served by an IP bus system instead of CAN or LIN. Under such constraints, CANoe can also be used to test and simulate such entities as IP nodes and gateways to graphically display and analyze the data in display windows.

For some time now, IP has also made an entry into the automotive field. For example, flashing is done over IP via one ECU to flash multiple ECUs in different bus systems. Also, developers encounter IP in WLAN form, because CAN messages can be sent over WLAN/ Ethernet when packed in Ethernet packets. Remote CAN is desirable wherever CAN bus systems are difficult to access or are mobile, such as in remote control or remote monitoring of engine dynamometers, HIL tests or wireless acquisition of CAN data [2].

IEEE 1588 Synchronizes Clocks

In cooperation with measurement technology specialists at Daimler, Vector extended the CANoe .Ethernet tool by adding the necessary functions for ACC tests at Daimler. The following solution approach was taken: If one makes the theoretical assumption that precisely synchronized clocks are available as time references in each of the two vehicles, all data packets can be sent out with highly precise time stamps and these time stamps can be used to clearly trace the data chronologically at the receiving end. However, in practice this does not succeed without additional measures, since synchronized clocks will drift apart from one another.

The primary focus here is on re-synchronizing the clocks and packing the messages with original time stamps from the remote CAN buses into WLAN packets. The method used for clock synchronization comes from an internationally recognized procedure standardized in IEEE 1588 [3] and designed for local networks such as Ethernet. While a reference clock, defined as the 'Master clock', supplies the 'Slave clocks' with synchronization messages, the transmission times (delays) are computed based on the response messages. This enables correction for the deviation (offset) in the next pass and control of the clock rate (drift), i.e. accelerating or slowing the oscillator. With progressive iterations, this approach converges toward synchronization. In this method, there is an agreement on which of the two times will be used as the reference, and the events of the other bus system are correctly mapped to this reference. This approach succeeds even in the case of astonishingly imprecise clocks, and in principle it can be applied to interface hardware produced by any manufacturer, provided that the latest version of CANoe .Ethernet or CANalyzer .Ethernet is used. For a general explanation and overview of the principles of this method, see [4].

Figure 3: Time-synchronous data transmission to CANoe .Ethernet over WLAN with a maximum deviation of 5 milliseconds.

Extended WLAN Range

Another problem had to be resolved as well – the problem of limited WLAN range. When commercially available WLAN antennas are used, range is limited to a maximum of 100 to 300 meters, depending on whether the antenna is located inside or outside the vehicle. These values were too low for test track use, so a special WLAN antenna with a high antenna gain of 15 dBi was used, which increased range to 1200 meters. These "large" antennas, which are 4 centimeters in diameter an 80 centimeters in length, are mounted to the car roof with magnetic feet; they also signal to all human test drivers that robotic vehicles are in use.

Thanks to CANoe's flexible and open concept – as well as the seamless teamwork of the two companies – Vector developers were able to make the necessary modifications and adaptations quickly. The results surpassed even the initial expectations of the customer. Despite the critical air interface, maximum time deviations of under five milliseconds were attained, which is more than sufficient for most applications. Using WLAN from the perimeter of the test track, Jürgen Luther and his colleagues now conveniently observe, evaluate, document bus traffic during the precise and coordinated maneuvers of the vehicles, and they can make immediately qualified assessments on the success of a test trial **(Figure 4)**.

Figure 4: CANoe Ethernet simulates a traffic scenario based on the example of a received skidding warning.

From Intelligent Vehicle to Car2x Participant

The pilot application at Daimler clearly shows the direction in which the focus of testing is moving. Advanced development of today's assistance systems in the direction of Car2x technologies has long begun. If one assumes that traffic density will continue to grow, intelligent vehicle communication will even become an absolute necessity to regulate the flow of traffic, better utilize roads and improve traffic safety. One indication of this is that the "CAR 2 CAR Communication Consortium" [5] is already working on standardization that will enable future communication between vehicles via Wireless LAN (WLAN) throughout Europe.

At automotive OEMs and suppliers, vehicle tests are increasingly being extended to cover more than one vehicle. The potential number of future Car2x applications and test scenarios is enormous. Vehicles further ahead in traffic can send warnings about hazardous situations to vehicles that follow. Intersection assistants can signal that a vehicle is hidden behind a building at a corner, indicating its position, driving direction and speed as shown in scenario C of Figure 1. This could be supported by beacons (also known as Road Site Units).

Test equipment that has been used so far for data acquisition in a single vehicle is reaching its ultimate limits when faced with such development challenges. At the same time, the topics addressed here are of central importance to a large group of developers. In the future, typical timing issues must be resolved for multiple vehicles, e.g. in blind spot monitoring when the bus system of the lanechanging vehicle gets the information that it must not pull out into the neighboring lane.

Developers will have to become accustomed to the idea of log-ging and processing data with a uniform time base for multiple vehicle domains. This will require forging many new paths, and special attention must be given to the air interface in particular. The roles of Ethernet, WLAN and IP will also grow in importance in the automotive industry. To satisfy future Car2x requirements, Vector is continually advancing the development of CANoe in collaboration with its users. The next step is to support the IEEE 802.11p standard, which is important in establishing WLAN in vehicles.

Translation of a German publication in Elektronik automotive, 7/2010

Image rights:
Initial Screen: Vector Informatik GmbH
Figure 1: Vector Informatik GmbH, based on documents from Daimler AG
Figure 2: Daimler AG
Figure 3: Daimler AG and Vector Informatik GmbH
Figure 4: Vector Informatik GmbH

Literature:

[1] H. Hurich, Wolfgang, H., Luther, J., Schöner, Dr. H.-P.: Koordiniertes Automatisiertes Fahren zum Entwickeln, Prüfen und Absichern von Assistenzsystemen ["Coordinated Automated Driving in Developing, Testing and Validating Assistance Systems"], 10th Braunschweiger Symposium AAET 2009
[2] Löw, T.; Nacke, A.; Schaal, H.-W.: Für schweres Gerät – Drahtlose Anbindung von Entwicklungs- und Analysewerkzeugen ["For Heavy Equipment – Wireless Interfacing of Development and Analysis Tools"]. Elektronik automotive 2008, Issue 2, pp. 38ff.
[3] Introduction to IEEE 1588: http://ieee1588.nist.gov
[4] Wiebel, Prof. Dr. H.: Uhren mit IEEE 1588 synchronisieren ["Synchronizing clocks with IEEE 1588"], Bulletin SEV/VSE 2004, Issue 4, pp. 35ff, www.ines.zhaw.ch/fileadmin/user_upload/engineering/_Institute_und_Zentren/INES/IEEE1588/Dokumente/Fachartikel_IEEE1588_02.pdf
[5] CAR 2 CAR® Communication Consortium: www.car-to-car.org

Links:

Homepage Daimler AG: www.daimler.com
Homepage Vector: www.vector.com
Product Information CANoe .Ethernet: www.vector.com/vi_canoe_ethernet_en.html
Information Vector Solutions for Car-2-x: www.vector.com/car2x

Jürgen Luther, Daimler
studied Physical Technology at the Technical College of Lübeck. Mr. Luther is a scientific assistant in Research & Advanced Development at Daimler AG and is responsible for support and ongoing development of new testing methodology with driverless robotcontrolled vehicles.

Hans-Werner Schaal, Vector
studied Communications Engineering at the University of Stuttgart and Electrical & Computer Engineering at Oregon State University in Oregon, USA. Mr. Schaal is Business Development Manager for the Open Networking product line at Vector Informatik GmbH. Prior to this, he worked in various industries as development engineer, project leader and product manager in the test tools area for several network technologies.

IP and Ethernet in Motor Vehicles

Challenges for the Development Tool, Illustrated by Today's Applications

Until just a few years ago, the prevailing opinion was that Ethernet would never be used for in-vehicle applications, with the exception of diagnostic access. Soon, however, camera-based driver assistance systems will be the first applications to utilize Ethernet technology as a system network. This presents new challenges to automotive OEMs, suppliers and development tool producers, because the Internet Protocol and Ethernet represent a new network technology for motor vehicles. Nonetheless, many of the issues can already be solved.

After the debut of the CAN bus in the Mercedes S-Class in 1991, the LIN, MOST and FlexRay bus systems also became established in the motor vehicle. Today, CAN continues to be used in automotive network architectures in all domains (from powertrain to body). LIN bus technology is ideal for simple and cost-effective data exchange of noncritical signals in the convenience area. Where bandwidths and real-time requirements run into limitations, CAN is replaced by FlexRay or MOST – in cases where it is economically justifiable. In today's vehicles, one often finds all of the named bus systems, segmented and networked via gateways.

Motivation for Ethernet

Ethernet has long been an established standard technology in office communications, industrial engineering (ODVA standards, Ethernet/IP and ProfiNet) and in the aerospace industry (AFDX®). In the automotive field, Ethernet had already proven itself in the vehicle for diagnostic access. In recent years, other use areas have increasingly been discussed in automotive research and development departments, because Ethernet's, scalable bandwidth and flexibility spoke strongly in its favor. Nonetheless, a suitable and economical wiring technology was lacking for the motor vehicle.

Currently, the main drivers for Ethernet usage in the vehicle are camera-based driver assistance systems. In camera applications in the vehicle, LVDS technology (Low Voltage Differential Signaling) has been used until now. The shielded cable that is generally used there does indeed assure electromagnetic compatibility, but it is expensive by industry measures, and it is very impractical to install in the motor vehicle. Most recently, a physical layer is available that offers full-duplex transmission at 100 Mbit/s on a CAN-like, two-wire cable (unshielded twisted pair), and in the opinion of various publications it is suitable for use in the motor vehicle [1], [2], [3].

Requirements of an IP Development Tool

First, known requirements of previous bus systems still apply to the development tool. Initially, what is required is a detailed protocol analysis with stimulation option that extends to script-based testing with automatic generation of test reports. The user also expects that the market-proven multibus capability will of course be extended to include Ethernet and IP, so that dependencies between events on different bus systems can be studied. Currently, for example, there is interest in correlation between LIN and CAN, and in the future interest will be between CAN and IP.

As previously, in protocol analysis the user needs easy symbolic access to all relevant application signals as well as the ability to further process them in any desired way – logically and graphically. However, there will also be new requirements, which on the one hand are imposed by the bus physics and on the other by the wide variety of IP protocols. The article explains – based on the current camera example and four other application areas of IP and Ethernet in the motor vehicle – how these measurement tasks present themselves in product development departments from the perspective of the system manager, and which special requirements result for the development tool.

1. Camera – Ethernet as System Network

A camera-based driver assistance system at BMW will be the first production implementation in the motor vehicle to utilize IP and Ethernet as the system network in the vehicle [1]. OEMs and suppliers will use the new BroadR-Reach physical layer to save on weight and costs compared to currently used LVDS technology [1], [4], [5]. BroadR-Reach will be licensed by other producers [6].

An example of a camera system network is shown in **Figure 1** together with potential measurement points. As an alternative, it would also be possible to connect all cameras directly via a switch. As in bus systems used so far in the motor vehicle, the data traffic must be observed, analyzed and compared time-synchronously at various points in the network. Therefore, the measurement hardware must initially support the current bus physics (e.g. BroadR-Reach), but must also be open to future physical layers. Desirable are multi-channel taps via tee-couplers, which disturb the system network as little as possible in monitoring. The tee-coupler should also be capable of injecting errors to validate system functionality. Beyond analysis tasks, stimulation or even simulation of entire sections of the network is also required (remaining bus simulation). This poses certain challenges on the measurement hardware.

In the camera application, there are heightened requirements related to time synchronization and Quality of Service (QoS) [4]. They should be addressed by protocol extensions of the Audio Video Bridging standard (AVB) [7]. Now that manufacturers have appeared to reach agreement on the bit transmission layer (OSI Layer 1), standardization is being sought in higher layers as well for cost and testing reasons.

If only because of the different protocols used in the camera application, there are new requirements for the measurement software, so that any desired signals from the payload of the various, some quite complex, protocols can be

Figure 1: Reliable analysis of camera-based driver assistance systems requires monitoring the data traffic at multiple points of the Ethernet network, ideally via "tee-couplers" with as little time offset as possible and with a common time base.

Table 1: IP protocols of automotive applications mapped to the OSI reference model (left-side columns) including administrative functions (right-side columns): Both new protocols (red) and those known from office communications (gray) are used.

presented and manipulated according to the application. The "Audio/Video" and "Control Communication" columns of **Table 1** (based on [7] and Vector) show the protocols used for AVB. There are also protocols for bandwidth reservation and other network management protocols **(Table 1, four columns on the right)**. These and other protocols listed in the table were added based on the application cases considered below.

2. Diagnostic Access

Using "Diagnostics over IP" (DoIP) technology, it is possible to centrally flash all ECUs connected to the various bus systems via high-performance Ethernet access **(Figure 2)**. System development at the OEM must validate this service. Since an ECU is used as the gateway, not only is there great interest in analyzing the transmission of diagnostic data in the various connected bus systems, but on the IP side as well. Relevant protocols are ISO 13400 and IPv4, and possibly IPv6 as represented in **Table 1**.

Figure 2: In validation of DoIP at a gateway, it is important to represent the data traffic both on the DoIP side (to left of the gateway) and on all connected bus systems (to right of the gateway). Ideally, all messages of all networks are transmitted with a common time base.

3. Electric Refueling Station – Smart Charging

Smart Charging goes far beyond simply plugging into a household electrical outlet. The electric vehicle to be charged is connected to the electrical grid via a charging station. Charging processes do not simply start up; first, the need to charge is communicated. Delaying individual charging processes by fractions of a minute can avoid overloads of the grid. The connected vehicles can also be used as storage media, and electrical provider billing can be automated.

This is made possible by communication between the vehicle and charging station over Ethernet on IP based protocols, in standardization defined in the ISO 15118 standard. The charging station communicates with the grid and the vehicle here. For the systems manager at the automotive OEM, communication between the car and the charging station is quite important. A detailed analysis and validation of the protocols is absolutely essential to safeguard the charging process. The development tool must also support these protocols **(Table 1, "Smart Grid" column)**.

4. Calibration, Debugging, Flashing

For many years now, Ethernet has been used with the XCP measurement and calibration protocol to calibrate, debug and flash ECUs in development. However, Ethernet access is no longer provided in the production vehicle for cost reasons. Therefore, calibration and reprogramming are currently performed using the existing working protocol (e.g. CCP or XCP on CAN). However, if Ethernet makes its way into the vehicle in the near future, measurement and calibration over XCP on Ethernet would also be very attractive in production vehicles due to its significantly higher measurement data rates.

5. WLAN and Car2x

Car2x is understood as the external communication between vehicles and the infrastructure. Applications range from convenience functions to traffic flow optimization and heightened traffic safety (driver assistance systems). The technology is already in pre-production development, and standardization is quite advanced. It is IP-based, and the IEEE 802.11p standard is used as the physical layer.

From the perspective of the systems manager measurement technology interest in Car2x applications extends to beyond the boundary of the individual vehicle to a number of other vehicles and RSUs (Roadside Units) in the near environment. The ECU to be evaluated not only communicates with bus systems located in the vehicle, but also over the air interface with other traffic participants. The development tool must therefore support these IP-based standards as well. In addition, other requirements arise in the high-frequency range (WLAN in the 5 GHz band).

New Variety of Protocols for Applications and Measurement Tool

Table 1 summarizes, by examples, the various application-dependent transmission layers and protocols, which the development tool must support simply based on cases occurring so far. Some of the protocols used in the area of

Figure 3: CANoe .Ethernet supports the development, simulation and testing of embedded systems that communicate over IP or Ethernet.

office communications are found here, while many others may be omitted, and certain others are added. The table shows in light gray those protocols that can be adopted from office communications. Those added due to the new automotive application are shown in red.
The measurement system has the task of resolving all relevant protocols and placing all network events in a causal relationship (correct sequence). Here it is desirable to be able to represent all bus domains with a common time base and with sufficient precision.

Validation of IP Production Projects

As the evaluation of the above application cases demonstrates, causality or even time analysis extending over multiple bus systems make it difficult to impossible to utilize standard Ethernet tools from office communications for multi-bus applications in the vehicle. Ethernet in the office field is not the same as Ethernet in the automotive field. The same applies to the specific Internet protocols that are used. They differ in type and complexity, depending on the application – as significantly as the requirements of the physical layer differ.
A suitable engineering format is important in representing the signal structure of the protocols in the development tool and in generating the embedded code. DBC format is the commonly used engineering format for CAN, while FIBEX is commonly used for FlexRay. However, the DBC format is no longer adequate as a database format for the new Ethernet and IP based system network. From the perspective of tool suppliers, it would be helpful if OEMs could agree on a common engineering format. Suitable candidates would be FIBEX 4.0 and AUTOSAR System Description formats. Experience from other industrial fields indicates that tool producers would provide suitable development tools for analysis and code generation soon thereafter.

Outlook for Vehicle Networks

In-vehicle use of CAN is expected to continue much longer than ten years into the future, while all of the other bus systems discussed here will be used for at least ten years. Nonetheless, applications will increasingly tend towards the use of IP and Ethernet due to growing requirements with regard to bandwidth, flexibility and cost-effectiveness. In upcoming years, multiple bus systems networked over gateways will be found just as they now exist. Ethernet and IP will simply be added. As in the case of the camera application, new challenges will arise on all protocol levels in future IP applications, yet it will be possible to overcome them with suitable development tools.

Outlook for IP Development Tools

In the automotive field, development tools conceptualized for IP continue to be advisable. On the one hand, they must support all protocol levels, but on the other they must also fit into the typical industry tool landscape. Suppliers are especially called upon to provide suitable development tools for validation of product development projects at the OEM. Naturally, this includes support and ideally tool producer assistance product introduction as well.
Today, Option Ethernet, which is based on the proven CANoe simulation and test tool from Vector Informatik, already covers the described requirements for an Ethernet development tool. With its wide variety of Ethernet-specific functions and multibus capability, CANoe .Ethernet can help to reduce development time, allowing valuable resources to be used more effectively on the application side **(Figure 3)**. CANoe .Ethernet for automotive network development offers the same development convenience as is already the standard for the established CAN, LIN, MOST and FlexRay bus systems. The development tool exhibits a high degree of scalability and basically offers three interface options **(Figure 4)**. In the simplest Case 1, it works with any network cards existing on a Windows computer. If BroadR-Reach is used, or if it should also be possible to inject errors, then in the future a device of the new VN56xx product line could be used as a hardware interface (Case 2). This significantly improves time synchronism between the IP channels and with other bus systems. If real-time behavior is required, CANoe .Ethernet could be operated together with the real-time hardware VN8900 in the future, which of course works seamlessly with the VN56xx interface hardware (Case 3).

Figure 4: CANoe .Ethernet with scalable hardware interfaces and optional real-time support

Translation of a German publication in Elektronik automotive, 4/2012

Literature:

[1] Bogenberger, R., BMW AG: IP & Ethernet as potential mainstream automotive technologies. Product Day Hanser Automotive. Fellbach, 2011.
[2] Neff, A., Matheeus, K, et al.: Ethernet & IP as application vehicle bus in use scenario of camera-based driver assistance systems [German lecture]. VDI Reports 2132, Electronics in the motor vehicle. Baden-Baden, 2011. pp. 491-495.
[3] Streichert, T., Daimler AG: Short and Longterm Perspective of Ethernet for Vehicle-internal Communications. 1st Ethernet & IP @ Automotive Technology Day, BMW, Munich, 2011.
[4] Nöbauer, J., Continental AG: Migration from MOST and FlexRay Based Networks to Ethernet by Maintaining QoS. 1st Ethernet & IP @ Automotive Technology Day, BMW, Munich, 2011.
[5] Powell, S. R., Broadcom Corporation: Ethernet Physical Layer Alternatives for Automotive Applications. 1st Ethernet & IP @ Automotive Technology Day, BMW, Munich, 2011.
[6] NXP Develops Automotive Ethernet Transceivers for In-Vehicle Networks November 09, 2011, www.nxp.com/news/press-releases/2011/11/nxp-develops-automotive-ethernet-transceivers-for-in-vehicle-networks.html.
[7] Völker, L., BMW AG: One for all, Interoperability from AUTOSAR to Genivi. 1st Ethernet & IP @ Automotive Technology Day, BMW, Munich, 2011.

Image rights:
Vector Informatik GmbH

Links:
Vector Solutions for IP and Ethernet:
www.vector.com/vi_automotive_ethernet_solutions_en.html

Product information CANoe .Ethernet:
www.vector.com/vi_canoe_ethernet_en.html

Vector´s know-how especially for Smart Charging:
www.vector.com/vi_electric_vehicles_en.html

AFDX® is an Airbus' registered trademark

Hans-Werner Schaal (Graduate Engineer)
is Business Development Manager in the areas of Automotive Ethernet, Car2x and open CAN protocols such as J1939 and ISO11783 at Vector Informatik GmbH.

Challenge of Ethernet Use in the Automobile

Flexible Interfaces and Software Tools Simplify ECU Development

Already this year, Ethernet will be used as a system network in the first production vehicles. Therefore, the next step is to integrate Ethernet with established technologies in the automotive industry: CAN, FlexRay, LIN and MOST. Functional development tools exist for them, which make it easy for developers to analyze heterogeneous networks. On the Ethernet side, however, only standard tools from office communications exist, but they do not support the special physical layers and IP protocols of the automotive world. Therefore, development and test tools are urgently needed that can be used to analyze and test existing bus systems together with Ethernet networks. But what would be the exact requirements of these tools?

It is already state-of-the-art technology to transmit camera images in vehicles at 100 MBit/s over a cost-effective, unshielded twisted pair connection. This technology is known as BroadR-Reach, which is standardized by the OPEN Alliance SIG consortium [1]. The next objective is to use Ethernet as a network for infotainment and driver assistance systems by 2015. Some OEMs predict that Ethernet will become a backbone technology starting as early as 2018 [2]. As described in a number of professional articles [3, 4], Ethernet offers flexibility, scalability and cost advantages in automotive use, especially in combination with certain Internet Protocols **(Figure 1, [1])**. Moreover, it offers the opportunity to enrich the proven automotive development process with methods from the IT world.

Challenges of an Automotive Ethernet Test Solution

The use of Ethernet in motor vehicles will require rethinking by developers and test engineers. First, efforts must address the issue of how to obtain a clear domain architecture **(Figure 2)**. In this architecture, the backbone is no longer a bus system, but rather it is implemented as a switched network with multiple full-duplex connections. When using it to implement real-time critical applications, synchronization technologies are required on higher protocol layers above the physical layer (OSI layer 1), e.g. AVB (Audio Video Bridging, **Figure 1**). Analysis requirements are also growing for the new architecture. For example, if the developer wishes to simultaneously analyze all data traffic on the backbone, access must be synchronized on all path branches **(Figure 2, a, b, c, d)**.

Figure 1: Along with protocols familiar from the field of office communications such as UDP, TCP and IP, protocols specially optimized for automotive use are also used. They are described in ISO CD 17215-1.

Second, developers must utilize new, relevant filtering strategies to process the enormous quantities of data. The situation will be further intensified by transmission rates in the gigabit per second range, which are already on the wish lists of OEMs. A physical layer that is suitable for this, known as RTPGE (Reduced Twisted Pair Gigabit Ethernet), is already in development.

Minimizing Effects of the Interface on System Performance

Unlike in a bus system, special measures must be taken to avoid measurement effects on the system. On the one hand, developers must consider testability early in the system design (Design-to-Test). On the other hand, the tool producer must minimize the effects of the interface. Presented in the following are various measurement setups for analysis and testing purposes; their undesirable effects are explained, and it is shown how these effects can be minimized.

Limitations of Previous Solutions

A natural approach to analyzing an Ethernet network is to use an additional port (monitor port) at the implemented switches in the system (mirroring). All packets received from the switch are forwarded at this monitor port. This provides access to the arriving data packets, yet these data packets are not interrelated by a common time reference – there is no time stamp. Moreover, often only valid packets are forwarded to the monitor port, which makes error analysis difficult. Furthermore, for cost reasons no additional monitor port is provided at the switch for mirroring in the production system. [4].

Figure 2: Potential domain architecture of future IP networks in the motor vehicle. To be able to analyze all Ethernet packets, the analysis software must access all Ethernet paths synchronously.

If no additional port is available at a given switch, an additional switch can be inserted in an existing connection. This additional hop is not transparent, however, and it causes a delay over the total transmission path. In networks that are synchronized by the AVB protocol this dynamic delay may disturb the time synchronization under certain circumstances. For this measurement setup, it is possible to utilize tools and switches commonly used in the IT field. However, in the BroadR-Reach networks that are generally used in the automotive industry it is necessary to perform a media conversion to standard Ethernet (IEEE 802.3). Moreover, from the perspective of automotive network development these tools are usually insular solutions, and they have no reference to bus systems that are still important and commonly used.

Transparent Ethernet Analysis

Instead of using a classic switch as an interface, it is desirable to monitor the network by a method that is as transparent as possible. The primary objective here is to avoid having the system affected by increased latency time or filtering of faulty packets. This can be done by a so-called TAP (Test Access Point) **(Figure 3)**, which acquires and routes the data passively on the physical layer **(path 1 in Figure 4)**. In this process, the latency time is not only very short, but constant as well, which is especially advantageous in the analysis of AVB systems. Another method of transparent monitoring involves using a switch with AVB time synchronization support. The AVB protocol then compensates for the latency time that occurs when the packets are routed.

Regardless of which method is chosen, to accurately analyze the packet data precise time stamps are needed which are acquired as close to the physical layer as possible. These time stamps must be synchronized with other interfaces, because the network analysis often focuses on more than just one Ethernet path **(Figure 2)**.

For an inactive interface a transparent behavior is also important. If the interface hardware is installed in the vehicle for a test drive, for example, the interface must autonomously assume a preconfigured stand-alone mode even if the measurement application is inactive. Otherwise, certain Ethernet paths would be interrupted during the drive.

TAP with Stimulation

Along with pure data analysis, the network must often be tested by intentionally sending certain packets. Here – as in pure monitoring – the connections between any two nodes should be affected as little as possible. However, these supplemental packets cannot be sent on the physical layer, because an additional flow control is necessary, which is not available until layer 2. Here too, dynamic latency times occur, which could be compensated by protocol support directly at the interface, e.g. by AVB.

One use is to send supplemental faulty data for test purposes while the normal communication between the two nodes is running **(path 3 in Figure 4)**. This data is either supplied directly by a test application, e.g. Vector CANoe .Ethernet, or by a packet generator which generates a defined bus load directly at the interface **(path 2 in Figure 4)**.

Remaining Bus Simulation

Particularly when developing individual ECUs, the remaining bus simulation [6] is a flexible way to test any types of scenarios before the ECUs are integrated in a real network. First, hardware is needed that permits unrestricted and high-performance network access. Second, the application must be able to forward logged or self-generated data to the hardware **(path 4 in Figure 4)**. And third, the combination of hardware and software must be able to receive packets, corrupt them and then send the corrupted packets. This provides a way to test ECU behavior in response to specific error cases such as protocol errors.

Figure 3: Possible wiring of Ethernet interfaces for analysis or for a remaining bus simulation. Synchronism with familiar automotive bus systems is also required.

Important Properties of a Flexible Interface/Software Combination

The described measurement setups illustrate how the analysis of Ethernet networks places different requirements on the hardware and software. To avoid having to change interfaces for the different measurement setups, it must be possible to use the interface flexibly as a TAP, converter or as a switch with supplemental functionality. The following properties are desirable here:

> In the simplest case, when the interface is used as a TAP, the TAP itself must only cause minimal and precisely specifiable latency times.
> The interface must be able to convert between all commonly used media such as BroadR-Reach, Fast Ethernet, Gigabit Ethernet and in the future RTPGE as well. This eliminates the need to use external media converters, which has been necessary so far.
> For test drives, it must be possible to install the interface in the vehicle, and while it is not being used it must not disturb the network (stand-alone mode).
> Packet generators are important on the software or hardware level, because along with analysis the automotive development process also requires controlled stimulation.
> In conjunction with the simulation software the hardware interface must allow real media access for one or even several virtual network nodes (remaining bus simulation).
> It must be possible to use the analysis and simulation tool to analyze and manipulate data on all OSI layers of interest and over all protocol levels.
> To support heterogeneous networks, it must be possible to synchronize the interface with all commonly used bus systems.

Figure 4: The VN5610 Ethernet/CAN interface passively and actively participates in communication in Ethernet networks together with CANoe .Ethernet/CANalyzer .Ethernet. A flexible configuration supports different measurement setups for analysis and test purposes.

The use of high-performance analysis tools from the field of office communications together with external media converters is often overly simplified. The noted requirements can only be implemented by specialized hardware that is closely intermeshed with the analysis and simulation software. One combination already used in practice is the VN5610 Ethernet/CAN interface from Vector together with the CANoe .Ethernet development tool. This solution is already being used by some automotive OEMs and suppliers.

Outlook

Over the next five to ten years, heterogeneous network structures will continue to be found as clusters of established bus systems in the vehicle. After the camera application, Ethernet will be used in other system domains and will to some extent replace other bus systems. After being used as a backbone, Ethernet and IP technologies will penetrate into other automotive application areas.

For developers of vehicle networks, multibus capability, remaining bus simulation and low-level time stamps for all data packets will continue to gain in importance. At Vector, the next development steps in the Ethernet and IP area will be to support users in signal representation over all IP protocol layers **(Figure 1)** and to comprehensively check real-time and service-oriented communication, e.g. via AVB and SOME/IP.

Translation of a German publication in Hanser automotive, April/2013

Image rights:
Vector Informatik GmbH

Literature References:

[1] OPEN Alliance SIG, http://www.opensig.org/
[2] Das IP-basierte Bordnetz kommt ("The IP-based on-board electrical system is coming"], Elmar Frickenstein, BMW AG, SEIS Status seminar, 2011-20-09, http://www.strategiekreis-elektromobilitaet.de
[3] Ethernet und IP im Kraftfahrzeug: Neue Anforderungen an das Entwicklungswerkzeug durch den Ethernet- und IP-Einsatz ("Ethernet and IP in motor vehicles: New development tool requirements for the use of Ethernet and IP"], Hans-Werner.Schaal, Elektronik automotive, April 2012
[4] Herausforderungen von Ethernet-Debugging ["Challenges of Ethernet debugging"], Helge Zinner, www.elektroniknet.de, October 2012

[5] ISO CD 17215-1 (E) of 2012-07-02
[6] Schnelle Wege zur Restbussimulation: Virtuelle Netzwerke ohne Programmier-Know-how erstellen ["Quick paths to remaining bus simulation: Creating virtual networks without requiring programming know-how"], Stefan Albrecht and Peter Decker, Automobil Elektronik, March 2012

Links:
Vector: www.vector.com

Vector Solutions for IP and Ethernet:
www.vector.com/vi_automotive_ethernet_solutions_en.html

Hans-Werner Schaal (Dipl.-Ing.)
is Business Development Manager for the area of IP, Car2x and open CAN protocols such as J1939 and ISO11783 at Vector.

Matthias Schwedt (Dipl.-Ing. (FH))
is FPGA developer for Ethernet, FlexRay and MOST150 network interfaces as well as overall project leader for the VN56xx Ethernet network interface family at Vector.

New Perspectives on Remaining Bus Simulation for Networks with SOME/IP

Validating Automotive IP Network Elements

Ethernet based on BroadR-Reach is already a reality in vehicles with camera applications, and it will reach other application areas as well. Specialized development tools even permit time-synchronous display of the communication of heterogeneous networks. To enable bandwidth efficiency, automotive IP networks – in contrast to static CAN communication – are set up in a dynamic and service-oriented way. Therefore, the development tools must also support service-oriented protocols such as SOME/IP.

Taking the example of SOME/IP (Scalable Service-Oriented Middleware on Ethernet / IP), this article shows how to implement a remaining bus simulation [1] for dynamic, service-oriented IP networks (**Figure 1**). Afterwards, the aspects of media access, synchronization and controlled stimulation [2] are discussed, and recommendations for the development system are derived from them.

Use of Service Protocols Based on the Example of SOME/IP

In the Ethernet and IP field, there are a tremendous number of protocols to choose from, and one might think that protocols already exist that could be used directly in all conceivable types of applications. However, since networking in the vehicle does not start at zero, certain specific requirements become immediately apparent. For example, the protocols used must be incorporated into existing systems. In particular, this relates to good integration in AUTOSAR and fast reaction times to keep delays short in case of error. BMW AG has developed and specified SOME/IP, which is an open protocol that fulfills automotive requirements. A SOME/IP implementation is available from Vector for use in ECUs, including a TCP/IP stack, Service Discovery and BroadR-Reach Ethernet driver.

SOME/IP offers interfaces for service-oriented communication (**Figure 2**). This distinguishes it from the pure signal-based communication that is usual in CAN, for example. SOME/IP is roughly subdivided into three areas: Service Discovery (SD), Remote Procedure Call (RPC) and access to process data. The SD area lets ECUs find services or offer their services in the network. The user utilizes methods offered by SD via RPC (**Figure 3, Part A**). In addition, it is possible to set up notifications for specific events (**Figure 3, Part B**). The application can use read and write functions to access any specific process data (**Figure 3, Part C**). The goal of SOME/IP is to optimally utilize available bandwidth to implement very flexible communication. The communication and the signals are described using FIBEX 4.1 or higher or AUTOSAR formats version 4.1 or higher.

For the remaining bus simulation, SOME/IP serves as a complex protocol and middleware with all of its degrees of freedom. To keep user complexity as low as possible, the remaining bus simulation is largely configured autonomously. This involves the use of standardized description formats such as FIBEX 4.1 or AUTOSAR. This gives users direct access to the signals. However, manual overwriting of the configuration is desirable in executing tests, e.g. to provoke error cases.

Figure 1: Excerpt from a heterogeneous network architecture with Ethernet networks.

Flexible Network Access for the Test Tool

Implementing the complex task of a test tool – such as a remaining bus simulation – as optimally as possible requires that the application has flexible and high-performance access on the physical medium (**Figure 4, remaining bus simulation**). Using this access, the developer must be able to generate error cases (e.g. CRC errors) on the network. If the primary focus is on analyzing a connection between two nodes, special measures must be taken to enable transparent, interference-free access. This is necessary, because although Open Alliance BroadR-Reach (OABR) is logically a bus system, electrically it is a point-to-point connection.

An obvious solution is to use an additional monitor port on the switches used in the system. However, for cost reasons this cannot always be done in the production version system [3]. Using what is known as mirroring, the switch forwards all received packets to the monitor port. One disadvantage of this solution is that the received data packets do not have any common time reference – there are no time stamps. For another, it is often the case that only valid packets are forwarded to the monitor port, which makes error analysis difficult.

Media access with little effect on the network under analysis is provided by a so-called TAP (Test Access Point) [2]. Unlike a standard switch, this TAP makes it possible to transparently analyze an Ethernet network without affecting the communication between two nodes (**Figure 4, flexible TAP**). The TAP may be operated in two different modes, depending on the following requirements:

> The purely passive operating mode guarantees short and constant latency time, but in this mode it is only possible to listen on the medium.

Figure 2: SOME/IP offers an interface for calibration.

Figure 3: SOME/IP offers various types of access such as RPC, notifications and read and write functions.

> In the active mode, additional data may be injected into an existing connection.

The active connection cannot be made on the Physical Layer (OSI Layer 1), however, because additional flow control is necessary. Flow control is not available until Layer 2. Nonetheless, flow control cannot guarantee constant latency time. Regardless of the method chosen, precise time stamps obtained as close as possible to the Physical Layer are necessary for precise analysis of the packet data. These time stamps must be synchronized with other interfaces, because network analysis often focuses on more than just one Ethernet path as well as on other automobile networks.

Choosing the Development Tool

Based on the above considerations, every developer should ask five questions before choosing a development tool:

> Does the development system support service-oriented communication like SOME/IP?
> Does the development system provide logging and controlled stimulation with and without protocol violation?
> Does the development system offer access to typical automotive networks such as OABR, CAN, FlexRay and MOST?
> Can the interface be used flexibly as a TAP for mirroring and as a converter and switch [2]?
> Can the interface for supporting heterogeneous networks be synchronized with all commonly used bus systems and IP networks?

The development tool CANoe .Ethernet supports all of these functions with the VN5610 Ethernet/CAN interface from Vector. Therefore, it is already being used at some automotive OEMs and suppliers.

Outlook

IP-based, service-oriented communication is making great strides forward. After being used in camera applications, Ethernet will be used in the infotainment area and then in other system domains, e.g. as a backbone. For developers of vehicle networks, the significance of multi-bus capability, remaining bus simulation and low-level time stamps for all data packets continues to grow.

Translation of a German publication in Automobil Elektronik, 4/2013

Image rights:
Vector Informatik GmbH

References:

[1] Schnelle Wege zur Restbussimulation: Virtuelle Netzwerke ohne Programmier-Know-how erstellen ("Quick paths to the remaining bus simulation: Creating virtual networks without programming know-how"), Stefan Albrecht and Peter Decker, Automobil Elektronik 03/2012

[2] Neue Werkzeuge für Automotive Ethernet ("New tools for automotive Ethernet"), Hans-Werner Schaal, Matthias Schwedt, Hanser automotive 3-4/2013

[3] Herausforderungen von Ethernet-Debugging ("Challenges of Ethernet debugging"), Helge Zinner, www.elektroniknet.de, October 2012

Figure 4: The VN5610 interface is used in various application cases.

Hans-Werner Schaal (Graduate Engineer)
is Business Development Manager in the areas of Automotive Ethernet, Car2x and open CAN protocols such as J1939 and ISO11783 at Vector Informatik GmbH.

Matthias Schwedt (Dipl.-Ing. (FH))
is FPGA developer for Ethernet, FlexRay and MOST150 network interfaces as well as overall Project Leader for the VN56xx Ethernet network interface product line at Vector Informatik GmbH.

Peter Fellmeth (Dipl.-Ing. (FH))
is Group Leader and Product Manager at Vector Informatik GmbH. He is responsible for developing products and customer-specific projects in the Ethernet, J1939 and ISOBUS fields.

AUTOSAR Learns Ethernet

Ethernet is a new and yet old and familiar network technology that is making its way into vehicles. At first, it is just being used for diagnostic applications and intelligent charging of electric vehicles, but onboard Ethernet networks are now being implemented as well. This article describes the properties and advantages of Ethernet and discusses the special aspects of integrating the technology in AUTOSAR. Finally, useful extensions are presented for an AUTOSAR Ethernet Stack which can be used to implement new applications.

Until just a few years ago, CAN and LIN were the only bus systems being used in vehicles. The desire for more bandwidth and growing requirements in the safety field, especially with regard to X-by-wire systems, led to the development and introduction of FlexRay. The MOST standard also became established for high-end applications in the multimedia field. Unlike CAN, FlexRay and MOST are complex and expensive bus systems. Because of this, and due to the lack of a service-garage network for these bus systems, CAN was still used for external access in vehicle diagnostics. However, the time required to program ECUs increased dramatically due to the limited bandwidth of CAN and the increasing amount of software content. The Diagnostics over Internet Protocol (DoIP) was developed several years ago to resolve this problem. This protocol is the first to be based on Ethernet as the network technology in the vehicle environment, and it is standardized in ISO 13400. Ethernet offers the advantage of high bandwidth, and it has primarily taken hold in the office and Internet worlds. It makes it easy to integrate a DoIP-based diagnostic tester in an existing service-garage network. DoIP laid the foundation for the use of Ethernet in vehicles. When electric mobility became a central topic a short time later, the focus shifted towards Ethernet-based Vehicle-to-Grid applications. In the charging process, the electric or hybrid vehicle communicates with an energy provider's charging spot. The communication is based on TCP/IPv6 and a dedicated Smart Charge Communication (SCC) protocol, in order to exchange such information as the charging type (AC/DC), date and time of charging, duration of charging and rate and payment information.

The standard shielded Ethernet cable with its high wiring costs prevented widespread use of the technology for in-vehicle networks. However, introduction of the new BroadR-Reach physical layer has made the option of Ethernet interesting for in-vehicle communications as well. Using twisted pair lines, BroadR-Reach offers a bandwidth of 100 MBit/s which represents a 100-fold increase in speed compared to CAN without increased costs for wiring. It also offers the benefits of a switched network, which enables implementation of a backbone architecture, for example **(Figure 1)**. Other applications that are currently of interest to automotive OEMs and their suppliers include Audio Video Bridging (AVB), network management and new gateway ECU concepts.

Ethernet in combination with the Internet Protocol (IP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) also enable the transition from a data-oriented to a service-oriented communication schema. BMW has developed a serialization protocol, the Service Oriented Middleware over Internet Protocol (SOME/IP), which among other things can be used for remote procedure calls. It is complemented by the Service Discovery (SD) protocol that was also specified. ECUs use Service Discovery to inform communication partners of the availability of their services. ECUs can also use it to search for services and register their events.

Ethernet and AUTOSAR

Ethernet has been part of the AUTOSAR standard since Version 4.0. In the AUTOSAR architecture, the Ethernet communications stack is laid out in parallel to the CAN, LIN and FlexRay stacks. However, unlike them, it exhibits a number of special aspects – which relate to the higher protocol layers IP, UDP and TCP in particular. The Ethernet Transceiver Driver (EthTrcv) and Ethernet Driver (Eth) modules are comparable to those of other network technologies. The Ethernet Interface (EthIf) module, on the other hand, is different. While the interfaces for CAN, LIN and FlexRay implement the AUTOSAR Protocol Data Unit (PDU) interface directly, the Ethernet Interface routes raw data to the TCP/IP stack or receives data from it. The IP, UDP and TCP protocols are processed in the TCP/IP stack, which however is not fully specified in AUTOSAR 4.0. The use of a common off-the-shelf TCP/IP stack is recommended here.

A paradigm upon which the TCP/IP protocol family is based is the use of sockets. A socket is uniquely identified by the combination of IP address and port of the remote and local end nodes. Via a socket, packet-oriented UDP and connection-oriented TCP user data are routed from the TCP/IP stack to the application or in the opposite direction. This paradigm is incompatible with the PDU concept of AUTOSAR. Transformation of socket-based communication into PDU-based communication and in reverse is the task of the Socket Adaptor Module (SoAd). It provides the familiar PDU interface to higher level modules, which fully integrates the Ethernet stack in the AUTOSAR architecture.

The Ethernet stack specified in AUTOSAR 4.0 has established a foundation for receiving and sending PDUs over Ethernet. It also considers the use case DoIP. The implementation of the DoIP protocol shall be realized as Socket Adaptor plug-in. Moreover, this AUTOSAR version supports ECU calibration via XCP over Ethernet and network management over UDP, and it offers an interface for connecting AUTOSAR Complex Drivers (Cdd). Automated data parameterization of the Ethernet stack is only partially covered in AUTOSAR 4.0. The user can represent Ethernet networks, frames and PDUs in the AUTOSAR System Description or in the ECU Extract of System Description.

Figure 1: Potential domain architecture for future motor vehicles with Ethernet networks.

Figure 2: An Ethernet stack is configured with an AUTOSAR 4.1 description file.

Pre-filling of data for higher protocol layers, e.g. definitions of IP addresses and ports, is not specified.

Extended Ethernet Support in AUTOSAR 4.1

With the introduction of in-vehicle Ethernet networks, new requirements have evolved, which an AUTOSAR 4.0 Ethernet stack does not fulfill. It is very difficult to implement efficient transmission of multiple PDUs, for example. Therefore, the Ethernet stack was revised significantly in AUTOSAR 4.1.1:

- The TCP/IP stack is now an AUTOSAR module.
- Besides version 4 of the Internet Protocol, version 6 is also supported. The two IP versions can be operated either individually or in parallel in one ECU.
- It is now possible to use Virtual Local Area Networks (VLANs).
- PDU-based data transmission over the Socket Adaptor is much more efficient.
- In its new version, the Socket Adaptor offers a generic interface to higher level modules.
- Implementation of the DoIP protocol was removed from the Socket Adaptor and relocated to a separate DoIP module.
- The Service Discovery protocol is also specified as a new AUTOSAR module.

The SOME/IP protocol and the use cases SCC and AVB are still not covered in AUTOSAR. The description of a sample implementation of SOME/IP is available as a supplemental document to the current standard.
In practice, only FIBEX 4.1 has been used so far as the description format for in-vehicle Ethernet networks. It has now been harmonized with AUTOSAR 4.1.1. This means that although the two description formats are not identical, their contents can be transformed from one format to the other without loss of information. To a large extent, this enables automated data parameterization of the Ethernet stack per AUTOSAR 4.1.1 **(Figure 2)**.

Figure 3: The AUTOSAR Ethernet stack MICROSAR.ETH from Vector contains AUTOSAR modules and useful supplements.

Useful Supplements from Practice

As already mentioned, some applications of the Ethernet stack, such as Smart Charge Communication, are not covered by AUTOSAR specifications. For this purpose, there are ISO and DIN standards, which Vector helped to create. Producers and suppliers of electric and hybrid vehicles need the protocols specified in these standards for intelligent charging. Ideally, the protocols would be seamlessly integrated in an AUTOSAR Ethernet stack.
Per specification, the Universal Measurement and Calibration Protocol (XCP) does not have routing capability. When Ethernet is used for vehicle access, it is also nessesary to calibrate all ECUs that are not directly connected to the Ethernet network over XCP. Vector developed a mechanism that enables this in cooperation with a German automotive OEM. Routing of DoIP over a gateway to a sub-Ethernet network is not standardized in the ISO 13400 specification. Nonetheless, solution approaches have already been worked out with various automotive OEMs.
The Ethernet stack defined in AUTOSAR is available from Vector as ECU software under the product name MICROSAR.ETH **(Figure 3)**. It contains the functionality specified in AUTOSAR 4.0.3 and 4.1.1 and is also available for AUTOSAR 3.x projects. The extensions mentioned above are included, as well as a resource-minimizing implementation of SOME/IP. The architecture of MICROSAR.ETH permits implementation of customer-specific extensions without any problems.

Outlook

One feature of AVB is that it enables time-synchronous transmission of audio and video streams over Ethernet. The IEEE 1722 transport protocol that is needed for this is already available from Vector. AVB support is currently being extended, e.g. by integrating time synchronization with the Generalized Precision Time Protocol.
In AUTOSAR version 4.2.1, there will presumably be some extensions related to the Ethernet stack. Current efforts include adopting data serialization via SOME/IP into the standard. These plans also include supporting data serialization for sender-receiver communication. Currently, this is only possible for client-server connections. Another document describes the introduction of a second communication module, which is specially designed for efficient sending and receiving of serialized data. Other concepts currently under discussion relate to the allocation of IP addresses and global time synchronization across different networks.

Translation of a German publication in Hanser automotive, October 2013

Image rights:
Vector Informatik GmbH

Links:
Vector: www.vector.com
MICROSAR basic software: www.vector.com/microsar

Marc Weber (M.Eng.)
is responsible for product management of the Ethernet stack in the Embedded Software product line.
marc.weber@vector.com

New Communication Paradigms in Automotive Networking

Ethernet and CAN FD are the New Trailblazers

CAN is the prevailing bus system for communication between electronic control units (ECUs) in today's vehicles. The communication demand has increased dramatically in recent years, and vehicle manufacturers are now reaching the limits of vehicle networking with CAN. Ethernet and CAN FD provide higher bandwidth and are taking over some of the tasks of existing bus systems. The higher bandwidth is one aspect, but also of high relevance is the introduction of new communication paradigms.

A classic CAN message transmits a payload of up to eight data bytes. For reasons of efficiency, it is advantageous to use all eight bytes to maintain the best possible ratio between payload data and protocol overhead. However, the individual data elements (signals) being transmitted, such as wheel speed, are often less than eight bytes in length. Several signals are therefore sent together. Each bit is valuable, and thus the task of defining each CAN message with the signals contained therein becomes very complex.
The communication matrix is defined in a database. For CAN this is done in one of the following formats: DBC, FIBEX, or AUTOSAR System Description. The database **(Figure 1)** contains not only the messages and their composition but also the associated send and receive relationships. Also defined in the database are the Protocol Data Units (PDU) – an abstraction layer between signals and messages.

CAN FD, the Improved CAN Bus

The maximum transmission rate of 1 Mbps on the CAN bus is no longer sufficient for all of today's communication demands. A solution to the bandwidth problem is to use CAN with flexible data rate (CAN FD). This enhanced version of the CAN protocol features transmission rates of up to 8 Mbps. This is achieved by two enhancements to classic CAN:

> 1. The payload data bytes are transmitted at a higher bit rate. In order for the other properties of CAN, such as the maximum cable length, to remain the same as possible, the header and trailer of a CAN message are sent at the normal bit rate.
> 2. A CAN FD message contains up to 64 payload data bytes. If an eight times faster bit rate is used and 64 payload data bytes are sent, the transmission time is comparable to a classic CAN message with 8 payload data bytes.

Figure 1: A database contains both system and data descriptions

With CAN FD, the previously mentioned complexity of defining the communication matrix is significantly increased once again. To ensure efficient communication in this case, network designers must logically group signals in messages that are up to 8 times larger. Some scenarios also require that PDUs defined previously for CAN are available on CAN FD as well. This would eliminate the advantage of the greater payload data length. Assistance is provided by the n-PDU-to-frame mapping, which allows the sending of multiple PDUs in one message.

Exemplary n-PDU-to-frame Mapping in a Gateway Scenario

In a gateway scenario **(Figure 2)**, the bandwidth of a classic CAN bus may no longer be sufficient. Let's assume that three CAN buses are connected to a gateway and all are already being operated at the bus load limit. An ECU that requires much of the data of the other two buses is attached to CAN bus 3. The gateway is responsible for routing the needed data to the ECU. If, due to a change of generation, the ECU on CAN bus 3 has a need for additional communication, the bandwidth of CAN bus 3 is no longer sufficient. CAN FD should therefore be used instead. In addition, there are further requirements:

> Use of all 64 payload data bytes for efficient bus utilization.
> The previously defined PDUs must not be changed.
> The gateway must not become significantly more complex.

All of these requirements can be met by the sending of multiple previously-defined CAN PDUs in one CAN FD message by the gateway.
Up to now, the content of a message was identified by the respective CAN identifier (CAN ID). The receiver can no longer use the CAN ID for data identification since a CAN FD message contains multiple PDUs. A possible solution would be to define the content of the CAN FD message statically in a database. In contrast, with n-PDU-to-frame mapping, the database defines the PDUs that can potentially be contained in the CAN FD message. The PDUs that the ECU actually sends are chosen during runtime. The described scenario is presented schematically in **Figure 2**. Each ECU on CAN bus 1 and CAN bus 2 as well as the gateway itself sends a PDU that is to be transmitted on the CAN FD bus. The gateway fills up the CAN FD message step-by-step with the PDUs up to the time of sending. A small header is put in front of each PDU so that a receiver can extract the

Figure 2: Possible scenario for introduction of CAN FD

individual PDUs from the message. The gateway no longer has to pay attention to the order of the PDUs and is spared the complex task of keeping the defined message layout. This keeps the demand for resources on the gateway as low as possible.

The sending of a CAN FD message is triggered by various events, such as:

> 1. The send buffer of the message is full.
> 2. The timeout defined for the complete message expires.
> 3. After expiration of the timeout defined for a single PDU, the message containing this PDU is sent.
> 4. A PDU has an additional attribute that triggers immediate sending of the message after the PDU is copied to the send buffer.

The option of sending multiple PDUs in one message has been defined in AUTOSAR since release 4.2.1. The mechanism is specified independent of the network technology in the I-PDU Multiplexer module so that it can be used, for example, for FlexRay or Ethernet in addition to CAN FD. Besides the pure collection of PDUs, the module also supports the different sending conditions and two different PDU header formats. For CAN FD and FlexRay, a 4-byte header is primarily used while an 8-byte header is generally used on Ethernet.

Ethernet Provides a Much Larger Payload Data Length

Compared to CAN, Ethernet provides a maximum payload data length that is more than 185 times larger. A classic definition of hundreds or even thousands of signals within a 1500-byte PDU is not really feasible. For gateway applications, however, it can be advantageous to forward existing CAN or FlexRay PDUs 1:1 on Ethernet. The sending of multiple PDUs within one message, which was described for CAN FD, can be used for Ethernet unchanged. In contrast to CAN FD and FlexRay, AUTOSAR specifies two equivalent approaches for Ethernet. The first approach is the n-PDU-to-frame mapping in the I-PDU Multiplexer, which has already been described for CAN FD. The same PDU collection algorithm is specified in the Socket Adaptor module, which is responsible for linking the TCP/IP stack to the rest of the AUTOSAR architecture. The user can use both approaches. When the Socket Adaptor is used, additional control possibilities for Ethernet-based communication are available.

Different Network Topology with Ethernet

Besides the payload data length, Ethernet differs significantly from other network technologies in regard to network topology and addressing **(Figure 3)**. CAN uses a classical bus topology while FlexRay can be physically implemented with a star topology, from a logical standpoint it behaves like a bus, too. In the case of both network technologies, a message is sent to all nodes. Each network node decides independently whether a message is relevant to it and will be further processed. This is done using the CAN ID for CAN and the Slot ID for FlexRay. In either case, the ID already classifies the content of the message. There is no option of selectively sending a message to only one receiver. For this reason, CAN and FlexRay are broadcast media. At the beginning of its success story, Ethernet was also a bus system. The cabling was implemented with coaxial cables and T-elements. Today, Ethernet is, in the vast majority of cases, an actively switched network with tree topology in which collisions cannot occur due to point-to-point connections.

Figure 3: Comparison between network topologies of CAN (FD), FlexRay and Ethernet

Every Ethernet node has a MAC address, which is used for unique addressing in the network. A node processes a message when its MAC address is specified as the destination. In other words, the sender must know the receiver and enter the corresponding destination address in the message. This 1:1 communication relationship uses so-called unicast addressing. If a unicast-addressed message is sent to all network nodes, only one receiver processes the packet. All other receivers discard it. To prevent flooding the network unnecessarily, the switch as an active network connection element was introduced. After a brief learning phase, a switch forwards messages only to the connection where the addressed destination can be reached. The available bandwidth is thus used efficiently. Furthermore, a switch enables multiple unicast-addressed messages to be sent simultaneously on the network. **Figure 3** shows an example Ethernet network with two switches: One in the ECU at the top of the figure, the second in the unlabeled ECU. Highlighted in color are parallel, non-interfering communication paths through the network. All connections support up to 100 Mbps full-duplex data transmission. As a result, the bandwidth of 100 Mbps is multiplied by the number of parallel communication paths. Ethernet provides 1:n (multicast) and 1:all (broadcast) communication as well. If these addressing methods are not used skillfully, the described advantages of Ethernet are lost.

In the case of unicast addressing, the intelligence migrates on the network from the receiver to the sender. The sender must know which receivers in the network are interested in which data (PDUs) and must assemble the messages appropriately using n-PDU-to-frame mapping. It is possible for PDUs or messages to be sent multiple times if there are multiple receivers for the same PDUs. This receiver-specific data fan-out can be implemented most easily in AUTOSAR with the Socket Adaptor.

Figure 4: Operating principles of service-oriented communication using Service Discovery

New Communication Paradigm: Service-oriented Communication

The properties of Ethernet as well as the desire of vehicle manufacturers for more flexibility and controllability of the networking complexity led to the introduction of service-oriented communication in the automotive environment. This familiar communication paradigm from the Internet world has been transferred to vehicle networking. However, specific protocols optimized for automotive applications are being used: Service Discovery (SD) and Scalable Service-oriented Middleware over Internet Protocol (SOME/IP). Up to now, we spoke of senders and receivers in the vehicle network. With service-oriented communication there is one ECU that provides a service (the server) and ECUs that use this service (the clients).

Servers use the Service Discovery protocol to announce which services they provide at runtime and how these services are addressed. Clients invoke methods of the server (Remote Procedure Calls) or subscribe at the server, in order to subsequently receive data updates automatically. In the latter case, the server provides certain data elements (called events) and sends their values to all subscribed clients. The application of the server triggers the sending of data updates. The n-PDU-to-frame mapping enables multiple events to be sent within one message. **Figure 4** graphically represents the two principles of service-oriented communication.

For method invocations and data updates, the length of the data to be transmitted can vary significantly. The support for this variable data length is a main feature of SOME/IP. It serializes structured and complex application data so that the remaining basic software of an ECU only has to take care of sending or receiving a byte stream. For this reason, the exact message layout is no longer defined in a database.

AUTOSAR fully supports SD and SOME/IP. Because of the platform independence of SD and SOME/IP, the protocols are also used for data exchange between AUTOSAR ECUs and other platforms.

Conclusion

With their greater payload data length, CAN FD and Ethernet provide new data transmission possibilities. They also create new challenges for vehicle manufacturers and their suppliers – for example, when creating the system description. Assistance is provided by n-PDU-to-frame mapping, which packs multiple PDUs in one CAN FD or Ethernet message. In the case of Ethernet, however, there is not only an increase in the user data length but also in available bandwidth. Furthermore the switched network and associated new addressing method represent a small revolution in the automotive environment. New mechanisms for data distribution are necessary. Building on this, data exchange is becoming more dynamic by means of service-oriented communication. AUTOSAR 4.2.1 already supports all of the presented mechanisms and thus facilitates implementation of the new communication paradigms. Basic software implementations of this AUTOSAR version are already available on the market. An example is MICROSAR from Vector along with the development tool CANoe which offers convenient analysis and testing of CAN FD and Ethernet networks.

A shortened version of this article was published in the German magazine Automobil Elektronik, issue 7/8 2015.

Image rights:
Vector Informatik GmbH

Links:
Website Vector: vector.com
Product information MICROSAR: www.vector.com/microsar
Product information CANoe: www.vector.com/canoe

M.Eng. Marc Weber
is responsible for product management of the Ethernet stack for the Embedded Software product line of Vector Informatik.

Full Transparency with Automotive Ethernet

Finally seeing what is really happening

Automotive Ethernet – in contrast to CAN, LIN and FlexRay – is not simply another bus that sparkles with very high transmission rates. Rather development and testing departments are confronted with an entirely different network topology, which in many aspects requires a different approach to thinking and acting. This article describes, based on the example of a new flexible Automotive Ethernet interface, the specific challenges that developers face. It also demonstrates the ideal way to structure hardware and software and have them interact to achieve optimal development, simulation and test results under the new constraints.

The central appeal of Ethernet in the automotive industry is that it offers tremendous gains in bandwidth, which is playing an increasingly more important role in current and future applications. Systems for autonomous driving and Advanced Driver Assistance Systems (ADAS) are always in need of most current data about the surroundings from cameras and radar systems. There, and in other areas, high transmission rates are needed, e.g. to quickly transmit video streams and sensor data. But the time and cost advantages of a high bandwidth are even welcome in such tasks as flashing large numbers of ECUs. Using the IEEE standard for Automotive Ethernet, 100BASE-T1 (OABR, Open Alliance BroadR-Reach), high-speed networks can be implemented easily and cost-effectively based on a single, unshielded twisted pair cable (UTP). The system is capable of transmitting a gross data rate of 100 MBit/s simultaneously in each direction. Based on the full-duplex method, it can therefore achieve an overall rate of 200 Mbit/s.

Network of 1-to-1 Connections and Switches Instead of Bus System

However, this large bandwidth advantage comes at the cost of a paradigm shift for the automotive industry, because Automotive Ethernet differs significantly from the topology of bus systems such as CAN, LIN and FlexRay. There are no bus conductors to which numerous ECUs, sensors and actuators are jointly connected. Instead, it is a network of switches and many electrical point-to-point connections. In such fully switched networks, the topology – i.e. the layout of individual nodes – plays an important role, especially when it comes to real-time capability. Technologies such as AVB (Audio Video Bridging) and TSN (Time Sensitive Networking) have implications for switches and ECUs, and in the future they will enrich Automotive Ethernet networks with functions such as prioritization of data packets and time synchronism.

All of this noticeably increases complexity and has effects on development, testing and simulation of the related networks. Convenient network access by test and simulation tools has always been an important aspect of the development process, and Automotive Ethernet is no exception. However, switched networks lack "the one bus" on which all signals and messages can be accessed or test messages can be sent out. This leads to the question: What capabilities and properties does an interface hardware device need to offer to meet current and future requirements?

Measurement With no Effect on The Network is Practically Impossible

The basic requirements of an interface for Automotive Ethernet do not really differ from those for other communication systems: with a rugged housing, reliable plug connectors designed for many plug-in cycles and an extended temperature range, the hardware is equally well-suited for the laboratory and for outdoor in-vehicle applications. The software tools require extensive network access, which means that the interface needs to be able to handle both passive read accesses and active write accesses. Important for all of these activities, of course, are highly precise time stamps and the ability to synchronize with other interfaces and other bus systems in multibus environments. The hardware communicates with the simulation or analysis tools via a stable and high-performance host interface.

Figure 1: Example of a switched Ethernet network that consists of multiple 1-to-1 connections.

As mentioned, a switched Ethernet network consists of many 1-to-1 connections (Figure 1) whose full-duplex transmission makes access without effect on the network nearly impossible. Even tapping the data signal at the network cable between two switches or ECUs with an ideal, high-impedance measurement system to passively listen to data traffic is of little use, because the tapped signals cannot be decoded. Only the counterpart ECU at the other end of the transmission line can decode the received information by differentially comparing it with its own signal from the signal mix.

In practice, there is no alternative to splitting a 1-to-1 connection of interest and inserting additional hardware such as a TAP (Test Access Point) (Figure 2). A TAP provides two Ethernet ports and a connection to the computer with the analysis tools that can distinguish the different operating modes of a TAP. For pure monitoring, passive TAPs are sufficient, which loop through the data on ISO-OSI Layer 1, i.e. the physical level. In this process, the TAP is essentially transparent with regard to key communication properties. Errors in transmitting Ethernet packets are passed and not rejected. To achieve time synchronization in the network, it is important to keep the unavoidable pass-through delay constant and minimal. However, as soon as one wishes to intervene in events and modify data, active TAPs are needed. Comparable to a switch, they operate on the Data Link Layer (Layer 2).

Functional Development vs. Network View

In pure functional development, a TAP is often sufficient, because here the focus is generally on an individual connection between two network nodes, e.g. from ECU-to-ECU or ECU-to-switch. Along with purely passive listening to individual connections, the software tools also perform tasks such as the remaining bus simulation and inserting additional data packets and disturbances. The situation

Figure 2: A typical measurement setup in which signals are accessed by splitting the 1-to-1 connections and looping them through interfaces. A common time base must be provided for precise analysis.

Figure 3: Data transmission from ECU 1 to ECU 5 via three connections and two switches.

becomes difficult when analyzing Automotive Ethernet scenarios that are more complex, in which developers need an overview of the entire system (Figure 3). Then the attention is on entire data paths. In addition to total transmission times and pass-through times in the switches, users are interested in detailed information about rejected messages and MAC address tables, for example. If users need to perform multiple accesses to the network simultaneously, they have to insert their interfaces at multiple points. If the latter are discrete devices, a synchronization mechanism is needed now at the latest, so that all interfaces are operating to the same time base and the tests can produce usable results.

In addition to network access, the topology also represents a challenge in switched networks. An example from practice illustrates the fundamental problem in testing an individual ECU: The real device under test is connected, via an Automotive Ethernet interface, to the test computer on which two virtual ECUs run for the remaining bus simulation. Usually the PC with the simulation tools has exactly one TCP/IP stack, so that the two virtual ECUs can communicate externally to the interface via the common stack. The data traffic generated by the PC does not by any means represent a real situation. That is because the ECU under test cannot distinguish the messages of the two simulated ECUs, since they act under the same IP-address and MAC-address. Conversely, the ECU under test is not able to address the virtual ECUs separately. It would be necessary to specially modify the ECU just to be able to perform testing – which is a highly inefficient working method.

Figure 4: Remaining bus simulation with multiple TCP/IP stacks. In the framework of integration, the simulated ECU 2 is replaced by a real ECU under consideration of the topology.

Shifting Complexity to Software and Hardware Tools

This situation can be remedied by a specialized tool that makes it possible to place a special TCP/IP stack instance after each simulated ECU (Figure 4). Together with the interface hardware, the user can then stimulate the real ECU so that the simulated communication matches the real communication. Over further course of testing, a colleague might erroneously come up with the idea – one that is familiar from CAN – of "just removing" one of the simulated ECUs from the simulation and replacing it by its real counterpart. When the user connect it to an available Ethernet port on the interface, would it work? Unfortunately, this approach ignores the fact that Ethernet networks only permit 1-to-1 connections. A 1-to-1 connection exists in the test layout, but it is already between the real ECU and the software with the remaining simulated ECU. In practice, other devices can only be added by establishing new, additional 1-to-1 connections, whereby the new network fragments need to be coupled to the existing network via switches. If the interface offers switch functionality internally or could be reconfigured to do so, the colleagues might successfully implement their plan and connect the second external ECU to an available port.

The Interface is More Than the Sum of its Ethernet Ports

To adequately meet user requirements, a high-performance Automotive Ethernet interface is equipped with numerous ports. Twelve 100BASE-T1 ports are sufficient to implement relatively complex test scenarios. But it is not just the number of ports that is crucial here, as has already been shown above. It is important to understand that – up to now – simply shifting a device between simulation and the real world was equivalent to a change in topology. If users are only equipped with conventional standard components and tools, such seemingly simple work steps can pose enormous challenges. Given this situation, what are the key properties of a powerful Automotive Ethernet interface that is suitable for the future? Ideally, an interface decouples the simulation from the direct Ethernet ports by inserting what is known as an application channel between them on the PC side. Moreover, it fills the gap between the application channel and the 100BASE-T1 ports with a user-configurable transfer function (Figure 4). A typical and frequently needed transfer function is the switch functionality.

The use of an FPGA makes it easy to implement further extionsions for new transfer functions, which can be selected and configured appropriately. The interface is therefore extremely flexible to adapt it to the practical challenges of different development and test phases. Furthermore, nearly every network configuration and topology can be mapped to a functionally identical test setup. If necessary, the user can configure multiple parallel application channels as well as transfer functions and connect them to the Ethernet ports. In addition to switches, transfer functions come into consideration, which include "PHY Bypass" (mapping on Layer 1) and "MAC Bypass" (store-and-forward principle on Layer 2) functions. The interface also allows the tools full access to the Ethernet ports at all times, either direct or relative, over the application channel. Native access to Ethernet frames at the bus transceiver

Figure 5: The new flexible VN5640 Automotive Ethernet interface offers 16 Ethernet ports, versatile transfer functions, highly-precise time stamps and extensive filter mechanisms for reducing data volumes.

during monitoring can reveal problems on the bus such as corrupt frames, which are filtered out by the MAC layer in conventional switches. This enables meaningful analyses that yield findings on what is happening directly on the bus and what is arriving at the ECU.

Scalable Line of Interfaces for Challenging Automotive Ethernet Applications

Vector Informatik has integrated all of the named functions and further functions into its new Automotive Ethernet Interface VN5640 (Figure 5). The device was specially designed for the application area of simulating or measuring on the network view level, and it has a total of 16 Ethernet ports: twelve for IEEE 100BASE-T1 and four for standard Ethernet connections such as 100BASE-TX or 1000BASE-T. A very fast host connection routes the enormous quantities of data that can arrive at the Ethernet ports. Since generally just a subset of the Ethernet frames is relevant for typical test and simulation tasks, the VN5640 hardware provides many filter mechanisms that can be activated. Based on the various protocols, the filters only pass the really necessary Ethernet frames to the PC and thereby make a significant contribution toward relieving loads in the simulation and test software.

Along with the usual interface mode, the VN5640 can also be used in stand-alone mode without a PC. In addition, there are future plans for letting the user connect an Ethernet-capable standard automotive data logger. Together with the VN5610 2-port interface, which has already been available for some time now, the VN5640 represents a scalable Automotive Ethernet interface family that works perfectly with the widely used CANoe test and simulation tool from Vector. In the future, many new functions of the VN5640 will also be supported by the VN5610 and be available after a driver update. All interfaces in this product line can be addressed over a common programming interface (API). In combination with CANoe, users can use description files to conveniently exploit the extensive configuration options. This also offers the advantage of sharing the files with other departments, suppliers and customers. Furthermore, CANoe offers access to extended statistical and switch information – such as quality information about transmission paths, the contents of MAC address tables and the queue states of switches – which provide useful insights regarding the loads of individual link segments.

Conclusion and Outlook

The proper methods for developing, testing and simulating Automotive Ethernet networks are generally more difficult than users are accustomed to with CAN, LIN and FlexRay, and the tasks require a differentiated approach. Modern interface hardware specially tailored to meet these new challenges offers valuable support. In combination with test and simulation tools that have been tuned for the new requirements, this interface hardware will sustainably simplify and accelerate development and test processes.

To meet the growing demand for bandwidth, IEEE is already working on a 1-GBit variant of 1000BASE-T1. Due to the flexibility in hardware setup the VN5640 interface is prepared for other physical layers and can be available quickly. Technologies such as AVB (Audio Video Bridging) and TSN (Time Sensitive Networking) will, in the foreseeable future, provide for guaranteed worst-case latencies, prioritized data transmissions and time synchronism. As they become available, these and other advanced developments will also be implemented in firmware and software updates so that users will always be able to work with Automotive Ethernet interfaces from Vector Informatik that represent state-of-the-art technology.

Peter Fellmeth (Dipl.-Ing. (FH))
is a group leader and product manager at Vector Informatik GmbH. He is responsible for the development of products for Automotive Ethernet, J1939 and ISOBUS.

Matthias Schwedt (Dipl.-Ing. (FH))
is a product manager for network interfaces at Vector Informatik GmbH. In this position he is responsible for the VN5600 Automotive Ethernet interface family.

Translation of a German publication in Elektronik automotive Ethernet edition April 2016

Image rights: Vector Informatik GmbH

Time Synchronization in Automotive Ethernet Networks

Balancing Act Between AUTOSAR, IEEE, and TSN

Numerous system functions in the automobile require high-precision synchronization of the underlying time bases in the ECUs. The entry of Ethernet into the automobile requires that developers and system architects have to deal with this issue because existing methods cannot simply be transferred to Ethernet. Coming from different directions, specialists are working in committees on making Ethernet capable for demanding time-critical tasks. This article presents an overview of time synchronization methods relevant to the automotive industry, points out the current limits of Ethernet networking, and outlines future solutions and improvements.

There are many different time-critical Ethernet applications in the automobile. They include advanced driver assistance systems (ADAS) with cameras and (radar) sensors, real-time audio and video streams, and backbones for synchronizing various physical domains **(Figure 1)**. Another use case is on-board data logging. Moreover, devices networked via Ethernet must naturally also be synchronized with other physical systems such as CAN and FlexRay.

Originating from the computer networking and office environment, Ethernet lacks the real-time capability necessary for technical applications. Although switched Ethernet topologies reduce unwanted collisions during network access, a fully deterministic behavior cannot be realized by this alone. In particular, the sensitive task of time synchronization requires complete information along the transmission path on the run times and latencies that are inherent in switches and gateways.

Anyone wanting to control processes in connection with Ethernet is confronted with these challenges. This explains why various standards development organizations are working in parallel to add real-time capability to Ethernet, including mechanisms for time synchronization. The principal methods for time synchronization in the automotive industry are currently based on AUTOSAR 4.2.2, IEEE 802.1AS, and the revised IEEE 802.1AS-ref, which was developed by the Audio/Video Bridging Task Group, which is now known as the TSN (Time Sensitive Networking) Task Group.

The Global AUTOSAR Time Concept

The concept of AUTOSAR 4.2.2 provides a statically defined Global Time Master (GTM). This ECU, distributes the time in the global time domain with maximum precision for the entire network. Subdomains derived from this can extend beyond various physical media. Time Gateways are passing on the time starting from one slave port through one or more master ports to the end points (Time Slaves) or other Time Gateways **(Figure 2)**. The synchronized time must be corrected by the amount of the Residence Time in the gateways. Alternatively, AUTOSAR also allows the establishment of separate independent time bases that are 'filled' with the current time via the respective gateway.

The type of network determines the details of the synchronization process. For example, with CAN and Ethernet, the Time Slave corrects the received global time base by comparing the time stamp from the transmitter side with its own receive time stamp. With FlexRay, the synchronization is simpler because FlexRay is a deterministic system with fixed cycle times that acts in a strictly predefined time pattern. The time is thus implicitly provided by the FlexRay clock. While the time stamp is always calculated by software in CAN, Ethernet allows it to be calculated by either software or hardware. In addition, up to 16 time domains can be implemented for CAN and FlexRay. Instead, Ethernet works with only one single time domain, the "Global Time Domain".

Basis for Synchronization of Ethernet ECUs: IEEE 802.1AS

Since AUTOSAR 4.2.1, the "Synchronized Time Base Manager" (StbM) has been available, which abstracts the respective bus-specific time base generator (Bus Provider Module) for FlexRay, CAN, and Ethernet to the run-time environment and makes the "Presentation Time" available for the application. The individual protocols for the respective physical bus are implemented in the Bus Providers, whereby the Ethernet Bus Provider uses the "generalized Precision Time Protocol" (gPTP) based on IEEE 802.1 **(Figure 3)**. To harmonize this with typical use cases in the automobile, various modifications, additions, and in some cases restrictions were needed within the scope of AUTOSAR.

To point out how Ethernet differs from CAN and FlexRay, we will start by summarizing the essential properties of CAN and FlexRay synchronization: Both have 16 synchronized time bases and optionally up to 16 statically connected offset time bases. The time information is transmitted with two messages in CAN. The first message contains the seconds information and the second message contains the nanoseconds information. For this reason, the second message contains an overflow detection in order to intercept possible overflows in the nanosecond range. A "Time Gateway Synchronization Status" is used by the application to detect whether it is operating synchronously with the subdomain or the Global Time Master. There is also a Sequence Counter and a Cyclic Redundancy Check (CRC).

Figure 1: Ethernet use cases in the vehicle

Automobile-specific gPTP Implementation

The standard method for time synchronization in a switched Ethernet network is described in IEEE 802.1AS and comprises the following three core procedures: Distribution of the synchronized time (Sync/Follow_Up), measurement of the message run time (Pdelay), and Best Master Clock Algorithm (BMCA). A two-step procedure consisting of a Sync message and a Follow_Up message is responsible for the actual synchronization. Pdelay is a special protocol for measuring the message run time between two ports. In addition, with the help of Pdelay, ECUs can detect whether they are part of a "Time Aware System", a network with time synchronization capability. By means of the Best Master Clock Algorithm, network nodes, preferably in dynamically structured networks, determine the ECU that supplies the best system time and elevates it to the Grand Master.

Implementation in the automobile, however, deviates from the IEEE standard in quite a few aspects. While the IEEE standard permits various dynamic configurations, for example with regard to the network topology or the BMCA (Grand Master selection), a lot is statically predefined in the automobile. This is necessary to achieve unique system descriptions and optimized communication matrices. Other deviations can be found in the time stamp support in software or hardware, the (lack of) payload data, and the need to equip vehicles with VLANs for security reasons.

Expanded Switch Drivers Supply Port Information

A serious obstacle to precise time synchronization via Ethernet remains – as of AUTOSAR version 4.2.2 – the limited switch functionality of Automotive Ethernet. The switch drivers cannot forward the port number of the currently received message to the overlying layer. This affects the port-specific sending of Sync and Follow_Up messages and practically disables the Pdelay mechanism for switches. Because an outgoing Pdelay request triggers multiple responses here (Pdelay_Resp/Pdelay_Resp_Follow_Up), these can no longer be assigned to the individual communication ports. The Pdelay therefore does not supply any useful information. In addition, switches in the current specification do not allow any (separate) port-specific time stamping. The delays in the switches cannot be determined and a precise time correction is therefore not made. AUTOSAR 4.3 will improve this situation and provide more support for IEEE-802.1AS functions. In particular, port-specific switch information will be consistently supported – from the switch driver and the Ethernet driver to the basic software components – so that the urgently needed input and output stamps will then be available.

Figure 2: The global time concept in AUTOSAR is specified for CAN, FlexRay, and Ethernet.

Figure 3: Simplified representation of the AUTOSAR software architecture for the global time concept

Payload Data: The Essential Ingredient

When IEEE 802.1AS is used in the automobile, action is still needed in a few other places. This involves issues such as (OEM-specific) "payload data", the support of multiple time domains, and redundancy and backup strategies. Payload data is understood as additional information in the Follow_Up messages that the standard does not cover but is essential for a specific application or functionality. Examples are debugging information and expanded status information. The solution is to place additional information in an AUTOSAR-specific Type Length Value (TLV) field in the gPTP messages. The Time Slaves would then evaluate this field or not, depending on their configuration.

Time Domains in Short Supply in the IEEE-Standard

While IEEE 802.1AS currently provides only one time domain, automotive applications require many more time domains for things like the UTC time, GPS time and evaluation of sensor-specific signals. Therefore the existing domain number in the gPTP message header must be enabled for values greater than zero. According to IEEE 802.1AS, only "0" is currently allowed as the domain number and all other numbers are ignored by the nodes. As an addition to IEEE 802.1AS, at least one other time domain is included in the current TSN specification (IEE 802.1AS-ref), but this will not be sufficient for the requirement in the automobile. As a solution, AUTOSAR offers the possibility to define additional time domains.

Backup Master To Ensure Fail-Proof Synchronization

Safety and security considerations are increasingly taking center stage in future vehicle architectures and AUTOSAR versions. This quickly raises the issue of a backup master or generally of redundancy concepts if the sole, statically defined time generator (Global Time Master) fails **(Figure 4)**. For this reason, at least a second time generator is needed that runs continuously as a backup master. The Best Master Clock Algorithm in its current form does not appear to be a solution for this. It is relatively slow and operates on the basis of timeout detection so that ECUs may already enter the synchronization timeout state that is actually meant to be avoided. Rather, the masters would have to substitute for each other before a synchronization

Figure 4: Increase of robustness for safety and security applications through redundancy and backup concepts for the Global Time Master

timeout takes place. These and many other issues will continue to ensure ample discussion material in the standards development committees.

Conclusion

The IEEE 802.1AS is a good reference for time synchronization for Ethernet and, with corresponding implementation, opens up the possibility to use the gPTP protocol according to the IEEE standard in the infotainment area. The coming TSN standard will take into account appreciably more automotive requirements. However, it is still not apparent how comprehensive that will be. In any case, IEEE and AUTOSAR are complementary and are moving toward each other. Due to different release cycles, however, that will still take some time.

It is important that system architects, developers, and suppliers are able to advance their Ethernet projects efficiently and economically. A valuable aid for this are development tools, such as the CANoe test tool and the AUTOSAR basic software MICROSAR of Vector Informatik, which is available – amongst others – for specific Ethernet hardware. The products support either time synchronization with gPTP according to AUTOSAR or gPTP according to IEEE. In addition, functions and software solutions are available that use the synchronized time for audio/video streaming, for example.

Author
Dipl. Ing. (FH) Bernd Jesse is Senior Product Manager of Embedded Software at Vector Informatik GmbH for the Ethernet AVB/TSN solution and is actively working on the global time concept in AUTOSAR.

Translation of a German publication in Automobil Elektronik, issue July/August 2016

Image rights:
Image 0, 1, 3 and 4: Vector Informatik GmbH
Image 2: Original version "AUTOSAR Specification of Synchronized Time-Base Manager, Release 4.2.1" (Section 2.2.7 on page 11), edited version: Vector Informatik GmbH

The Future with SOA, POSIX, TSN

Automotive Ethernet: Trends and Challenges

Ethernet is a driver of innovation in vehicles. The two main reasons for this are the available bandwidth as well as the range of functions and the flexibility of this network technology. Based on these premises, this is a good time to take a closer look at three technological trends: The modeling of service-oriented architectures and Automotive Ethernet networks; the use of Automotive Ethernet in POSIX-based ECUs; and reliable communication using Time Sensitive Networking.

The growing use of Ethernet in vehicles is resulting in a number of factors which influence one another. Automotive requirements encourage technological innovations and advances, for example mechanisms for the transfer of time-critical data (Time Sensitive Networking). At the same time, Ethernet is permitting changes to the E/E structure in vehicles and is making it easier to use service-oriented architectures.

What is happening in the field of Automotive Ethernet? In the IT industry, service-oriented architectures (SOA) have already been used for many years to describe and structure distributed systems. However, service-oriented design is now also growing enormously in importance in the automotive industry.

Figure 1: Service-oriented architecture and service distribution

Modeling of Service-Oriented Architectures and Automotive Ethernet Networks

An SOA creates a service landscape that is characterized by clearly defined interfaces **(Figure 1)**. Ideally, these interfaces should be correctly described both syntactically and semantically.

In addition, in an SOA, the software components are connected via middleware known as a "service bus". The middleware mediates between the two roles that participate in a service: The service provider and the service consumer. The SOA provides a loose coupling between the service roles in a system. It is therefore more flexible than monolithic software architectures. Two other positive aspects are the increased reusability and better ease-of-integration of new service components. The middleware regulates communication between service provider and service consumer as well as the establishment of this communication. It also usually defines the serialization of data in the Ethernet frame. The serializer determines how the data is converted into a serial bit stream and is then deserialized again after reception.

Nowadays, there are various description languages for interfaces, known as interface description languages (IDL). These typically have technology-independent components as a formal description of a service. In addition, they can also have protocol-specific elements. Technology-independent properties include available methods, fields and events, including the data types used. The middleware-specific startup behavior, on the other hand, can be regarded as a technology-dependent characteristic. These include service discovery (SD) and the use of data serialization and protocol-specific identifiers (ID). IDLs are particularly well suited for the formal description of individual interfaces and automatic code generation. They are mostly present in hierarchically structured text files. However, text files are not particularly well suited for designing complete SOAs. As a result, model-based design tools are becoming more popular. One strength of modeling is that the data is managed centrally and that the maintenance effort is therefore greatly reduced. Model-based design also guarantees that the data is formally correct at the time of design. The clear definition also makes it possible to structure the services. In addition, larger systems usually use layer architectures that make the dependencies of services understandable and represent clear design rules. When services build on each other, the necessary design is called "service choreography". In a model-based outline design, the data communication can be partially derived based on the modeling of the service. At the same time, model-based design makes it possible to manage complexity considerably better than in the IDL-based approach.

The use of an SOA permits the dynamic establishment of communication relations at runtime. However, in current automotive systems that already use SOA, service communication is often still statically designed due to limited resources. Thanks to the use of middleware, the connection between the service provider and service consumer is established at runtime – and not at the time the system is designed, as in the past. In the future, we can assume that fully-dynamic service communication will predominate, especially in systems with high-performance hardware. Infotainment and driver assistance systems, in particular, are driving this trend forward. External communication with online map services or OTA (over-the-air) updates is also supporting this trend. Service communication requires much more protocol information at the network level than a signal-based approach. Due to the required bandwidth, Ethernet is therefore the main network technology in the vehicle.

Automotive Ethernet in POSIX-Based ECUs

The modeling of service-oriented E/E architectures and the lower-level Ethernet networks as described in the section above is already relevant for many ECUs of different performance classes. The spectrum extends from intelligent sensors such as camera, radar and LIDAR systems through to central high-performance computers that implement sensor fusion and the higher-level driver assistance applications. ECUs in the lower and medium performance classes are often based on a conventional AUTOSAR software architecture.

However, other operating systems are necessary for high-performance ECUs. On the one hand, support is needed for the corresponding hardware accelerators, for example for graphical units. On the other hand, there is a need for greater flexibility in application handling and the updating of data. This is why POSIX-based operating systems are mostly used in this field. The best-known software solutions in the automotive environment include Linux – for safety-relevant applications under Linux, a hypervisor such

Figure 2: Use of AUTOSAR Classic modules in ECUs with POSIX operating system

Figure 3: TSN is extending Automotive Ethernet for time-critical data in the areas of scalability, reliability and timing guarantees.

as Sysgo's Pike-OS is also necessary – and Integrity from Green Hills Software and QNX. However, these operating systems do not provide any integrated support for Automotive Ethernet. To meet automotive requirements, a number of specific protocols based on Ethernet and TCP/IP have been defined for vehicle use. These include Scalable Service-oriented Middleware over IP (SOME/IP) for efficient, service-oriented communication, Diagnostic Communication over IP (DoIP) for vehicle diagnoses, User Datagram Protocol-based Network Management (UDPNM), Universal Measurement and Calibration Protocol (XCP) for measurement and calibration tasks, as well as time synchronization as specified in IEEE 802.1AS. Unlike AUTOSAR Classic, modern POSIX systems have no native support for these protocols. However, they do support subsequent integration.

One possibility is to execute proven AUTOSAR Classic modules in a POSIX process **(Figure 2)**. In AUTOSAR Classic, SOME/IP-SD, DoIP, UDPNM, XCP and conventional signal- and PDU-based communication (PDU: Protocol Data Unit) do not directly access the AUTOSAR TCP/IP sockets because these are encapsulated by the Socket Adaptor Module. However, if the socket adaptor uses Berkeley sockets (BSDs) instead of AUTOSAR sockets, then this module – and therefore also the above-mentioned automotive-specific protocols – can be executed within a POSIX process.

If we go a step further and also emulate the AUTOSAR operating system, then it is even possible to use the runtime environment (RTE) and associated SOME/IP transformer within a process. Layer 2 protocols such as the time synchronization mentioned above and the audio/video transport protocol (AVTP) according to IEEE 1722 must be considered separately. Since these are not based on TCP/IP communication, the BSD sockets used for this purpose must be opened in a special mode that allows access to the entire Ethernet frame. An adapted Ethernet driver and the corresponding AUTOSAR Classic modules allow the implementation and use of the automotive-specific layer 2 protocols in a POSIX process.

A second possibility is to use AUTOSAR Adaptive. This is a runtime environment that has been standardized for POSIX operating systems and that will be used for many innovative automotive applications in the future. AUTOSAR Adaptive provides automotive-specific extensions for fully dynamic POSIX applications. In the field of Ethernet communication, these are presently SOME/IP and DoIP. Work is currently being undertaken to support further protocols. The BSD sockets of the POSIX operating system are also used for TCP/IP-based communication, thus simplifying integration. For the support of layer 2 protocols in the context of time-sensitive networking (TSN), OS-specific adaptations may be necessary in AUTOSAR Adaptive, since some additional information required is not available via the standard POSIX interfaces. The question of which of the above variants is most suitable for an ECU project depends on various factors. For example, if it is possible to call on existing AUTOSAR software components (SWCs) or diagnostic implementations, then one good solution may be to use conventional AUTOSAR modules under POSIX. However, if the full flexibility of native POSIX applications is needed then AUTOSAR Adaptive is recommended.

Reliable Communication Using Time-Sensitive Networking (TSN)

In vehicles, real-time capabilities are vital for certain applications **(Figure 3)**. To ensure that these are available, it is necessary to have mechanisms that directly access the hardware resources in the Ethernet controller. TSN makes it possible to build a scalable Ethernet network. To do this, various message classes are scaled in terms of availability, and their latency and priority are categorized. Each message class is assigned a guaranteed bandwidth. In addition, redundant Ethernet systems are supported and security criteria are defined to ensure stable data exchange.

The IEEE Audio/Video Bridging Task Group defined mechanisms and protocols to ensure low-latency data exchange and synchronize applications in terms of time. Audio/Video Bridging (AVB) is mainly aimed at infotainment applications. The widespread introduction of driver assistance systems, on the other hand, calls for even stricter rules with

Figure 4: The TSN bridge only passes on data streams that are registered with it.

Figure 5: TSN mechanisms in an Automotive Ethernet network.

regard to sending and receiving behaviour. Therefore, the IEEE Time-Sensitive Networking Task Group continues the work of the Audio/Video Bridging Task Group. The focus of this working group is on deterministic data transmission, further latency reduction and stable and secure transmission in Ethernet networks **(Figure 3)**.

Time synchronization using gPTP as described in IEEE 802.1AS ("Timing and Synchronization for Time-Sensitive Applications") is essential to ensure that Ethernet networks with time-based data analysis are stable. In this process, a Grand-Master sends a global time to all time-sensitive ECUs. In addition, the average runtime of a defined Ethernet frame is measured and taken into account when calculating the synchronized time. In this way, all time-sensitive ECUs can synchronize to a discrepancy of no more than one microsecond.

The sender and recipient then assign this synchronized time to the individual data packages. The data is distributed to the endpoints using a specific send procedure (Shaper). During this procedure, the credit-based shaper that is known from audio/video bridging ensures that the data arrives at the recipient within a defined time (presentation time). By contrast, the time-aware shaper added by TSN ensures that the data is deterministically sent in accordance with a fixed, predefined time grid. In particular, this must be ensured by every switch. For a specific portion of the bandwidth, this results in a time-driven network, similar to the case of FlexRay. The two shaping procedures can coexist without problems in parallel with the regular Ethernet traffic in a network. This is an advantage compared with FlexRay.

TSN also supports mechanisms to favour high-priority Ethernet packets over low-priority Ethernet packets that are currently being sent without waiting for their complete transmission. For this purpose, the latter are preempted at the next possible point in time in favor of the Ethernet packet with higher priority and continued later. This method reduces the maximum latency even further without affecting the rest of the transmission behavior.

The Stream Reservation Protocol **(Figure 4)** is used to monitor the bandwidth. In this protocol, the TSN bridge checks that only data streams that are registered with it are passed on. At the same time, the current message volume for a message class must be in a previously defined range. If this budget has been used up, this data stream cannot be sent at the present time.

In the case of safety-relevant systems, it is also possible to duplicate the packages for redundant transmission over separate network paths. In addition, filter and control mechanisms ensure that only message classes that are actually expected are passed on and that this only takes place with the required frequency. Both increase the robustness of Ethernet communication in the event of an error - whether due to interrupted network structures or unwanted Ethernet traffic.

Many of the TSN mechanisms require hardware support **(Figure 5)**. The TSN specification IEEE 802.1Qbv ("Enhancements for Scheduled Traffic"), which is now stable, is playing a leading role and is already supported by a wide range of hardware. By contrast, in the case of IEEE 802.1Qbu ("Frame Preemption"), it remains to be seen how the interrupt mechanisms on Ethernet will establish themselves. In Gigabit Ethernet, the shorter wait times through to the transmission of a high-priority frame will hardly make any difference. Instead, this will more likely have a significant influence on 10/100 megabit systems.

Outlook

In future vehicle platforms, the dynamic protocols and methods that are familiar from IT are becoming fused with conventional automotive requirements in terms of time response, availability and robustness. Service-oriented communication and safety-relevant and deterministic Ethernet traffic are merging to form complex systems. On the one hand, at design time there is no binding specification about the ECU on which a function is to be implemented and the runtime environment in which it is to be executed. On the other, the network architecture must guarantee reliable communication that meets the needs of the application.

With its embedded software, Vector is already able to meet these requirements today. What is more, Vector's Ethernet design and test tools provide ways of representing and testing dynamic, service-oriented and time-sensitive communication relations. Thanks to Vector's active work on various boards, the market also benefits from the early availability of solutions for future trends in the automotive industry.

Author
Bernd Jesse
Graduate Engineer and Senior Product Manager for Embedded Software at Vector Informatik and responsible for Solution Management for the Ethernet AVB/TSN products. He supervises the Global Time concept in AUTOSAR.

Author
Marc Weber
Master of Engineering and Senior Product Manager for Embedded Software at Vector Informatik and responsible for the management of the Ethernet solution.

Author
Markus Helmling
Graduate Engineer and Senior Product Manager for Embedded Software at Vector Informatik and responsible for Solution Management for the Ethernet solution as of 2017. Before that, he worked on product management for the design of AUTOSAR communication in PREEvision.

Translation of a German publication in magazine Automobil-Elektronik, issue 11-12/December 2017 (unabriged version)

Image rights: Vector Informatik GmbH

From Signal to Service

Challenges for the Development of AUTOSAR Adaptive Applications

Automotive Ethernet and AUTOSAR Adaptive are key technologies for highly automated driving and comprehensive connectivity services. AUTOSAR Adaptive relies on a service-oriented architecture to ensure communication between the ECUs." This raises many new challenges at the level of the development, test and calibration tools.

Today's applications in the automotive field primarily make use of sensor/actuator combinations. In these solutions, the participating software and hardware components are closely coupled. The communication relationships between the participating ECUs are statically defined during the system design phase and are based on signal-oriented communication. The focus is placed on hard real-time capabilities and the use of cost-effective microcontrollers as well as of networks such as CAN and LIN. The established AUTOSAR Classic platform reliably covers these requirements. The focus is not on extending or adding applications.

Highly-automated driving and back-end connectivity demand dramatically enhanced computational performance and more flexible communication concepts. In addition, it must be possible to extend or add applications across the entire vehicle lifecycle. Ethernet is a suitable transmission medium for this generation of applications. On the one hand, it provides the bandwidth that is required, for example, for the transfer of image data. On the other, IP-based protocols running on Ethernet allow the simple integration of end-user devices and access to back-end infrastructures.

AUTOSAR Adaptive – A Brief Overview

The AUTOSAR Adaptive platform **(Figure 1)** addresses the requirements of these future applications. It forms the central computing cluster for complex applications and establishes the connection to back-end services. At the same time, it interacts with AUTOSAR Classic ECUs in order to control the sensors and actuators. Consequently, AUTOSAR Adaptive is not a substitute for the Classic platform but instead a complementary development platform. The main differences between the two platforms are outlined in **Table 1**.

As the central architectural pattern, AUTOSAR Adaptive adopts a service-oriented approach **(Figure 2)**. Within this, the overall system is modeled as a set of service interfaces.

Figure 1: The AUTOSAR Adaptive platform forms the central computing cluster for complex in-vehicle applications.

These encapsulate a certain sub-functionality comparable to a class in object-oriented programming. The service interfaces are implemented by so-called providers and are used by consumers. The communication path between a provider and a consumer is no longer defined at development time but only at runtime. The establishment of the communication paths and the conversion to a physical transmission medium is performed by a middleware. This approach permits the flexible provision and updating of applications across the entire vehicle lifecycle.

Currently, AUTOSAR Adaptive supports the Ethernet-based SOME/IP protocol (Scalable service-oriented middleware over IP) as the middleware protocol. In the future, further protocols will also be integrated. REST-based (Representational State Transfer) services using HTTP are expected to be used here for communication to the back-end services, for example. With AUTOSAR Adaptive, system architectures from conventional information technology are making their way into vehicle applications.

Testing Adaptive Applications

The introduction of these system architectures is confronting development and testing engineers with new challenges. Development and test methods from the conventional ECU development field have to be extended and complemented. Thanks to the service-oriented approach, future applications will be developed largely independent of any assignment to a specific ECU. Suppliers will frequently not provide a fully integrated ECU but instead only a set of applications. To do this, they must be able to test these in isolation. Similarly, in the future, vehicle manufacturers will perform testing more intensively at the level of individual subsystems. This is the only way to control the rapidly growing complexity of the overall system.

The tests of the lower communication layers must be strictly separated from the application tests. The mapping of service interfaces to concrete network protocols should therefore take place downstream and be interchangeable. Such an approach makes testing possible throughout the entire development process. Tests and simulation models can be defined in the early phases and then be extended incrementally. This permits the gradual transition from purely simulated to partially simulated environments.

The service interface level is clearly suitable for application-related testing. These interfaces encapsulate the behavior of the applications and form a clearly defined system boundary. They are defined within the framework of a service-oriented system design and are available even in

	AUTOSAR Classic	AUTOSAR Adaptive
Initial Release	2005	2017
Programming Language	Procedural (C)	Object-oriented (C++)
Communication Model	Signal-oriented – with focus on CAN, LIN, FlexRay	Service-oriented – with focus on Ethernet
Operating System	OSEK-based > Cooperative multitasking > Static number of tasks	POSIX-based (PSE51) > Preemptive multitasking > Dynamic number of tasks

Table 1: The most important differences between AUTOSAR Classic and AUTOSAR Adaptive

early development phases. AUTOSAR Adaptive describes the service interfaces and data types within the standardized exchange formats. As a result, the programming interfaces required for testing are generated automatically.

For development and test engineers, a type of service toolbox from which they can choose the service interfaces required for the current test goal would seem useful here. These interfaces are then stimulated and analyzed in the test environment. In such cases, it should be possible to modify the timing conditions during a test, for example in order to check how a consumer reacts to different provider response times.

The dynamic establishment of communication paths at runtime is a further challenge. First of all, it is necessary to take account of the fact that the availability of providers and consumers can change at runtime. Secondly, in some scenarios, the participating static communication endpoints are unknown, for example in the case of data exchange between vehicles. In such cases, development and test engineers need a flexible way of varying the number of participating communication endpoints.

Tools for Service-oriented Testing

One test tool that is available on the market and that supports service-oriented testing is CANoe from Vector. This offers a uniform, cross-network communication concept. This means that CANoe provides a network-independent interface for the application models and test scripts. Actual access and transmission over a physical network are configured independently of this. This approach permits application-related modeling of service-oriented communication. Service interfaces are imported from AUTOSAR Adaptive descriptions and are modified and extended as required **(Figure 3)**. To this end, CANoe is based on the idea of a service toolbox **(Figure 4)**: Users are free to choose the service interfaces required for the current development status and the specific test objective.

The service interfaces, together with the associated methods and events, are available as modeling artifacts directly within CANoe. Users stimulate and visualize these directly using interactive controls. At the same time, they are also available via C#-based programming interfaces within test and simulation scripts.

The CANoe communication concept permits the dynamic creation of communication endpoints during the simulation. At the same time, it is possible to modulate the availability of a service. It is therefore possible to simulate dynamic topologies of the sort that occur in the connectivity field. To permit early validation, CANoe supports the concept of application-level simulations for which users do not have to configure any low-level protocol layers. For the development and testing of real ECUs, CANoe provides connections via Ethernet and SOME/IP.

Challenges in the Field of Measurement and Calibration

Another area that needs to be taken into account when considering Ethernet and AUTOSAR Adaptive is measuring and calibration. This does not relate to use cases based on the standardized XCP calibration protocol but to the use of service-oriented communication as a transmission protocol for measurement and calibration data. The use of point-to-point connections in the service-oriented architecture makes this approach radically different from solutions in which the measured data is read out via a bus topology. How can service-oriented communication be used for measurement and calibration? The SOME/IP protocol used in

Figure 2: In service-oriented communication, an application provides a service that can be called by other applications.

Figure 3: CANoe and CANape filter out the unneeded service interfaces using a type of service toolbox.

AUTOSAR Adaptive enables the acquisition and writing of data on the ECU side.

The acquisition of measurement data can be subdivided into two use cases: On the one hand, measurement data can be recorded simply by monitoring bus activity. In the case of Automotive Ethernet, this is achieved by interrupting the Ethernet connection and using a so-called network TAP (Terminal Access Point) to create the connection to the measuring tool. In a switched Ethernet network, two participants communicate via point-to-point connections.

Figure 4: The CANoe user chooses the service interfaces needed for the specific use case from the service toolbox.

It is then possible to monitor the data exchanged between these two participants and extract and record it. On the other, data can also be acquired by executing a service in an application. In this case, the measuring tool behaves as if it is another consumer of the service's data. The advantage of this method is that it makes use of functionality that already exists in the ECU. Data serialization is implemented within SOME/IP and the description of the data is present in the service interfaces.

This active pathway also makes it possible to write data to the application. Here again, the data that is to be transferred is described via the service interface. Thanks to a relatively simple extension, it is possible to add or extend a service during development and then use it for calibration.

However, this active communication method leads to an additional load on the system and the existing communication paths – just as when performing measurements using XCP on CAN. In the case of data-intensive measurement and calibration operations, it is therefore preferable to choose a solution that makes use of the ECU's debugging or trace interfaces, for example.

Measuring and Calibration Tools

Vector's CANape calibration tool is a good way of acquiring measurement data from service-oriented ECUs. It is based on the service interfaces and is able to import AUTOSAR-Adaptive descriptions **(Figure 3)**. In this case, CANape also imports the communication descriptions present in the files. This makes it easier for users to configure their measurement setups. CANape uses the service interface

Figure 5: CANape records the measurement data either by actively requesting it from the service providers or by monitoring the communication between two participants.

definition to interpret the measurement data and is able to save it in the usual measurement data format, such as MDF for example. As mentioned above, the data is acquired either by monitoring into communications between two participants or by actively requesting the data from the service providers **(Figure 5)**.

Conclusion

AUTOSAR Adaptive transforms vehicles from being closed systems into systems that are networked with the environment. As a result, system architectures from the conventional IT world are increasingly being introduced into vehicles. Existing test concepts and tools must take account of this. First tools for the testing and calibration of service-oriented applications are already available. They enable development engineers to reduce the effort involved in constructing their test environments. In the future, the automotive industry will see the emergence of ever more tools that support the principle of service orientation.

Dr. Simon Frohn
studied Computer Science at the University of Stuttgart and received his PhD from the Chair of Computer Networks at the University of Leipzig. In 2012 he joined Vector and, as Senior Software Development Engineer in the SOA & Adaptive team, is responsible for the development of CANoe.

Fabian Rees
Fabian Rees studied Computer Engineering at the University of Constance and joined Vector in 2011. As a software manager, he is responsible for AUTOSAR Adaptive in the field of measurement and calibration.

Translation of a German publication in Elektronik automotive, Special Edition Automotive Ethernet, March 2018

Image rights: Vector Informatik GmbH

Testing Car2x Applications

Requirements for Test Tools Based on Example of the Road Works Warning

Carmakers plan to introduce Car2x communication in production vehicles starting in 2015. Car2x application development, which has already begun, poses new challenges in the execution of component and application tests. This article derives related requirements for test tools based on the example of the Car2x "Road Works Warning" application.

Intelligent and safe – that is how the German Transportation Minister described the expressways of the future in mid-June of this year. These objectives are to be achieved by introducing intelligent transport systems and services (C-ITS, "Cooperative Intelligent Transport Systems") that are based on Car2x communication. What is meant here is communication between vehicles and the infrastructure with the aims of improved safety on roads and early avoidance of traffic jams. As a first step, warning trailers will be equipped with the necessary Car2x technology for radio transmission of road works information to vehicles.

The scenario of the Road Works Warning is one of the Car2x applications already defined by the European Telecommunications Standards Institute (ETSI) [1], and plans call for technically implementing it starting in 2015. Here, the warning trailer sends its information in real time to vehicles within WLAN radio range per the WLAN standard IEEE802.11p (ITS-G5) **(Figure 1)**. A 5.9 GHz frequency band is available, which was specifically reserved for ITS-G5 usage. The GeoNetworking protocol specified by ETSI [2] is responsible for packet routing in ad-hoc networks; the information about the highway construction zone is contained in standardized application messages [3].

They include the "Decentralized Environmental Notification Message" (DENM), which contains all necessary information about an event (road works warning, end of traffic jam warning, etc.) and is only sent at the onset of the relevant event. Here, the ITS station transmits general information about event status and position and about the applicable duration and zone. For the Car2x "Road Works Warning" application, the warning trailer also sends information on the speed limit and lane closures, which are dynamically modified as a function of construction zone status and topology.

The ITS stations use the "Cooperative Awareness Message" (CAM) to periodically transmit their own status information such as position and driving direction, speed and acceleration information as well as status information about the vehicle lighting of the ITS station. Since all ITS stations send this information, the information lets the warning trailer perform such tasks as calculating the traffic density at the construction zone so that it can modify the speed limit in the construction zone accordingly or route traffic jam information to the central traffic control office.

Onboard Functionality of the Construction Zone Application

When a vehicle receives a construction zone specific DENM, it checks the message for relevance. It requires information such as the position, speed and driving direction of the vehicle to do this. Ideally, map data that describe the vehicle's environment more precisely will also be available. This information is typically supplied via the in-vehicle bus systems such as CAN and Ethernet.

In checking for relevance, information such as the time stamp lets the vehicle determine whether the received message is still valid. The location of the construction zone and of the vehicle is used to determine whether the construction zone is located along the driving route. If the message is relevant to the vehicle, the data is forwarded to the actual construction zone application. It ensures, for example, that other applications such as driver assistance systems or HMIs are supplied with necessary data when the vehicle enters the defined construction zone.

Increasing Efficiency with Test Tools?

To comprehensively test a construction zone application in the vehicle, application-relevant data for all necessary test scenarios must be provided to the application, and these scenarios must precisely match those of real driving situations. By using test tools, testers do not have to perform complex tests in the real environment. In early test phases, it is often impossible to conduct real tests, because not all components are available.

During development and test phases, it is even sufficient to simply provide the application developer with suitable data from just the communication perspective for testing. This involves creating a simulated environment for the ECU under test in test tools. This ECU can be put in all of the states to be tested using simulated functions and components.

This means that a "simulated road works warning trailer" is needed to test the construction zone application in the vehicle. It can be configured for each individual test so that different scenarios of lane closures and speed limits are possible, for example, and can serve as the foundation for tests. In parallel, the construction zone application is stimulated with data on a vehicle's geographic position, direction vector and speed for each individual test case. The test tool also simulates the motion profiles of the vehicle and generates the related CAN frames for stimulation. Afterwards, it provides this data to the construction zone application via the ECU's CAN bus interface **(Figure 2)**.

Test Tool Requirements for Car2x Applications

For a tool to be used effectively in testing Car2x applications, it must fulfill a number of requirements. The test tool must be able to send and receive WLAN data packets conformant to the IEEE802.11p (ITS-G5) standard over the

Figure 1: Simplified representation of communication in a construction zone scenario.

WLAN channels defined for this purpose. This includes the test tool's ability to interpret
protocol-specific information such as GeoNetworking. It must be possible to access signals and data fields of the application messages, which are described in ASN.1 (Abstract Syntax Notation.1) and are coded by the PER method (Packed Encoding Rules). A description of this information in a database that the test tool can read offers maximum flexibility here and simplifies the process of modifying data if changes are necessary. The database description serves here as the foundation for test generation. Using a database approach for creating and executing tests has already proven itself in established automotive networks.
In order to use an environmental simulation for stimulation purposes during the test, a programming tool is needed to create simulated nodes. Ideally, the tool would offer a Car2x-specific function library, which makes it possible to create and send application messages, set and read their signals and data fields, and execute PER coding of data before sending. Functions for generating UTC time stamps, which are frequently needed in the Car2x environment are helpful too. This makes it possible to configure valid construction zone specific DENMs for each individual test case.
An integrated test environment simplifies test execution. The tests are created with the help of editors and Car2x-specific function libraries and can simply be duplicated and modified for other test scenarios. Test flow control then handles execution of the selected tests and documentation of the results in a test report.
Since a lot of geographical positions are processed in the Car2x environment, it is extremely helpful to visualize them on a map. Information from the DENM is also intrepreted and displayed on this map **(Figure 3)**. At a glance this makes it easy to see exactly where the construction zone and vehicle are located, whether the relevance zones and waypoints are coded correctly, and what information is transmitted in the message regarding lane closures.
The construction zone application, or the ECU, obtains this information via internal vehicle networks such as CAN and Ethernet and externally via ITS-G5. Therefore, the test tool must support this process as well as offer multibus capability.

Summary

Car2x technology is not only being studied in the research departments of OEMs, but also by production development departments are now exploring how this technology might offer added value and how it might be used to better support driver assistance systems. The quality of these new functions must be assured by suitable test tools.
At this point, Vector supports the introduction of this technology in production vehicles with its CANoe .Car2x development and test tool. Along with tests for automotive networks such as CAN, FlexRay and Ethernet, tests can also be developed for the Car2x environment. They assist in developing and integrating Car2x technology in vehicles and in the infrastructure. With the signing of the Memorandum of Understanding initiated by the CAR 2 CAR Communication Consortium [4] Vector is assuring its customers that future test tool requirements will be implemented as well.

Translation of a German publication in Automobil-Elektronik, 6/2013

Image rights:
Vector Informatik GmbH

Literature references:
[1] ETSI TR 102 638 V1.1.1 (2009-06)
[2] ETSI EN 302 636-1 V1.1.0 (2010-03)
[3] ETSI TS 102 637-2 V1.2.1 (2011-03), ETSI TS 102 637-3 V1.1.1 (2010-09)
[4] Memorandum of Understanding, CAR 2 CAR Communication Consortium, Version 4.01.02 (2011-06-27)

Links:
Vector solutions for Car2x:
www.vector.com/vi_car2x_solutions_en.html
Product information CANoe .Car2x:
www.vector.com/vi_canoe_car2x_en.html

Thomas Löffler (Graduate Engineer)
is group leader and product manager for the ITS/Car2x area at Vector Informatik GmbH.

Cathleen Kunze (Graduate Engineer)
works as a technical writer for Car2x at Vector Informatik GmbH.

Figure 2: Schematic layout of a Car2x test scenario: The construction zone warning application is stimulated by the test tool to test functionality in various scenarios.

Figure 3: The entire test scenario can be implemented and visualized in CANoe .Car2x. Clear representation of the ITS stations in the map window with a display of the driving direction.

Car2x – From Research to Product Development

How Automotive OEMs and Suppliers are Successfully Completing Production Car2x Projects

Car2x systems present entirely new challenges for managers in product development departments. For one, the Car2x ECU under study must communicate with a large number of vehicles and beacons in its environment. This increases the number of information exchanges and their complexity compared to previous network development in production vehicles. For another, IP standards have now made their way into the vehicle; however, this is uncharted territory for most developers using the IEEE 802.11p air interface. These challenges can already be overcome with tools that are adapted to this interface.

Car2x communication (also known as Vehicle-to-Vehicle and Vehicle-to-Infrastructure communication) is the exchange of information between traffic participants and the infrastructure with the goal of enhancing safety and convenience and optimizing traffic flow. The higher-level engineering system for assuring Car2x communication is known as the Intelligent Transport System (ITS). The basic concept of Car2x communication involves sending and receiving standardized messages over the air interface and enabling interpretation of the status information they contain by traffic participants. The ITS station (ITS-S) keeps the messages up-to-date based on the momentary traffic situation and sends them either periodically or they are event-driven. The most important status information is transmitted via the message types CAM (Cooperative Awareness Message), DENM (Decentralized Environmental Notification Message), SPaT (Signal Phase and Time) and TOPO (Topology Specification). The European Institute for Telecommunication Standards (ETSI) has already specified the CAM and DENM messages. SPaT and TOPO are currently handled on a project-by-project basis. This system gives the intelligent processing units (ITS-S) of the receiving traffic participants, e.g. a vehicle, the opportunity to acquire information about the immediately relevant traffic situation over a broad area and to warn the vehicle driver if necessary or even intervene in vehicle control.

Scenario of the Broken Down Vehicle

In the following, the requirements of development tools that support the system manager in developing and validating the ITS-S are derived from the example of a Car2x scenario defined by ETSI [1]. Similar traffic scenarios are described in the CAR 2 CAR Communication Consortium and in the DRIVE C2X project [2].

In the "Car Breakdown Warning" scenario, the goal is to avoid having a broken down vehicle pose a hazard to approaching traffic or even cause an accident. Therefore, the ITS-S of vehicle A sends a standardized message that can be received within its WLAN transmission range **(Figure 1)**. An approaching vehicle B receives and processes this message and forwards it. This extends the WLAN transmission range so that even further distant vehicles C and beacons (Road Side Units – RSU) can receive the message and forward it. This gives vehicle C enough early notice to avoid the hazard area by choosing an alternate route. Thanks to the early warning, vehicle B can brake in time, e.g. when the view is impaired by fog or by obstacles such as a curve with limited visibility.

To assure that information is current and to avoid faulty information, a distinction is made of whether the message is coming from the original source A, or whether it was just forwarded by another sender (receiving vehicles B, C). Since forwarded messages have a limited life, they are only routed for a specific time period. Based on geo-positioning and a defined dissemination area, a decision is also made regarding whether vehicles B or C should forward the message at all.

Requirements of the ITS-Station in the "Vehicle Breakdown" Scenario

The ITS-S must derive a sufficiently complete picture of the traffic situation from the context of its surroundings, i.e. the totality of the CAM, DENM, SPaT and TOPO messages obtained from various sources, and it must initiate actions for its own vehicle. Real-time requirements are high here. Per specification, DENMs of the above example only need to be updated at a rate of 10 Hz. However, the latency time for the above scenario is specified as less than 100 ms [1]. This makes transmission via GSM unrealistic. The real-time requirement can only be satisfied with WLAN technology per IEEE 802.11p or LTE, and LTE cannot be considered currently due to its low coverage.

The ITS-S units in vehicles within the reception area must first decide whether received messages are relevant to their own vehicles; i.e. whether they are affected, and whether they should forward them. They are affected if they are located on the same street or on the way towards that street. This can be determined by the "heading" message contents of the received CAM message and "waypoints" in the DENM message. Other factors playing a role here are the route of the specific vehicle and information on the topology and status of traffic light systems. Finally, the ITS-S units must evaluate whether the information is potentially relevant to other units in the environment. If so, it must route the information correctly.

Requirements for Validating ITS-Stations

Software development tools can support the system manager in all phases of the V-model to assure functionality of the ITS-S. Unlike network development that is limited to a single vehicle, here it is absolutely necessary to consider the environment. This yields the following requirements for the tool.

Debugging of the Air Interface

In terms of measurement technology, the scenario described above can be reduced to **Figure 2**, possibly with a greater number of ITS stations. The functionality of the ITS-S is indeed standardized, but it is implemented by different manufacturers. In case of error, it is often first necessary to determine whether all participants are sending and receiving on the same radio channel. This requires support of the air

Figure 1: Vehicle A is at a standstill, and it sends DENM messages via its ITS-S to traffic participants B in its local surroundings. They extend the transmission range by forwarding the message to other vehicles C and roadside units D.

Figure 2: Challenge of the air interface: The data traffic can only be checked with an IEEE 802.11p compatible analysis tool.

interface as a communication medium. Only if it can be verified that communication over the air interface is operating correctly does it make sense to conduct a protocol analysis.

Protocol Analysis

The ITS-S system testing manager in product development needs to have the received message contents presented in an application-oriented way in the development tool, i.e. either as physical parameters (with units) or interpreted. For example, the signal "Generation Time" (in CAM and DENM) is expressed with units and the CAM signal "Vehicle Type" is interpreted, e.g. as a "Car" or "Motorcycle". Similar examples are the DENM signal "EventPosition" (with latitude and longitude, i.e. values with units) and the signal "Cause Code" (interpreted).

Visualization of the Vehicle Signal on a Geographic Map

Unfortunately, even interpreted representation of message contents with filtering is often inadequate due to the number of traffic participants and the complexity of the communication. Important data must be recognized immediately, even if it is not obvious from the set of interpreted information.

The scenario described above illustrates how the relevance of a Car2x signal for the receiving vehicle can only be determined in the context of other traffic participants in its relevant environment and can therefore only be validated in this context. For validation, the geo-positions and heading vectors of the participating vehicles must be taken into consideration. A map representation is recommended, which can clarify the relevance **(Figure 3)** in practical driving tests, e.g. at the test site in Helmond, Netherlands [3], and has already proven its value in integration support at the ITS World Congress [4] in Vienna. The evaluation is greatly reduced for the developer if identifying supplemental information is assigned by the senders, such as make, vehicle type, model or license plate. Intuitive symbols and color codes provide an easy to understand overview of the traffic situation. For example, mobile senders are depicted on the map in **Figure 3** as arrows, waypoints as flags and RSUs as circles. In contrast to the pure protocol representation of **Figure 4**, problems with incorrectly sent driving directions ("headings") can be detected immediately here. In addition, it is possible to show topographic information (TOPO) on the map as well as hazard information (DENM), which also plays a role in many scenarios.

Immediate Availability and Quick Reconfiguration

In practice, it is very helpful if the test system can immediately show the entire data traffic without a long setup time. Here, knowledge of the underlying data model is necessary, which is usually defined by ASN.1 (Abstract Syntax Notation). Nonetheless, ASN.1 cannot represent any networks, and it lacks the desired ability to manage signals with physical units. Therefore, the analysis and development tool must permit easy parameterization of this information that is supplemental to the ASN.1 description. The ASN.1 file, on the other hand, should ideally be automatically importable. This ensures, for example, that if communication rules change due to updates, the new signal values are immediately available in the tool without having to perform another step such as recompilation.

Stimulation and Simulation

Even with functioning prototypes, there is often a wish to actively participate in the communication, e.g. to send individual CAM, DENM, SPaT or TOPO messages correctly or as corrupted messages. This lets the Car2x developer to test first prototypes by targeted stimulation very efficiently. However, the development tool must be able to send messages conformant to the protocol and the air interface. On test drives, it is helpful to show RSUs or other vehicles that do not exist in real form with the map representation introduced previously. Then the development tool can assume the role of individual traffic participants on the test drive, or even all participants, and can also simulate their communication over the air interface. On the test drive, the ITS-S is no longer able to determine whether received Car2x signals originate from real sources or from the simulation. This assumes that separate software models can be saved separately for all traffic participants and that they can then be individually activated and associated with the map display.

Compatibility to Development Strategy for Previous Bus Systems

Today's vehicle networks are based on CAN, FlexRay, LIN, MOST and most recently IP (Internet Protocol) as well, e.g. in the form of BroadR-Reach technology [5]. The method of remaining bus simulation is typically used to develop individual ECUs. It makes it possible to develop ECUs in parallel and independent of one another. The network hardware that is relevant to the ECU under test, but is not yet available, is simulated in software by the development tool. Since the ITS-S is generally also a participant in one of the

Figure 3: On the map, vehicles are depicted by direction arrows and RSUs by circles. The hazard information transmitted in the DENM (e.g. OW – Obstacle Warning) and the waypoints that lead to this hazard point are depicted as well as the dissemination range of the message for this hazard.

Figure 4: Protocol representation of Car2x information in the CANoe and CANalyzer trace window.

above named vehicle buses, remaining bus simulation is also a useful method here.

Development Tools for Car2x Communication

What are the implications of the necessary Car2x extensions for a development and validation tool for production implementation?

The key to a solution is to combine the approaches described above with the usual practice-proven methods of conventional network development in the automotive industry. CANoe and CANalyzer have thoroughly proven their capabilities as multibus tools for developing onboard networks based on CAN, LIN, FlexRay, MOST and Ethernet. Option "Car2x" extends these tools for the development of convenience and driver assistance functions. This involves extending the simulation setup shown in **Figure 5** by adding the air interface. If necessary, the test system can substitute for the entire environment of the ITS-S and can both send and receive. In this approach, the above named requirements of protocol analysis, quick reconfiguration and visualization are already considered in a map view.

The WLAN Packet Builder with its intuitive user interface can be used to intentionally send faulty information for validation purposes. It makes it easy to create and send out either correct or corrupted pWLAN packets for test purposes. For more complex simulations of traffic scenarios with vehicles and the infrastructure, the Car2x developer uses specific function libraries prepared in CAPL or as a DLL.

CANalyzer .Car2x covers the most important requirements of a Car2x development tool such as protocol analysis, support of the air interface and stimulation. Visualization and quick reconfiguration capabilities also increase the usability of the development tool substantially. In addition, CANoe .Car2x extends the tool's range of use to include many different simulations and test functions.

Outlook

Future driver assistance systems based on ITS-S will have to incorporate additional vehicle dynamic data that supplements the Car2x communication, and this supplemental data is available on CAN, FlexRay or IP networks in the vehicle. It is therefore increasingly important for the development system to be able to represent both the Car2x communication and communication on conventional bus systems with high timestamp accuracy and over multiple channels. CANoe .Car2x and CANalyzer .Car2x are already equipped for these tasks today.

Figure 5: ITS-S test system: If necessary, the test system can substitute for the entire environment of the ITS-Station and can both send and receive. The usual simulation setup is extended by adding the air interface. As in a realistic situation, the ITS-S can communicate via the air interface and local bus systems.

The map window (**Figure 3)** makes an important contribution to the analysis. The next development step might be to use this map to define scenarios and constraints on the behavior of the simulated traffic participants. A radio adapter already installed in the vehicle could be used as the 802.11p WLAN interface hardware for communication between vehicles or between a vehicle and the infrastructure. It could be used together with the vehicle application or exclusively as a measurement interface. This is a pragmatic and flexible solution for many application cases. However, the measurement precision of this radio adapter might be inadequate for some tasks. Consequently, there is some debate over whether even more precise, further advanced measurement hardware might be made available in the future.

Translation of a German publication in Elektronik automotive, 12/2012

Literature:

[1] ETSI, Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions, ETSI TR 102 638 V1.1.1 (2009-06)
[2] CAR 2 CAR Communication Consortium, Related Projects, www.car-to-car.org/index.php?id=6&L=oksjfr
[3] Making cooperative systems cooperate, DRIVE C2X @ DITCM Helmond, NL, www.drive-c2x.eu/news-item/items/drive-c2x-ditcm-making-cooperative-systems-cooperate
[4] ITS World Congress, Vienna, http://2012.itsworldcongress.com
[5] Schaal, H.-W.: Ethernet and IP in motor vehicles, Elektronik automotive. Issue 4/2012, pp. 38ff.

Links:

Vector solutions for Car2x:
www.vector.com/vi_car2x_solutions_en.html
Product information CANoe.Car2x:
www.vector.com/vi_canoe_car2x_en.html

Hans-Werner Schaal (Graduate Engineer)
is Business Development Manager in the areas of Automotive Ethernet, Car2x and open CAN protocols such as J1939 and ISO11783 at Vector Informatik GmbH.

Thomas Löffler (Graduate Engineer)
is Senior Software Development Engineer at Vector Informatik GmbH working in the area of Car2x. His areas of focus are product definition and project management of customer projects.
Mr. Löffler also represents Vector on a number of Car2x standardization committees.

K-Line: Flexible Solutions for a Classic protocol

From Precise Monitoring to Data Manipulation for Generic Byte Protocols

In the past, the K-Line diagnostic protocol was the standard for diagnostic tasks in various vehicles, and it is still used widely used today. The age of this interface has not made it obsolete in today's diagnostics, development projects and The K-Line diagnostic protocol no longer plays a substantial role in new developments, because systems such as CAN and Ethernet have long taken over diagnostic tasks once performed by the K-Line. Nonetheless, automotive OEMs, suppliers and service shops worldwide cannot overlook the fact that many vehicles and ECUs still use K-Line technology, and this will remain the situation for some time. ECUs with a K-Line interface are still used in passenger cars, the truck sector and in motorcycles.

Those Presumed Dead Live Longer

Millions of passenger cars and motorcycles with K-Line technology are still driving on the roads, especially in markets such as China, India and South Asia. They are generally vehicles whose level of technology is outdated by around 10 to 15 years. Many European vehicle developments of that time were and are still being built in Asia under license, although their production ceased many years ago here. It is still the usual practice – especially in cases of smaller production volumes – to continue to use proven ECU developments in subsequent or related product lines, and this too has extended the life of the K-Line.

Serial UART Diagnostic Protocol with Bus Characteristic

The K-Line is a diagnostic protocol that conforms to the ISO 14230 standard. Like the standard RS232 serial interface, it is based on the technology of typical UART (Universal Asynchronous Receiver Transmitter) circuits. In asynchronous transmission, the sender and receiver use start and stop bits for synchronization purposes. This means that the system does not need a supplemental clock line, and a single-wire line suffices. In contrast to RS232, the K-Line – like a bus system – enables communication with different ECUs by addressing them. The standard transmission rate is 10,400 baud, and speeds up to 115.2 kbaud are used for such purposes as programming of flash memories.

The K-Line is suitable for both on-board and off-board diagnostics, and it offers two special initialization patterns: Fast-Init is based on a 10,400 baud standard, and it sends a wake-up pattern. There is also what is known as the 5-Baud Init pattern, in which the system sends an address byte at five baud, and the receiver detects this slow transmission rate. Also characteristic of the K-Line are special Key Bytes that are used to identify header formats and timing parameters.

One important task of automotive OEMs in the after-sales market is to support the service of all K-Line vehicles worldwide by providing service shops with suitable K-Line testers. In ECU development with K-Line, new functions are provided that need to be tested. Therefore, manufacturers and suppliers need powerful hardware and software tools that support the K-Line protocol for K-Line test equipment and ECUs.

Figure 1: Different K-Line interfaces: from single-channel USB interface to HiL module

More Stringent Requirements for Test Hardware

A basic prerequisite for any diagnostic or test process is a suitable interface hardware, which produces the connection between the diagnostic PC and the device under test. It is possible to use a PC's conventional UART/RS232 interface to test K-Line devices, but this method quickly encounters limitations. It lacks the advanced properties that are needed to check for conformity and to verify correct functionality. This also requires knowledge of how close the DUT is to operating at its specified limits, or expressed differently, the size of its functional reserves.

In contrast to RS232 solutions, efficient K-Line interfaces enable precise acquisition of communication timing. Both sent and received K-Line frames are provided with exact time stamps. They also offer automatic detection of baud rates – including fast initializations and 5-baud initializations – and it is also possible to manipulate K-Line timing and data and to send raw byte streams. These interfaces can be connected to any PC via USB, and they work together with software tools ideally, e.g. over a specialized K-Line API, which enables easy access to all hardware functions in test scripts.

Scalable K-Line Solutions

Vector offers a product line-up of K-Line components that are tuned to one another for the purpose of testing and simulating K-Line-developments; these components con-

Figure 2: K-Line test and simulation environment

sist of high-quality interface hardware and high-performance software tools. The solutions cover all conceivable requirements and are flexibly scalable – from a single-channel K-Line monitoring tool to solutions that enable simulation of K-Line diagnostic testers and ECUs, and finally large HIL systems. The latter are characterized by such aspects as real-time properties, and they can simulate multi-channel ECU environments for test runs, in which other bus systems such as CAN, LIN and FlexRay can be integrated along with K-Line. Vector can supply various types of interfaces for connecting to K-Line – via a USB interface or PCI bus. They include the VN1600 and VN8900 interface products as well as plug-in cards such as the VN7570 and the VT6204 for the VT System **(Figure 1)**. The 7269 LIN transceiver, which offers optimal K-Line support, handles transmission on the physical level.

Support of Proprietary K-Line Variants and Byte Protocols

CANoe and CANalyzer are two alternative software tools that are available from Vector. While CANoe represents the universal solution for (automated) tests and simulations, the focus of CANalyzer is on analysis and monitoring tasks **(Figure 2)**. These tools permit access to all K-Line parameters and settings. Testing personnel can conduct the tests, measurements and injection of errors on different levels: on the diagnostic and communication levels and – a unique capability – on the byte level. This makes the tools usable for proprietary K-Line variants that deviate from the standard as well as generic serial byte protocols. Trace and analysis windows display timing, baud rates, header bytes, useful data, inter-byte and inter-frame spaces with high precision **(Figure 3)**. Other windows permit interactive sending of K-Line frames. The application programming language, CAPL, can be used to send raw frames and inject errors. Simulations can also be created with CAPL in conjunction with a special K-Line API. Test modules then produce the automatic test sequences and generate reports.

Summary

High-performance and modern tools are also available for the K-Line protocol, which has certainly aged over the years, but is still used for such purposes as maintenance of diagnostic testers and ECUs. They not only give automotive OEMs and suppliers qualified tests at a high level of quality; they also enable troublefree advanced development and reuse of existing K-Line components.

Reprint of the online publication in Automotive EE Times Europe, May 2015

Image rights:
Vector Informatik GmbH

Peter Decker
has worked at Vector since 2002 where he is currently Product Manager in the Networks and Distributed Systems product line.

Figure 3: K-Line analysis on different communication levels

Serial Bus Systems in the Automobile

MOST for Transmission of Multimedia Data

A premium class car is growing to resemble a mobile office. In response to customer demand, increasing numbers of entertainment and information media that are making their way into automobiles. The most significant challenges in this area are, first, to keep wiring expense as low as possible, and second, to fully satisfy the heightened functional requirements of an infotainment system in the car. As a result, the MOST (Media Oriented System Transport) bus system is now used to transmit audio and video signals in approx. 50 model series.

Electronics is responsible for a large number of innovative safety and convenience functions in automotive technology. Experts predict that in just a few years electronics will represent a share of up to 30 percent of vehicle value, and the worldwide market for electronics in cars will grow by approx. 6 percent annually to 230 billion euros by the year 2015. The automotive industry is forecast to exhibit rapid growth rates, above all in the infotainment area, given the continually increasing vehicle-kilometers on Germany's roads (according to DIW [1] approx. 700 billion). The average citizen spends about 270 hours in a car annually, whether it is on the way to work, shopping or vacation.

Over the course of time, the car radio was supplemented by the CD and MP3 player. This came to include CD changers and navigation devices, and finally display screens also made their way into cars for replaying DVD and video films. Moreover, hands-free Bluetooth units with integrated microphones and iPod control are gradually turning the cockpit a multimedia center, in which all of the play lists and directories of a digital MP3 player can be displayed and started directly on the in-vehicle display.

The already extensive wiring cost and effort are increasing due to continual growth in networking of continually higher performance infotainment devices of dimensions that can hardly be managed any longer. Fortunately, some automotive OEMs recognized the advantages that bus networking would also offer in this area early on. In the mid-1990s, BMW and Daimler began to develop a uniform communication technology for serial transmission of audio and video signals in the vehicle based on the D2B bus (Digital Data Bus) developed by Matsushita and Philips.

MOST Cooperation

In 1998, BMW, Daimler, Harman/Becker and SMSC (formerly OASIS SiliconSystems) founded the MOST Cooperation [2]. Since that time, MOST has established itself as a de-facto standard for the transmission of multimedia data in the vehicle – the MOST Cooperation is made up of 15 international automotive OEMs and more than 70 device producers. The user organization laid the foundation for success of the technology by defining an extensive specification. Version 2.5 of the MOST specification has been in existence since October 2006. It is organized into the areas of Application, Network and Hardware.

The "Application" area describes a logical device model based primarily on object-oriented methods, with the purpose of transparent modeling and control of distributed infotainment systems. Furthermore, it defines a hierarchical communication model as well as services for managing an infotainment system. The "Network section" describes the MOST Network Interface Controller and its services, network management, and handling of data transport in a MOST system. The "Hardware section" deals with aspects of the hardware structure of a MOST device.

Figure 1: Structure of a MOST device that among other things hosts the "Audio Disk Player" functional block. For system management, the Net Block is mandatory for each MOST device, and system functions are mandatory for each function block.

Functional Modeling

A MOST device is subdivided into a functional level and a network level (MOST Network Interface). On the functional level, infotainment functionalities are embodied in so-called function blocks. Each function block, e.g. the Audio Disk Player, provides the MOST network with a dedicated set of functions, e.g. "Track position", that can be accessed by operation types such as "Set" for setting a track or "Set-Get" for setting and reading a track **(Figure 1)**.

Functional addresses (FBlockID, FktID) are assigned to both the function blocks and to the functions provided by a function block. They can be taken from the so-called "Function Catalog", as can the identifiers of the operation types. For example, the "Audio Disk Player" FBlock has FBlockID=0x31 and the "Track Position" function has FktID=0x202. The separation of function and network and functional modeling make it possible to implement a functional communication model that is fully independent of physical components (MOST devices). Therefore, it does not matter which of the MOST devices is used to contain a specific function.

Hierarchical Communication Model

MOST systems are patterned on a three-stage hierarchical control philosophy based on the "Master-Slave principle" **(Figure 2)**. Placed at the uppermost hierarchical level is the HMI (Human Machine Interface), an exposed controller that provides the user with overall functionality. On the middle hierarchical level are the usual controllers. They cover part of the system functionality, and they share their partial system knowledge with the HMI as the "System Master".

The lowermost hierarchical level is made up of the system slaves, whose functions are used by one or more controllers. They are not equipped with any system knowledge, and this substantially enhances their flexibility with regard to configuration. It is easy to add system slaves or remove them from a MOST system. MOST commands are used for control communication. Their core components are the FBlockID, FktID, Operation Type and up to 65535 useful bytes.

Figure 2: Hierarchy of a MOST system

Figure 3: Difference between NIC and INIC architectures in a MOST device

System Management

The Application section defines higher-level function blocks and functions for system management. System functions include the "FktIDs" function (FktID=0x000) that is used to query the functions supported by a function block, for example. The "Notification" system function (FktID=0x001), on the other hand, enables creation of the "notification matrix" for a function block. Emerging from the "notification matrix" is information on which MOST device should be notified if a certain property of a function block has changed. This mechanism prevents an unnecessary increase in bus load in the MOST system.

To query its function blocks and addresses, each MOST device has the "Net Block" (system) function block with FBlockID=0x01. The function blocks can learn about the function blocks implemented on a MOST device using the FBlockIDs function (FktID=0x000). The FktIDs 0x002, 0x003 and 0x004 are used to find the physical address, logical address and group address of a MOST device.

The Network Master plays an important role in the management of a MOST system. It is responsible for the system start and management of the "Central Registry". This registry contains the logical addresses of the MOST devices implemented in a MOST system and the addresses of function blocks contained in the MOST devices.

MOST Network Interface

The MOST Network Interface **(Figure 3)** ensures that the function blocks housed on the various MOST devices are capable of real communication with one another. The MOST System Services (Low Level System and MOST Network Services) provide the communication functionalities needed to transport all multimedia relevant data (time-continuous bit streams, packet data and control data). Low Level System Services (Layer 2 services) are implemented in hardware (Network Interface Controller – NIC) and are placed over the Physical Layer.

MOST Network Services, which encompass the Transport Layer in the form of Basic Layer System Services and higher management in the form of an application socket, are housed on an external Host Controller (EHC) and control the NIC. It must be ensured that the EHC can serve the time-critical parts of the Network Interface. Over time, with the progressive development of MOST technology from MOST 25 to MOST 50 and MOST 150, this architecture has now encountered its limits.

In new developments, INIC (Intelligent Network Interface Controller) replaces NIC. While INIC assumes control of execution of time-critical portions of the network driver of the EHC, just a relatively small part of the network driver still runs on the EHC, which essentially represents a socket for the application. The INIC architecture thereby relieves the load of the EHC. For control, the INIC provides the EHC or MOST API (MOST Network Services) with a functional interface, the so-called INIC-API. The functions of the INIC are encapsulated in a function block (FBlock INIC).

MOST Networking

MOST technology enables transmission of continuous bit streams (bit streaming) without buffering or unnecessary overhead. This involves having a specially designated MOST device (Timing Master) feed the MOST frame **(Figure 4)** at a fixed frequency (44.1 KHz or 48 KHz) into the transmission medium, which is typically optical.

In a MOST25 system, the MOST frame provides 60 streaming channels at 8 bits (or 15 quadlets of 4 bytes each) for transmission of continuous bit streams (source data area). The bandwidth of a streaming channel yields either 352.8 KBit/s (44.1 KHz) or 384 KBit/s (48 KHz).

Since the MOST devices are physically interconnected into a ring, each MOST frame must pass through every MOST device at the frequency prescribed by the Timing Master. As soon as the relevant communication partners (data source and sink) have connected to the same streaming channel, bit streaming begins **(Figure 5)**.

Connection or disconnection is usually made by a query by the function block "Connection Master – CM" (FblockID=0x03). For this purpose, the CM provides the two functions "BuildSyncConnection" and "RemoveSyncConnection". In the framework of building a connection, the CM requests that the relevant data source, e.g. the TV tuner, have the suitable number of streaming channels allocated by the Timing Master. That is because the Timing Master is responsible for management of the "channel resource allocation table". The CM passes the addresses of the allocated streaming channels to the data sink, e.g. to the display, so that it can connect to the streaming channels. Finally, the CM updates the "sync connection table", which it uses to manage all synchronous connections. Disconnection is performed according to the same scheme.

To enable transmission of data packets, the user has the option of reducing the number of streaming channels by up to 24 (six quadlets) using the "Boundary Descriptor". All those streaming channels that are not reserved for bit streaming, are combined to form the packet channel. While a maximum transmission rate of up to 12.7 MBit/s is possible at a frequency of 44.1 KHz, a maximum rate of up to 13.8 MBit/s is attained at 48 KHz. The boundary descriptor is managed by the Network Master function block (FBlockID=0x02). It can be set via the "Boundary" function (FktId=0xA03).

A Layer 2 protocol is used to transmit data packets. The frame comprises the arbitration field, source and target address, data length code, data field (either 48 or 1014 byte) and data protection. A token circulating in the ring regulates bus access. The MOST device that takes the token from the ring may access the packet channel.

Finally, the MOST system must transmit the MOST commands needed for management and control. Control messages **(Figure 6)** are used here, which are transmitted on the control channel (2 bytes). Therefore, 16 MOST frames (MOST block) are required to transmit a control message. The bandwidth at 44.1 KHz is 705.6 KBit/s, and at 48 KHz it is 768 KBit/s. Transmission of control messages is also based on a Layer 2 protocol. Bus access is implemented by the CSMA method (Carrier Sense Multiple Access).

Figure 4: Layout of the MOST frame: Sent in administrative byte 0 are synchronization information and the boundary descriptor, and in administrative byte 63 the status bits and a parity bit for protection of the MOST frame.

Figure 5: Principle of bit streaming: The Timing Master transmits MOST frames at a frequency of 48 KHz. 40 streaming channels (10 quadlets) are available for allocation, each operating at 384 KBit/s (boundary descriptor = 0xA). The packet channel (20 bytes) provides a bandwidth of 7.68 MBit/s for the transmission of data packets.

Figure 6: Control message. A MOST block is required to transmit a control message. The control message is composed of: arbitration (a), target address (b), source address (c), message type (d), data field (e), data protection (f), acknowledgment (g), and reservation (h).

Background knowledge on signal transmission in a MOST system via POF

When a light beam passes from one transparent medium to another, it is refracted. The greater the angle of incidence, the greater the refraction. The medium in which the light beam forms a smaller angle with the primary axis is the optically denser medium. In the transition from the optically denser to the optically less dense medium, the beam is refracted away from the primary axis. The angle of refraction a can be calculated if the so-called indices of refraction n of the two media are known (Snellius Law). If the light beam exceeds an incidence angle a0 in the transition from the optically denser medium to the optically less dense medium, then total reflection occurs.

This property makes it possible to transport light in an optical conductor. In the MOST system, polymer fibers are usually implemented for optical signal transmission, where a core of PMMA (polymethylmethacrylate) is encased in a thin sheath of fluorinated acrylate. PMMA has a larger refractive index than the fluorinated polymer. If the angle of the incident light beam is greater than the limit angle, then the light is conducted in the core due to total reflection.

The smallest attenuations for transmission of light in a so-called step-index PMMA fiber are obtained at 520 nm (green), 560 nm (yellow) and 650 nm (red). Red LEDs are generally used (attenuation 0.14 dB/m), since they are very inexpensive.

Figure 7: Background knowledge on optical signal transmission in a MOST system

Physical Layer

Today, optical conductors of polymer fibers (POF – polymer optical fiber) are the state-of-the-art technology for transmitting audio and video signals in the MOST system **(Figure 7)**. Overall, the technical properties of polymer fibers are far superior to those of electrical transmission media. Especially noteworthy are its excellent electromagnetic immunity and relatively high signal transmission rate of up to 500 MBit/s. Furthermore, the combination of POF, a red light-emitting diode as the light source (wavelength 650 nm) and a silicon PIN photodiode as the receiver represents a very economical and comparatively simple and manageable form of optical signal transmission.

MOST 150, which follows MOST 50, is a MOST system that is current ready to start. It is based on this sender and receiver technology and offers the user a transmission rate of 150 MBit/s. It can therefore handle the relatively short paths in the car of up to 20 meters can without any problems.

Development, Testing and Analysis of MOST Systems

Vector Informatik GmbH has been an associate member of the MOST Cooperation since 1999. Besides its extensive activities in the area of serial bus systems such as CAN, FlexRay and LIN, the Stuttgart-based networking specialist has been supporting the development and analysis of infotainment solutions in the automobile since the year 2000. It offers a comprehensive product lineup of analysis, development and test tools for applications such as high-end audio systems, multimedia streaming, telephone and navigation. Hardware interfaces for bus access, a multibus data logger as well as training courses and engineering and development services round out its offering [3]. The Vector Academy [4] supplies the necessary basic knowledge related to ECU communication in the automobile in the framework of seminars on CAN, LIN, FlexRay and MOST.

Literature and links:

[1] www.diw.de
[2] www.mostcooperation.com
[3] www.vector-group.net/most/en
[4] www.vector-academy.com

Eugen Mayer (Graduate engineer; technical teaching certificate); after vocational training as a communication electronics technician, he studied electrical engineering and technical education at the Technical College of Ravensburg/Weingarten and the University of Stuttgart. Since 1999 he has been employed at Vector Informatik where he works as a Senior Trainer.

VECTOR

PREEvision
Technical Article

Controlling Complexity in the Vehicle Wiring System: Model-based Wiring Harness Development

Handwritten corrections on printouts, giant spreadsheets, and countless tool changeovers: these and similar approaches are used in an effort to keep a handle on the development of a vehicle wiring system. More efficient approaches are needed to control the complexity of modern electrical distribution systems in the long run. The model-based development methods already in successful use in many industrial sectors provide the needed tool for this.

The Earth is flat and a wiring harness is a drawing! We all know what became of the notion of a flat Earth. And we also know that uncritical adherence to established beliefs, behaviors, and habits slows down progress – or stops it altogether. From time to time it is good to ask: Are there new ways I can try? Can I apply knowledge and methods from other industry sectors to solve new challenges?

For vehicle wiring systems development, challenges include current multi-voltage electrical distribution systems, autonomous driving, electric drives, and increasing networking. These pose a challenge to many established development processes and bring them to their limit. Model-based development is an efficient and future-proof method for keeping the complexity of the coming electrical architecture generations under control over the long term.

Development Process with Model-based Tools

Digital modeling of subareas to whole vehicles has been a common practice in automotive mechanical engineering for many years. A digital model is built up from component parts with modern CAD systems. These component parts are linked with complex overall systems. If a change creates a conflict in an adjacent area, the tool points this out to the developer as a matter of course. The developer can directly intervene and eliminate the conflict immediately. This matter of course is also possible in the field of electrics/electronics (E/E) with model-based development tools such as PREEvision. The model-based approach promotes consistent development from a technical and timing point of view. The basis for technical consistency is an overarching E/E data model: It includes the system requirements, architecture and design levels for software, network, and components as well as the wiring harness and the geometry. All object dependencies between these levels and elements are described in the data model. Time sequences and relationships are controlled by a versioning process and a central data storage **(Figure 1)**.

Figure 1: Consistent model-based tools, reuse concepts and variant management reduce the complexity of development processes in spite of increasing vehicle functionality (schematic diagram).

Modeling of the Wiring Harness

The nonuniform tool landscapes often used in development departments are an impediment to achieving the desired technical consistency. Different tools are used for the component design, electronics logic, wiring harness engineering and variant management. Added to that is the manual transfer of data between development tools or an electronic exchange with time-consuming, error-prone consolidation. If the communication and coordination between separate team and department structures is not ideal, this quickly gives rise to misunderstandings and confusion. This leads to technical errors.

Another obstacle from the classic electrical architecture development is that the architecture is often still developed completely decoupled from the specific wiring harness. In the best case, the architecture specifies a loose framework for the vehicle project development. There is little or no interaction and exchange or common understanding; possible improvements based on feedback and "cross-fertilization" thus go unnoticed. There are many arguments in favor of improving the technical consistency in vehicle wiring system development.

The first step in the model-based development process is a uniform data model for all parties involved. This forms the basis of teamwork by all users, teams, and whole departments from the required technical areas of expertise. This standardized integrated model uses a defined description to ensure a common understanding for tasks and solutions. The consistency is already checked during acquisition of information. The data linked multiple times in the model permit efficient engineering of the wiring harness not only for an individual but also across area boundaries. When time-consuming and error-prone tool changeovers are eliminated because all development steps take place in one environment, this additionally contributes to technical consistency.

Efficiency Through Feedback Loops

An example of technical consistency from the architectural design to the detailed wiring harness: The architects group models the overarching design of the electrical architecture for the entire vehicle project. The wiring harness development does not start from zero. Rather, the design created in the model is used as a starting basis for development of the wiring harnesses of specific vehicles. The network archi-

Figure 2: Technical consistency from the network architecture, electronics logic and wiring harness to the geometry. All drawings are based on the same model data (exists only once) and represent the desired sub-aspects.

tecture is the binding specification from which the electronics logic is derived and developed. Here the potentials are defined and the power supply of the components and their fusing is supplemented. The result is a wiring harness with all wires, cables, connectors, and pins. The assignment of the wiring harness components to a vehicle geometry and the automatic routing of the wires give rise to the cable harness with isolation points as well as cable protection and mounting points. As a result of continuous further development and refinement across the individual areas of responsibility, time- and resource-saving development is made possible.

The efficiency gain arises not just from one direction. The tight integration means that findings from wiring harness development are directly taken into consideration again in the underlying architecture, thereby increasing the quality of the design each time. The model-based development and data management also facilitates the technical consistency check. After all, a variety of semantic framework conditions in the technical data model do not permit faulty modeling in the first place. Moreover, further rules and checks can be easily defined and executed. Through these rules and checks, the user will receive direct warnings of inconsistencies during development of the wiring system. In this way, technical errors will be discovered early and eliminated **(Figure 2)**.

Central Data Storage and Time Consistency

In order to work on an integrated development model together with many other people, time consistency must be ensured in addition to technical consistency. The backbone of such a solution is the central data storage in a database, which results in a uniform and, if necessary, globally available database. This database ideally contains a finely subdivided versioning of the development artifacts, which can be reverted to at any time if necessary. Model-based development tools like PREEvision meet this requirement, for example, by managing artifacts of a development line in revisions or by building up parallel development lines using branches. Intermediate versions can always be made independent of the release of a version in the central data backbone. They are then also available to other employees at all times.

Nowadays, processes, work steps, and process-oriented working methods are also digitally supported during electrical architecture development. Here, a tool changeover should be avoided because information is thereby separated from the technical data and exported from the model – the time consistency would no longer be ensured. For this reason, model-based development tools support the management and release of data using life cycles in the technical model. With these life cycles, the current status can always be directly recognized and there is no need to look for this in another tool. This impedes errors such as the use of incomplete or faulty releases and versions.

Figure 3: Time consistency through central data management and simultaneous teamwork by many users. The data are managed in versions and branches and controlled with life cycles.

Change management also plays an important role in time consistency, since a wiring harness undergoes many changes on the way from development to production. During this life cycle, all changes must be recorded, managed, compared, evaluated, and finally implemented on the wiring harness. Model-based tools can also provide effective support to the vehicle wiring system developer here. As a precondition, they must include an integrated ticket-based change management process that is located in the same system as the technical data.

Besides enabling the efficient management and evaluation of changes in one tool, the creation of "change sets" is also possible. Change tickets are linked directly to the relevant technical data. Each change is ideally documented and completely traceable. A model-based system also offers the advantage of maintaining changes centrally in the model data. Because all drawings, bills of materials, and reports are based on the model data, they are always current. The tedious manual entry of corrections in many different places is eliminated **(Figure 3)**.

Required and Unwanted Complexity

The concentration of functions important for wiring harness development in one tool minimizes unwanted complexity from redundancy and unintended variance. Besides the technical complexity, there are other aspects that argue for the use of a comprehensive model-based development tool. If different tools are used in the development process, training and expertise is needed for handling each of them. Checks and corrections of transferred data at tool changeovers make development processes lengthy and-prone to errors. And, having a variety of data formats, data storage locations, and information contents often leads to different worlds of thought of employees – not a good precondition for smooth communication and collaboration. For efficient vehicle wiring system development, it is necessary to reduce this unwanted process-related complexity to a minimum.

On the other hand, there is the desire for effective, error-free control of the necessary complexity. This necessarily means the user-chosen complexity, such as the increasing functionality resulting from more and more electronic systems in automobiles and their networking. These require an increasing number of essential components that must be supplied and controlled. The same goes for the finely graduated, individual configuration options of vehicles, which yield a wide range of variants. A tool for developing wiring harnesses for current and future vehicle generations must therefore also provide an efficient variant management process and represent and manage the variety of resulting wiring harnesses.

Figure 4: Product line approach for managing model-based data: Frequently used units are contained in libraries and are expanded or further developed for a product line. Cable harnesses for specific vehicle variants are then derived from the product line.

But even that is not enough – the data volumes that arise when developing wiring harnesses for vehicles, vehicle families, and whole platforms must also be structured and controlled. The product line approach has proven to be useful here. For managing model-based data in vehicle wiring system development, frequently used units, such as connectors, wires, and add-on parts as well as components or whole subsolutions, are provided in libraries. Individual vehicle projects help themselves from this central library; they reuse component parts or larger units without having to rebuild them completely again. Error-prone and redundant developments are thus avoided and unwanted complexity is further reduced **(Figure 4)**.

Future Wiring System Generations

The model-based approach is an alternative that has already proven effective in other application areas and opens up new and future-proof opportunities for vehicle wiring system development. The use of model-based development tools such as PREEvision from Vector Informatik in the E/E area is a progressive step that can be used to master development projects in the automotive industry today and in the future. It allows competitive solutions and ever shorter development cycles.

However, immediate replacement of all established development processes overnight is not possible. Before that can happen, organization-wide rethinking is necessary that launches a change process and allows new, model-based processes on an incremental basis. Wiring harness developers, in particular, will benefit from a powerful tool they can use to face future wiring system generations with ease. After all, they know that keeping a handle on the wiring harness of the future requires more than a drawing.

Dipl.-Ing. Tobias Bitzer
Tobias Bitzer studied aerospace engineering with a concentration on control technology at the University of Stuttgart. He has worked in product management at Vector Informatik since 2007, where he initially concentrated on the issue of collaboration. Since 2013 he has been responsible for the hardware architecture, electrical design and vehicle network design in PREEvision.

Translation of a German publication in Elektronik automotive, issue S4/2016 "Bordnetz"

Image rights: Vector Informatik GmbH

PREEvision
Technical Article

Eye On the Whole System: Why Consistent Implementation of the AUTOSAR System View is Worth it

Started thirteen years ago, the AUTOSAR standards enables efficient electric/electronic development today. Besides continuous additions in recent years, the systems thinking remains a mainstay of the standard. Rather than focusing on the individual ECU or on a communication bus, AUTOSAR always looks at the whole system. This system view is playing a more and more important role in the digitization of the automotive industry and, together with the "Adaptive Platform", is paving the way for the next generation of vehicle electronic systems.

AUTOSAR describes the electric/electronic architecture of complete vehicles with all configuration details: software, topology, bus communication and ECU-specific parameters. For this purpose, the methodology universally uses process blocks, which are largely organized into activities and work products **(Figure 1)**. By connecting these process blocks, the methodology indicates the way from the creation of the system configuration to the generation of an executable file for a particular ECU. The system-wide view is firmly anchored in the AUTOSAR standard and shapes the methodology throughout.

From Abstraction to Software Component

The methodology starts with the design activity "Develop an Abstract System Description". Here, the functional architecture of a vehicle is described in detail without going into the technical details and the implementation. The result is an abstract system that reflects the formal idea of the overall vehicle system as a network of logical functions and their interfaces and connections. The benefits for further development are obvious: an abstract description makes system dependencies and possible optimizations much easier to analyze. Through the linking of logical functions and all system components, the design always remains transparent. Logic errors or redundancies in functions are detected early and avoided. The creation of the "Abstract System Description", however, is included only as an optional step in the AUTOSAR methodology and only a handful development tools can implement it truly universally up to now. The additional effort for this process step is still worth it because of the enormous efficiency gains. Architects and designers are given an ideal work basis for subsequent distribution and implementation of functions among specific ECUs **(Figure 2)**.

Whether or not the abstract system description comes first, the functional design is implemented in the AUTOSAR software system architecture with the concept of the "Virtual Functional Bus" (VFB). The activity "Develop a VFB System Description" defines the software functions in the vehicle more precisely. The resulting work product is the "Overall VFB System", a system of software components

Figure 1: AUTOSAR overall workflow with activity and work product process blocks.

connected by ports that communicate with one another via defined interfaces. It makes no difference whether the information exchange subsequently takes place later via a physical bus or within an ECU – the VFB abstracts this communication. The individual software components within the VFB system are described as a "Delivered Atomic Software Component" – or frequently also as: Software Component Description **(Figure 3)**.

If the overall system is not continuously kept in mind during this phase, incompatible interfaces are often observed later during integration. Here, for example, multiple interfaces were then accidentally defined for the same software components – with the same logical meaning but with different form and technical details. If these errors are not identified until later in the development process, expensive correction measures are the result.

Modern tools for electric/electronic development, such as PREEvision, provide an effective means of preventing this. Thanks to an integrated connection to the "Abstract System Description" and the implementation of the AUTOSAR-VFB concept in the tool, inconsistencies and redundancies in the design are detected at an early stage and avoided. Ideally, all components, interfaces, and data types are also created only once and kept in a central software library. They are thus available across projects and are reused as required.

Topology, Design and Standardization of System Communication

With the "Design System" activity, the software components defined in the VFB system are distributed among a topology of networks and ECUs. System architects normally perform this task. They either take the topology of an existing vehicle and expand it to include new functions as needed, or they create a new topology from the ground up.

In both cases, they define the bus technology (CAN, CAN-FD, LIN, FlexRay or Ethernet) and hardware properties, assign the software components to the ECUs, and analyze the expected communication load within the system. The objective is optimal load distribution on the bus systems and ECUs **(Figure 4)**.

The communication on the individual bus systems is described in the AUTOSAR system design by the activity "Design Communication". A major advantage of the AUTOSAR methodology over conventional formats such as LDF or DBC is that gateways can also be described without difficulty in the communication matrix. Gateways enable transparent transmission of signals and messages from one bus system to another. The route of the signal from the sender to the receiver is always traceable system-wide – independent of the utilized transmission technology **(Figure 5)**.

Figure 2: The AUTOSAR Abstract System Description in PREEvision: logic functions describe behavior and interfaces of the system and are linked with software components.

Transport protocol, network management, and subnets are additional features on the communication layer that are standardized by AUTOSAR: With the transport protocol, large data packets are segmented into smaller data packets and thus transferred over conventional bus systems. After their transfer, the data packets are reassembled by the receiver. This function is supported in AUTOSAR across gateways and is used mainly for diagnostic purposes but is also available for application data.

With "Network Management" and "Partial Networks", system-wide subnets are created that can wake up, keep awake, or shut-down a group of components in order to save energy. The resulting network management configurations can comprise multiple communication clusters. If required, other aspects are also implemented in the AUTOSAR system design:

> If large or very complex data units will be transmitted, a data transformation is defined. Data on the sender side are serialized and then deserialized on the receiver side. It is no longer necessary to map complex data types onto individual signals, so the complexity of the configuration is reduced.
> If multiple ECUs in a distributed system work together in an explicitly defined sequence, a system-wide time synchronization (Global Time Synchronization) is necessary.
> Timing constraints, events, event chains, and information from topology, software distribution, and signal mappings are used to define mandatory time requirements for the system.
> System variants of a vehicle are defined using conditions and binding times on elements of the system description.
> Each system element can be referenced by safety information or provided with an ASIL level.

Figure 3: Representation of AUTOSAR software architecture with software components, ports, and connections in PREEvision.

The result of the system design is the work product "System Description": It includes a complete vehicle design, which is exchanged between organizations and development tools in XML file format. The AUTOSAR methodology pays attention to different exchange points and recommends the appropriate scope for the respective data delivery. The "Deliverable" generated in this way is then an AUTOSAR XML file (or group of AUTOSAR XML files).

AUTOSAR – Development by the Book?

For those who want to implement the AUTOSAR system view consistently, the question remains: Is a defined development process prescribed in the AUTOSAR standard? The answer is an unequivocal "no". AUTOSAR merely describes typical blocks from which a development process can be put together. The goal of AUTOSAR is a standardized data exchange between organizations. Only the portion of the system design needed in each case is transferred at the exchange point and then used seamlessly to perform typical vehicle development tasks (Figure 6).

Figure 4: AUTOSAR cluster diagram in PREEvision: a gateway connects CAN and Ethernet communication clusters made up of ECUs and associated software components.

One Replacement Format, Maximum Flexibility

The AUTOSAR development process normally has two main phases. The system design usually takes place at the automobile manufacturer (OEM). The subsystem design – also known as the component design – is normally assigned to one or more development partners or suppliers (Tier 1). The system design is a binding specification of the OEM for implementing all vehicle functions and is broken down into functional subsystems.

A Tier 1 is responsible for development and design of a subsystem. Integration of the subsystems and the system tests take place at the OEM again. Scopes and functional details of the subsystems are defined differently from OEM to OEM, and many are even different from project to project at the same OEM. Their content is ultimately always defined in deliverables in a standardized and unambiguous way and is exchanged in this form via the AUTOSAR interface. This pertains not only to communication with a supplier but also to communication within an organization. Departments and teams exchange any parts of the system in this way, and designs are then refined or processed in a configuration or in program code.

An example of this is the extract of a "Software Component Description" from the "Overall VFB System": It is used for model-based development and simulation or in code generators. A "System Extract" can, in turn, be created from the "System Description". It contains a subsystem with one or more ECUs, the description of software components running on the ECU(s), and the communication configuration for the interface to the bus system. The "System Extract" is suitable both for internal use, such as simulations or tests of subsystems as well as for passing on to third parties.

For communication between OEM and supplier, the "ECU System Description" is often also used. It contains the software architecture in its original structure and provides a full description of an ECU with all configuration aspects. The commonly used "ECU Extract", on the other hand, describes an ECU using atomic software components. The configuration of the ECU is then derived directly from it.

All formats always remain subsets or derivations of the system description. This acts as the central and only database for all AUTOSAR process steps and exhibits so to speak the AUTOSAR system view.

A Standard that Shapes Development Processes

Originated more than 10 years ago, AUTOSAR is today a standard in the automotive industry that is significantly shaping development processes. It never remains static. On the contrary, it is always open to new additions and technical developments. Prominent examples of these include integration of IP technology and Ethernet in AUTOSAR 4.

Still, what will the future bring and is AUTOSAR ready for it? If you talk to manufacturers today, terms like "autonomous driving" and "Car-to-X" are already being used for the next vehicle generation. Vehicles will be controlled with different levels of autonomy. The vehicle will communicate more intensively with its environment for reasons of safety and comfort and will be further networked with services and its environment.

One thing is clear: the complexity of needed electronic systems and the transported data volume will continue to increase rapidly. To control this, it will be increasingly important to keep an eye on the whole system during the development process.

This will be achieved by already consistently implementing the AUTOSAR methodology for current vehicles today. And, in order to technically master the looming digitization of the automotive industry, AUTOSAR is bringing the "Adaptive Platform" into play. The AUTOSAR Consortium has been working since the start of 2015 on this new platform, whose technical objectives include the following points:

> Dynamic adaptation of systems during operation
> Dynamic start of applications and services – even from previously unknown external clients
> Specification of communication paths only during system start or during actual run-time of an application or service
> Safeguarding of communication in the vehicle and with nodes from the environment (cybersecurity)

The "Adaptive Platform" must also ensure by means of methodology that classic ECUs can interact with adaptive ECUs as well as with ECUs that were not developed according to the AUTOSAR standard. For this, they must be represented together in one system.

Figure 5: PREEvision network topology with Ethernet switches in detail.

Figure 6: Typical design and development process with blocks of the AUTOSAR methodology.

With these objectives, a paradigm shift in the system concept of AUTOSAR is taking place. To develop service-oriented software architectures in which different nodes subscribe dynamically to services, a different type of system description is needed. The necessary, service-oriented way of thinking in the system design has already been indicated in the last AUTOSAR versions. With the introduction of "Scalable Service-oriented Middleware Over IP" (SOME/IP) in AUTOSAR 4, the cornerstone has been laid for this.

The "Adaptive Platform" will continue this trend and thus strengthen and further establish the systems thinking. It will help developers of vehicle electronic systems even more to keep an eye on the whole system. That has always been a major benefit of development according to the AUTOSAR methodology. And it will remain so in the future.

Dipl.-Ing. Marcelino Varas
Product Manager for PREEvision at Vector Informatik GmbH. Focus on network communication and AUTOSAR. Since 2012, he has participated in the AUTOSAR Consortium "Work Package-M (Methodology)".

Translation of a German publication in Elektronik Automotive, issue 10/2016

Image rights: Vector Informatik GmbH

PREEvision
Technical Article

Design and Optimization of E/E Architectures: PREEvision as Model-Based Tool for the Next Generation

Design and optimization of modern electric/electronic architectures is more than ever a task for specialists. They must take into account numerous design criteria and connect classic vehicle domains with new requirements and trends: the introduction of domain computers, Ethernet for onboard communication, connectivity gateways and, last but not least, the increasing requirements for functional safety and information security. To master these complex tasks, model-based development tools such as PREEvision with versatile engineering functions are needed.

Many functions provided by electric/electronic (E/E) systems of modern vehicles are associated with the classic powertrain, chassis, body and multimedia domains. As a result of the advance of new technologies in the vehicle, however, domains such as driver assistance and connectivity are gaining tremendously in importance. Completely new functions are emerging here that are no longer limited just to the vehicle but are also provided in the backend outside the vehicle. Such a complex network of functions and ECUs can be handled much more easily if well-founded design decisions can be made during the design concept phase **(Figure 1)**.

Functional and ECU Network

Most E/E functions fall under the categories of open-loop control, closed-loop control, monitoring or diagnostic functions. They interact with the mechanical vehicle components via sensors and actuators. Because the sensors and actuators are installed at different geometric locations in the vehicle, E/E systems are naturally geometrically distributed systems. Many functions are networked within a domain but are also networked with one another across domain boundaries. Driver assistance functions, in particular, are closely integrated with the functions of the powertrain, steering and brake systems. For this reason, the distribution and assignment of functions to the ECU network represents a major degree of freedom in the design – and is thus an important design decision. Practically all evaluation criteria of an E/E architecture are influenced directly or indirectly by the design of the ECU network and by the assignment of functions to ECUs.

The resulting requirements for the wiring harness or the onboard communication are important criteria that must be taken into consideration early in the design process. Also there are many others: For one thing, the design criteria must be expanded to include offboard communication requirements. For another thing, driver assistance systems with cameras and radar sensors provide an environment model of the vehicle. Ethernet technology is increasingly being used here to realize the necessary transmission performance. If, as expected, Ethernet establishes itself as the future standard that is used in combination with CAN, LIN and FlexRay, enormous technical consequences for the

Figure 1: Design of the ECU network in PREEvision.

E/E architecture will arise. Networking will no longer be limited to the vehicle. Rather it will also include functions provided outside the vehicle. This, in turn, will result in increasing security requirements for the E/E architecture since the vehicle is connected to communication partners and services via wireless "over the air" connections. The trend toward automated driving, which entails increasing safety requirements, is another challenge, as is the growing number of vehicle variants and the associated increase in importance of product line, variant and platform approaches. Even the widely accepted AUTOSAR platform for the classic domains is subject to change – the AUTOSAR Adaptive Platform is currently being discussed for the new domains.

Defining the Most Suitable E/E Architecture

How can all these aspects come together in the architectural design? How can the most suitable E/E architecture be defined for this? It is obvious that the complete system must be considered. To design and optimize only a part of the system will result only in a local optimum. Moreover, the feasibility of new concepts must be evaluated early on to reduce risks in later series development. Finally, the optimization goals of a "good" architecture must also be defined **(Figure 2)**. These start with the "hard" goals that are derived from the goals of the overall vehicle, such as cost barriers, weight specifications, installation space and geometry specifications and maximum permissible electricity consumption values. These goals influence the E/E components from the hardware perspective and also influence the wiring harness **(Figure 3)**. The architecture optimization is not limited to this, however. There is another group of goals that are derived from the implemented vehicle functions. These include real-time requirements for buses and ECUs and all diagnostic and service requirements. Bus load limitations and safety and security requirements are also in this group.

E/E architectures are ultimately designed not just for one individual vehicle. Rather, a portfolio of vehicles, vehicle variants and vehicle options must be supported by an E/E architecture. For this reason, the other essential goal category of product line requirements takes into account:

> Variants and options
> Expected quantities and proportions of vehicles with the equipment
> Function-oriented decomposition or component-oriented reuse
> Building blocks with dedicated use strategies for components and subsystems

Here the solution is always a compromise between two extremes. On the one hand, there are special E/E components for a dedicated variant with low production quantities. On the other hand, there are general E/E components for multiple variants with high production quantities but with additional costs for possibly unneeded functionality. The definition of the "best" E/E architecture is thus a non-trivial multidimensional optimization problem.

Figure 2: Optimization goals of an E/E architecture.

Model-Based Approach with PREEvision

To take into consideration all relevant aspects of an E/E architecture, PREEvision supports a model-based approach. That is, all aspects of an architecture are modeled in one integral approach: network and functional distribution; hardware, wiring harness and geometry; communication and software; and all functions, features and requirements. In order to handle complex E/E systems, PREEvision follows three proven systems engineering principles – abstraction, decomposition and reuse – and supports the modeling of product lines and product variants with various model layers **(Figure 4)**.

Figure 3: Wiring harness design in PREEvision.

In the PREEvision model, the abstraction layers, from the geometry layer with installation locations and routing paths to the wiring harness layer and electric circuit layer to the ECU network layer, are represented in the vertical direction. The software and communication details are modeled in parallel with that. Hardware and software aspects are described abstractly in the layer of the logical function architecture; an even greater degree of abstraction occurs on the layers for requirements and customer features. The horizontal direction of the PREEvision model supports the decomposition. Hierarchy concepts that can be modeled bottom up or top down are available in each layer. The third direction is orthogonal to the other two directions and enables reuse and variant concepts on each model layer and hierarchy level.

Figure 4: The abstraction layers in the PREEvision model

The engineering data model is inspired and shaped by the relevant automotive standards: RIF and ReqIF for requirements, customer functions and test cases; AUTOSAR for the system, software and communication design; KBL and VEC for wiring harness and geometry specifications.

An additional benefit arises from the logical function architecture that is abstracted from the implementation in hardware and software. For many vehicle functions, this layer remains stable over many years, while the hardware, software and communication technologies for their implementation change from one vehicle generation to the next. All these layers are interconnected via so-called mappings. Safety aspects are supported by safety analyses such as HARA (Hazard and Risk Analysis), FMEA (Failure Mode and Effects Analysis) and FTA (Fault Tree Analysis) and performed on the existing product line model. The integrated model-based approach supports the single source principle. The model can be checked for inconsistencies and completeness.

With the customer feature model, a valid selection of customer features for a vehicle variant is possible that is connected to the corresponding artifacts of all other layers. As a result, an E/E architecture can be derived for selected variants and evaluated using customized optimization criteria (metrics). PREEvision supports teamwork with a collaboration platform in which a team of architects can work in parallel on one product line. All architecture artifacts are subject to version management and are linked with tickets for coordinated change and release processes.

The model-based architecture development generally does not start from scratch but rather is a further development of existing architectures. Optimizations or innovations are thereby taken into account for the next architecture generation. Typical steps here are the import of data of the predecessor architecture and its integration and validation. The innovations for the next architecture generation are then conceptualized; various alternatives are evaluated and compared with user-defined optimization goals. When the best solution is determined, the architecture is exported as the basis for the series development. In this way, a high concept quality for new E/E architectures is ensured from the outset. However, architecture development is not just an activity before the start of the development project. Rather, it is continuously performed and evaluated during series development.

Figure 5: Multidimensional evaluation of an architectural design.

Evaluating and Optimizing the Architectural Design

In practice, the E/E architecture is evaluated on the basis of typical vehicle configurations and equipment variants. Often, three configurations – basic equipment, top-selling vehicle and full equipment – are used for this **(Figure 5)**.

The basic equipment must be supplied at a low price – the cost-benefit ratio is the dominant design criterion here. The top-selling vehicle is defined as the vehicle with the highest anticipated production quantities and whose functionality must be provided for a competitive price. In the full equipment category, all E/E systems and options must work together from the perspective of all design criteria and boundary conditions such as weight, installation space, bus load, power consumption, etc., must be satisfied. The expected quantities are low and the vehicle price is high – cost criteria are not dominant here.

The optimization goals are thus not only multidimensional but they are also different for various equipment variants and vehicle configurations. In addition, they change over time such as, for example, when optional vehicle functions are rarely ordered at first but later increase in demand and even become basic equipment. For these reasons, there is currently not a generally accepted optimization criterion. Rather, the optimization strategy is viewed as an intellectual property of the vehicle manufacturer.

For this reason, PREEvision provides access to the complete data model through scripts (metrics), and the user can thus flexibly determine its optimization strategy. Over 10 years of successful use of this concept in projects at leading vehicle manufacturers around the world speak in favor of it. Large vehicle product lines here encompass millions of artifacts; the data model, engineering tools and metric interfaces undergo continuous further development in close cooperation with users.

Solid Architecture Work with Optimal Tool Support

A model-based E/E architectural design must enable the many new functions of the next vehicle generations. Here, the driver assistance and connectivity domains, in particular, are leading to major technical changes. E/E architects face the task of having to connect these innovations to the classic powertrain, chassis, body and multimedia domains.

In doing so, they must often meet multidimensional optimization goals. Approaches that consider only one category, such as the global vehicle goals of cost, weight and installation space, are not adequate.

The growing functional scope of future E/E systems is further raising the importance of solid architecture work and orderly architecture decisions and architects need a tool that supports them in all relevant aspects. PREEvision provides this support with versatile functions and and also takes into account the latest technologies and development standards of the automotive industry.

Optimization criteria can be calculated for selected equipment variants and can be flexible defined. An integral consideration of the model-based approach is that architecture work is an ongoing process that takes place in parallel with series development and requires continuous close communication among all parties involved. This is just one of the reasons that PREEvision is an eminently suitable tool for design and optimization of modern electric/electronic architectures.

Dipl.-Ing. Jörg Schäuffele
Product Manager for PREEvision at Vector Informatik GmbH.

Translation of a German publication in Automobil Elektronik, issue 11-12/2016

Image rights: Vector Informatik GmbH

VECTOR

PREEvision
Technical Article

E/E Development for Future Vehicle Innovations: An Integrated, Model-based Approach

The development of modern electric/electronic (E/E) architectures is a challenge today more than ever. Numerous development criteria must be taken into account, and classic vehicle domains need to be linked to new functions in the areas of driver assistance and autonomous driving. Completely new functions that are no longer limited to just the vehicle, but are also provided as services in the "IT backend" outside the vehicle, are emerging here. The introduction of service-oriented architectures and powerful domain computers, Ethernet to on-board communication, connectivity gateways and, last but not least, increasing safety and security requirements represent profound and radical changes for every development organization. To master these complex development tasks successfully as a team, a development platform and E/E database, which can be implemented in different ways, are required.

Many functions provided by the E/E system of modern vehicles can be associated with the classic powertrain, chassis, body and multimedia domains **(Figure 1)**, as before. As a result of the use of new technologies both inside and outside the vehicle, however, issues such as driver assistance, automated driving and connectivity (Car2x/V2x) are gaining tremendously in importance.

A comprehensive function and ECU network such as this can only be controlled efficiently if the development organization can depend on a computer-aided development environment with a centralized E/E database. Without this development environment, the expenditure required for internal coordination and the management of redundant and potentially inconsistent data increases.

This results in higher development costs, more time being required and, in the worst case scenario, errors that are first noticed out in the field.

Controlling the Function and ECU Network

Most vehicle functions provided by the E/E system concern open-loop/closed-loop control, monitoring and diagnostic functions; they interact with the mechanical vehicle components via sensors and actuators. As the sensors and actuators are installed at different geometric locations in the vehicle, E/E systems are naturally geometrically distributed systems. Many functions are active within one domain, but can also operate with one another across domain boundaries – on a function network. Driver assistance functions, in particular, work closely together with the functions of the powertrain, steering and brake systems. This can be illustrated using control chains of the required sensors, ECU functions and actuators required for implementing a vehicle function **(Figure 2)**.

Figure 1: ECU network

The definition of the function and ECU network and the distribution of the functions on the ECU network represents a high degree of freedom in the design and is thus an important architecture decision. Nearly all evaluation criteria of an architecture, such as total cost, weight, installation space, bus-load, safety and security are influenced by this. For the serial development process, this means that a differentiation must be consistently made between the two layers, i.e. function network and ECU network. In addition, these layers are not just developed for one special design variant.

Development always covers a greater number of vehicles designed based on a common platform. Typically, many people are involved and several variants (generally more than 20) are developed together. This means that the complexity of the task is enormous. For an organization the only way this can be overcome, is to keep an overview of all development artifacts and their interaction through common storage of the artifacts, visualization, computer-aided evaluation and design automation.

Process Control in the Partitioning and Integration of the E/E System

This differentiation between the function network and ECU network represents a decisive prerequisite for the successful partitioning and integration of the E/E system and the cooperation between vehicle manufacturers and suppliers. Building upon this, the tasks, roles and responsibilities for partitioning, integration of the function and ECU network can be precisely defined, planned and tracked. An overview of the entire process in E/E development is presented in **Figure 3** [1].

Figure 2: Functional network

Figure 3: Process control as a challenge in E/E development [1]

The development process must be supported by suitable IT systems. For this purpose, specific development tools are used for development disciplines such as mechanical construction, E/E development, and software and hardware development **(Figure 4)**. All the development results arising there are supplied to a Product Life Cycle Management System (PLM system) that supports the subsequent production and service processes.

Figure 4: IT support in development, production and service

Exchange Standards as the Key to Success

In E/E development, particularly at the interface between vehicle manufacturers and suppliers, the advantages of standardized exchange formats for E/E artifacts have already been recognized for a while.

By now, the required standards are available, consolidated, widely accepted and applied for years:

> **ReqIF** or **RIF** is used for the exchange of requirements, function descriptions and test cases.
> AUTOSAR provides a base software platform for ECUs that can be configured using comprehensive description formats. The most important format here is the **AUTOSAR system description**, which fully covers the areas of software architecture, the ECU network and communication for CAN, LIN, FlexRay and Ethernet with regard to the data to be exchanged between vehicle manufacturers and ECU suppliers.
> KBL is widely used for the exchange of wiring harness and geometry information, and it is expected that this standard will be supplemented with VEC over the next few years and replaced with it in the medium term.

This means that substantial volumes are covered for the E/E data exchange between the supplier and manufacturer. Text-based specification documents are increasingly being supplemented or even replaced by the exchange of "digital" specifications in these formats.

Team Platform for E/E Development

The creation and maintenance of these E/E specification elements is always distributed across teams and departments, often even beyond company boundaries. This team-based work must be enabled by the development tools used. This immediately raises the question of a suitable team and database platform on which the following basic requirements are placed:

> Multi-user operation: Simultaneous, parallel processing of artifacts by multiple users with as little potential for conflict as possible
> Traceability of the requirements over the course of implementation up to the test cases (especially with safety-related functions required by ISO 26262)
> Archiving, versioning and reuse concept for the arising E/E artifacts on a granular level
> Well-organized change and release management with integrated release and project planning, as well as transparent project tracking
> Controllable maintenance expenditure for the E/E artifacts based on a single-source principle for data storage
> Granular concept for authorization and maturity states for the support of different roles
> Support for large organizations with many users, large data quantities and various different development locations
> Integration of customers and suppliers
> Justifiable administration and operating expenditure, including backup support
> Future-proof thanks to consistent further development of the E/E development environment and migration of the available data
> Openness thanks to flexible integration into the existing IT, process and tool landscape

There are a variety of alternatives for the implementation of an E/E development environment, which are different in principles:

1. Document-based method of working:
Use of different authoring tools and storage of the created file artifacts in a centralized database (Figure 5a)

The advantages of using authoring tools tried and tested in use with the corresponding dataset are balanced out by the disadvantage of teamwork only supported on coarse granularity and traceability only available on the files created here. Close cross-team cooperation on the level of detailed E/E artifacts using the single-source principle is thus not feasible. Consistent development beyond the boundaries of the individual authoring tools is difficult and requires additional manual effort.

Figure 5a: Document-based method of working

Figure 5b: Model-based, but distributed method of working

Although many widely used authoring tools only support this method of working up until now, this is only suitable for a single workplace, and no longer appears appropriate for use in a team.

From the operational standpoint, a variety of different authoring tools have to be rolled out and a common database needs to be operated and maintained. This file-based method of working is used in the neighboring field of software development and is supported by file-based merging, updating, versioning and branching processes.

Limits can be seen in this discipline as well, however. For example, it is difficult to link a requirement to a software function present in a large file. In E/E development, the artifacts are of much finer granularity, such as requirements, functions, signals, ECUs, buses, pins and plugs.

The relationships between these artifacts are very important and, if these artifacts are "hidden" in larger files, are difficult to trace. This is why a model-based approach is required in E/E development, whereby the various different E/E artifacts and their relationships can be managed on fine granularity.

2. Model-based, but distributed method of working:
Use of different authoring tools and storage of the created model artifacts in decentralized databases (Figure 5b)

Many authoring tools are accompanied by their own team and database platform to fulfill this perceived need, but do not support all the required use cases. The integration of isolated solutions with authoring tools for various different use cases thus appears obvious at first glance. The quick use of tried-and-tested authoring tools by the team is advantageous. Limited integration continues to have a negative effect.

From the operational standpoint, multiple databases and authoring tools must be maintained and possibly even integrated. Point-to-point connections are the result. Integration is possible using standards such as OSLC, and the integrated artifacts are saved in one location based on the single-source principle, but are visible and even editable using different authoring tools.

Relationships beyond database boundaries that arise in this way are weak and cannot be reliably versioned or archived, however. Additionally, such relationships may be destroyed during data migration on a tool version update (assuming that data migration is even possible), or be impossible to create retrospectively if a new tool version introduces a new relationship possibility.

As a result, overlapping, redundant data management with proxy objects is often accepted as well; relationships must be ensured using cyclical export/import and update processes. Other disadvantages include (as before) the lack of an integrated data model, the always noticeable tool boundaries, various different architectures (e.g. rich client or web client) and different release, versioning, backup and migration concepts. This results in high integration and operational expenditures.

3. Model-based and integrated method of working
Use of a consistent authoring tool and storage of the created model artifacts in a centralized database (Figure 5c)

The weak points of a model-based, but distributed, method of working can be resolved using a consistent authoring tool and centralized storage in a database. Wherever the advantages of a consistent E/E engineering data model with centralized data storage have been identified, proprietary developments (frequently databases) often arose as the solution in the past, as a commercial product solution was simply not available.

Proprietary Solutions:
High Effort and Expensive
The advantages and disadvantages are obvious. With a proprietary solution, the data model and process can be supported at the organization in a highly targeted way. Proprietary solutions lead to high development, maintenance and testing expenditures, however. The user interface is often not state of the art as well.

Product Solution with Broad Configuration Options:
Feasible and Economical
A product solution with broad configuration options is easily justified through efficiency savings and greatly reduced integration efforts. PREEvision follows precisely this approach. A consistent and complete automotive E/E data model could not be designed until acceptance of the standards described previously was achieved in the automotive industry.

Figure 5c: Model-based and integrated method of working

The PREEvision E/E data model is the foundation for a consistent model-based method of working from the requirements to all the implementation steps through to testing.

Figure 6: The abstraction layers in the PREEvision data model

All the required aspects of an E/E architecture are modeled in an integrated approach:

> Functions, features and requirements
> Network aspects and function distribution
> Software and communication aspects
> Hardware, wiring harness and geometry aspects

In adopting this approach, PREEvision follows the three proven systems engineering principles of abstraction, decomposition and reuse and supports the modeling of product lines and product variants with various model layers **(Figure 6)**.

Considering the data model layers **(Figure 6)**, the abstraction from the geometry layer with installation locations and routing paths, the wiring harness layer and electric circuit layer to the ECU network layer is represented in the vertical direction. The software and communication details are modeled in parallel with that. Hardware and software aspects are described abstractly in the logical function architecture layer. An even greater degree of abstraction occurs in the layers for the requirements and customer features. The horizontal direction of the PREEvision data model representation supports decomposition. Hierarchy concepts that can be used bottom up or top down are available in each layer. The third direction is orthogonal to the other two directions and enables reuse and variant concepts in each model layer and hierarchy level.

An additional benefit arises from the logical function architecture abstracted from the implementation in hardware and software. The control chains for vehicle functions, which are often stable for many years, can be described here, whereas the hardware, software and communication technologies for their implementation change from one vehicle generation to the next. This applies for functions of both control and monitoring natures in the domains of powertrain, chassis, driver assistance and body, as well as for the functions of the multimedia systems. All these layers are interconnected via so-called mappings.

Figure 7: Architecture of the PREEvision collaboration platform

The storage of files created using tools for the behavior modeling of functions for simulation, rapid prototyping and code generation is possible in many locations in this model context.

Import and export interfaces for the creation and processing of the specification formats RIF, ReqIF, AUTOSAR and KBL are available. The PREEvision data model aggregates the data to be exchanged here into easy-to-understand, easy-to-edit and reusable engineering artifacts, such as ECUs, plugs, pins, software components, signals and messages and provides the necessary relationships between them. Powerful engineering tools, graphical diagrams and table-based editors are available for editing.

The integrated model-based approach supports the single-source principle, through providing cross-domain visibility of items to be analyzed. Any sub-section or the whole of the model can be checked for consistency and completeness with configurable validations. Particular benefits may be realized when considering safety aspects; the process for functional safety is supported by tools for well-known methods, such as hazard analysis and risk assessment, FMEA and fault-tree analysis, which are carried out using the existing product line data.

By using a feature model according to Feature-Oriented Domain Analysis (or "FODA") principles in the customer feature layer, a valid selection of features can be defined for a vehicle variant [2]. The features can be linked with the corresponding artifacts of all other layers. As a result, an E/E architecture can be derived, edited, checked and exported for a selected variant.

PREEvision supports teamwork with a collaboration platform in a three-tier architecture **(Figure 7)**. A team of users can work in parallel on one or multiple product lines. All E/E artifacts are subject to fine-granular version management and are connected with tickets for coordinated change and release processes.

Comprehensive configuration options come along with the mature, tried-and-tested and disclosed data model and many development tools in the product scope: roles and rights, additional application-specific attributes, maturity states, reports, table editors and views, scripts, validations and specific, even automatable, import and export interfaces.

This enables the specific needs of the respective development organization to be flexibly mapped and PREEvision to be integrated into the existing process and tool landscape. In addition, so-called perspectives with filter and editor pre-settings are able to be used to set up a user-friendly operating interface for dedicated use cases **(Figure 8)**.

Figure 8: Supported dedicated use case scopes

Summary and Outlook

Last but not least, future process support is ensured thanks to intensive maintenance and development of PREEvision by Vector to keep pace with the rapid innovation seen in the automotive industry. A migration mechanism for the inventory data in the database is provided as part of the delivery of a new version. This enables further development of the data model and the switch to new PREEvision versions with justifiable expenditure, a demanding task whose meaning is often not realized during proprietary development until it has become acute.

The PREEvision approach has been applied in productive use for years and supports the entire E/E development process in a consistent, model-based and future-proof way, from requirement management to architectural design, function-oriented system and component design, software and communication design and wiring harness development through to testing and release for serial production.

Literature References

[1] Hans-Georg Frischkorn, Herbert Negele, Johannes Meisenzahl, BMW Group, München: The Need for Systems Engineering. An Automotive Project Perspective. Key Note at the 2nd European Systems Engineering Conference (EuSEC 2000), München, 13. September 2000.

[2] Kang, Kyo c. et al: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21 ESD-90-TR-222, Software Engineering Institute, Carnegie Mellon University, Pittsburg, Pennsylvania, USA, November 1990. www.sei.cmu.edu/reports/90tr021.pdf

Authors

Dr. Thomas Beck
Vector Informatik GmbH
Managing Director

Dr.-Ing. Clemens Reichmann
Vector Informatik GmbH
Technical Director PREEvision

Dipl.-Ing. Jörg Schäuffele
Vector Informatik GmbH
Product Manager PREEvision

Originally published in "ATZ elektronik" magazine
Issue 06 – December 2016

PREEvision
Technical Article

Integrated Development of a Complete E/E Architecture: Reaching Goals Reliably Using a Model-Based Approach

Specification and modeling tools for subtasks in the development of electric/electronic systems are widely used today. This is not due in small part to the fact that their use is what enables today's complex development processes to be controlled. Abstraction allows for transparency in domain concepts and systematically avoids repeated maintenance, which increases development speed significantly. Integrated, model-based system development for the entire E/E architecture still seems like a far-off dream, though.

An E/E architectural design has to enable the many new functions of the next generation of vehicles and must also consider and reserve space for future model upgrade functions. The driver assistance (ADAS) and connectivity (Car2x/V2x) domains, in particular, are leading to major technical changes here. Completely new functions which are no longer limited to just the vehicle, but are also provided in the "backend" outside the vehicle, are emerging here. The increased complexity and variety of concepts also has to be faced. E/E architects face the task of having to connect these innovations to the classic powertrain, chassis, body and multimedia domains. In order to do this, you will need a tool which supports you in all the important aspects and enables solid architecture work and well-founded architectural decisions **(Figure 1)**.

Domain-Specific Language

In comparison to a purely textual description, a system using a model-driven approach is formally described. The formal description requires a specialized language which is clearly comprehensible within a development domain. If the language can also be learned intuitively, it quickly ensures interpretation-free communication within the organization. Specialized views are also required for modeling domain-specific technical data. These views enable both the entry of data as well as presentation **(Figure 2)**.

Figure 1: Communication relationships between vehicle domains from a functional viewpoint.

Figure 2: Data model and domain-specific representation in PREEvision.

The advantages of a formal system description are alluring. In addition to easy, time-saving entry, the technical data is also always syntactically correct. Semantic inconsistencies and incomplete models are possible and, at certain times, are even tolerated in the development cycle. Formally described systems can be automated for consistency and completeness checking at any time, however, and errors can even be corrected automatically on occasion here. This consistency checking is an early quality assurance measure at the time of design. It minimizes the risk of errors or missing information not being noticed until late in the development or manufacturing process and then requiring extensive correction. Quality-assured architecture modules also enable more efficient integration processes during the late development phases.

Another advantage of formally described technical data lies in multiple use, where identical data can be displayed from different viewpoints in specific views. The channeling of data into documents and reports is also possible, as is the conversion into standard formats for data exchange. In sub-domains, e.g. the development of application software, a model-driven development approach with the advantages mentioned above is indispensable today.

Modeling Principles

To be able to take all the relevant aspects of an E/E architecture into consideration and keep them consistent, a model-based procedure which models all the required features in an integrated approach and displays them in specific views is favorable. The required technical layers include functions, features/requirements, software/communication, network/function distribution, hardware, wiring harness and geometry. Two other things are required to control complex E/E systems. The first thing is to follow the three tried-and-tested system engineering principles: abstraction, decomposition and reuse. Secondly, support for modeling in product lines and variants with model layers is required. Dividing a technical model into layers also enables a decoupled manner of working. For one thing, it allows many distributed users to work in a collaborative environment and, at the same time, on a single model. Temporal decoupling from development phases is also possible, such as with software and hardware development.

The PREEvision system design tool is structured in the technical layers mentioned **(Figure 3)**. In the data model, everything from the abstraction layers from the geometry layer with installation locations and routing paths to the wiring harness layer, the electric circuit layer and the layer of the ECU network are represented in the vertical direction. The software and communication details are designed in parallel with that. Hardware and software aspects are described abstractly in the layer of the logical function architecture; an even greater degree of abstraction occurs on the layers for requirements and customer functions. The data model is built in the layers mentioned above in such a way that it is understood intuitively in each respective technical area and functions as a domain-specific language. At dedicated points, the layers are available via mappings, which contributes to the decoupled manner of working.

The horizontal direction of the PREEvision model supports decomposition. Hierarchical concepts are available in each layer through decomposition. The third direction in the model is orthogonal to the other two directions and enables reuse and variant concepts on each model layer and hierarchy layer. Development progress can also be tracked via integrated version management, and work products can be frozen at specific points in time.

The engineering data model is inspired and shaped by the relevant automotive standards: RIF and ReqIF for requirements, customer functions and test cases; AUTOSAR for the system, software and communication design; DBC, LDF, FIBEX for specific bus technologies, KBL and VEC for wiring harness and geometry specifications.

Figure 3: PREEvision modeling layers for E/E system design.

Function and ECU network

In previous E/E architectures, most E/E functions can be classified as control, regulation, monitoring or diagnostic functions. They interact with the mechanical vehicle components via sensors and actuators. As the sensors and actuators are installed at different geometric locations in the vehicle, E/E systems are geometrically distributed systems. Many functions are networked within a domain, but are also networked with one another across domain boundaries. Driver assistance functions, in particular, are closely integrated with the functions of the drive, steering and brake systems. For this reason, the distribution and assignment of functions to the ECU network represents a major degree of freedom in the design – and is thus an important design decision. Practically all evaluation criteria of an E/E architecture are influenced directly or indirectly by the design of the ECU network and by the assignment of functions to ECUs.

The resulting requirements from function distribution for the wiring harness or onboard communication are important criteria which must be taken into consideration early in the design process. There are many others as well: For one thing, the design criteria must be expanded to include offboard communication requirements. For another thing, driver assistance systems with cameras and radar sensors provide an environment model of the vehicle. Ethernet technology is increasingly being used here to realize the necessary transmission performance. If, as expected, Ethernet establishes itself as the future standard that is used in combination with CAN, LIN and FlexRay, enormous technical consequences for the E/E architecture will arise. Networking will no longer be limited to the vehicle. Rather it will also include functions provided outside the vehicle.

Optimization Goals

How can all these aspects, which typically have several hundred criteria, be united in the architectural design? How can the most suitable E/E architecture be defined for this? It is obvious that the overall system must be considered. To design and optimize only a portion of the system will only result in a local optimum. Moreover, the feasibility of new concepts must be evaluated early on to reduce risks in later series development. Finally, the optimization goals of a "good" architecture must be defined.

The goals of the overall vehicle, such as cost barriers, weight specifications, installation space, geometry specifications and maximum permissible electrical energy consumption values are to be considered first. These goals influence the E/E components from the hardware perspective and also influence the wiring harness. Architecture optimization is not limited to this, however. There is another group of goals that are derived from the implemented vehicle functions. These include real-time requirements for buses and ECUs and all diagnostic and service requirements. Bus load limitations and safety requirements are also in this group. E/E architectures are ultimately not just designed for one individual vehicle. Rather, a portfolio of vehicles, vehicle variants and vehicle options must be supported by an E/E architecture. For this reason, the other essential goal category of product line requirements takes the following into account:

> Variants and options
> Expected quantities and proportions of vehicles with the equipment
> Function-oriented decomposition or component-oriented reuse
> Building blocks with dedicated use strategies for components and subsystems

Here, the solution is always a compromise between two extremes. On the one hand, there are special E/E components for a dedicated variant with low production quantities. On the other hand, there are general E/E components for multiple variants with high production quantities but with additional costs for possibly unneeded functionality. The definition of the "best" E/E architecture is thus a non-trivial multidimensional optimization problem.

Architecture Development Process

Model-based architecture development generally does not start at zero, but rather is a further development of existing architectures. Innovations and optimizations are thereby taken into account for the next architecture generation. Typical steps here are the recording and consolidation of data of the predecessor architecture and its integration and validation. The innovations are then designed for the next-generation architecture. Various alternatives are evaluated and compared with user-defined optimization goals. When a solution is determined, the architecture is exported as the basis for series development. This enables the fast introduction and implementation of new architectures in serial developments with high concept quality at the start of development.

Architecture development is not just an activity carried out before the beginning of the series development project, though! In fact, it occurs progressively and at the same time. Architectural decisions and changes are continuously being made during series development and thus affect the architecture of the following generation. This is why close cooperation and exchange between the architecture team and series development is absolutely necessary during the entire development process.

Summary and Outlook

To control the growing functional scope of future E/E systems, an integrated view of the overall architecture within and outside of the vehicle is required. Service-oriented E/E architectures, in particular, are increasingly being used in the driver assistance domain and connectivity environment. First and foremost, they identify the challenges of future architectures!

The consistent use of model-based tools such as PREEvision lends itself to the approach and mastery of these challenges. The many advantages of such comprehensive solutions for the overall E/E development process help architects to implement the innovations of the next generation of vehicles and reliably help them reach their goal.

Dipl.-Ing. (FH) Markus Helmling
Product Management Engineer at Vector Informatik GmbH. Responsible for PREEvision since 2012 with focus on AUTOSAR communication design.

Translation of a German publication in Hanser Automotive, issue 11-12/2016

Image rights: Vector Informatik GmbH

PREEvision
Technical Article

Service-oriented Architectures and Ethernet in Vehicles: Towards Data Centers on Wheels with Model-based Methods

Service-oriented architectures ('SOAs') have been used for years in the IT industry to describe and structure distributed systems. However, service-oriented design is also becoming extremely significant in the automobile industry: how else can customers' increasingly urgent demand to adapt the scope of functions of the automobile, similar to the possibilities of today's smartphones, be met reliably? It is also necessary to handle additional requirements from model maintenance and to introduce technology such as autonomous driving and vehicle-to-X communication.

The software development challenges for today's vehicles are enormous. Besides the complexity in automobile manufacturing, which has been increasing for years, the introduction of SOA (pronounced "so-ah") also leads to completely new development paradigms and processes. Conway already postulated in 1968:

> "organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations." [1]

Thus it is no wonder that the software architecture of today's vehicles mostly has a silo structure and reproduces traditional organizational structures according to domains such as powertrain, body and chassis. However, effective introduction of a SOA requires completely different cross-domain coordination and communication. Here responsibilities move beyond the previous departmental and organizational boundaries; cooperation and collaboration in a company become implicit preconditions for determining and further developing a SOA and its interfaces. The organizational change also affects the personnel domain. New roles and job requirements arise in the company; the necessary specialists must be trained or recruited.

On the technical side, through service-based communication the requirements intensify the trend towards enormous growth in bandwidth demand in onboard networking. In this regard vehicle manufacturers are relying increasingly on Ethernet technology, which has proven successful in the IT industry. This allows implementation of appropriate bandwidths for applications and communication within the vehicle, as well as between the vehicle and its environment or the Internet. Therefore, tools that take these requirements into account during design and maintenance of the systems are necessary in order to develop future-proof electrical/electronic architectures.

The Advantages of Service Orientation

With a SOA, a service landscape arises that is characterized by clearly defined interfaces. Ideally, these interfaces are described formally and unambiguously, both syntactically and semantically. Clear definition makes it possible to structure services. For example, layered architectures are generally used which make convey the dependencies on services and represent clear design rules. Furthermore, in a SOA the components are loosely coupled to each other with a service bus as the middleware. The middleware is the intermediary between the service provider and the service consumer; it controls the communication between the service provider and service consumer at system start and usually also defines the data serialization that is managed by the Basic Software ('BSW') or Operating System ('OS') of the controller **(Figure 1)**.

This precise calculation rule, called a "serializer", describes the data transfer in the physical bus system. The serializer determines how the data is transformed into a serial bit stream and how it is deserialized on the receiving side.

The use of middleware creates the connection between the service provider and service consumer at runtime – and not at the system design time as previously. This addresses various problems of previous software development. For one thing, partial updates of the system can be performed: if backwards compatibility is ensured, the change to a service does not necessarily have an effect in a software adaptation for all consumers of a service. "Soft" migration scenarios are also possible through use of version management and revision control of service interfaces; this allows all parts of the system to use an older version of the service while another part is already using a new aspect of the service interface. Furthermore, failure safety can be introduced in the design: in the event of failure of one service instance, another equivalent, or at least compatible, service instance can take its place. In distributed systems, that means that not all Electronic Control Units ('ECUs') involved in a service necessarily need to be updated.

Last but not least, a consistent service design also allows more powerful design patterns in the implementation. For example, Remote Procedure Calls ('RPCs') are common practice in a service-based architecture. RPCs were previously only used within an ECU but now lead to use of client-server paradigms in onboard networking as well. Consequently, the signal orientation in the communication design of vehicle, which had previously been so pronounced, is suppressed.

The SOA Method – A Top-down Approach

If automobile manufacturers and suppliers want to demonstrate the advantages of a SOA, they need an appropriate method that supports the design of new systems with all necessary development steps. The goal here is to manage the complexity while simultaneously making quality-assured designs possible. The method described below explains the development process from the requirements, through the service architecture, to the communication design.

Figure 1: Schematic illustration of services and middleware

Figure 2: Logical architecture for static description of the service orchestration

The goal is an AUTOSAR Classic compliant system description for a vehicle or vehicle subsystem. However, the method can also be applied in the context of AUTOSAR Adaptive or to systems outside of AUTOSAR.

The development process starts with the idea of a new functionality whose features are specified. Nowadays this is usually structured according to use cases. The notation of use case diagrams from the Unified Modelling Language ('UML') standard has proven successful and established itself for this purpose. With use cases, services can be derived from a catalog of features. Various views are need in order to structure services and illustrate dependencies.

One obvious and already-established view is oriented towards the information flow and thus illustrates the dependencies of functions. The so-called "logical architecture" has established itself for this purpose in the system tool PREEvision: here the logical architecture can be represented in the form of a block diagram and thus describes the functional connection of individual service functions independently of the implementation in hardware or software through event chains **(Figure 2)**.

In addition the UML profile "SoaML", which is specialized for the description of service architectures, defines logical nodes called "participants". Participants arise as both providers and consumers of services – in both cases, the exchange takes place via so-called "SOA ports". These make dependencies between services visible – for example if a participant is the consumer of one or more (basic) services and offers high-quality services in turn. This design step is called "orchestration" and is similar to the widespread event chain view for control functions.

However, in fully dynamic systems in which multiple instances of a service exist within the system as different versions, this representation of the allocation of services to participants is a scenario that can only arise at runtime. Since in the automotive sector services are currently primarily assigned statically to dedicated hardware, this representation can also be understood as the allocation of a service role (provider/consumer) to an ECU. Besides that, participants also make various constellations that arise at runtime visible. Thus the representation of logical service nodes as participants can also be seen as a helpful analysis tool. Last but not least, being familiar with and documenting these constellations is an important foundation for testing and safeguarding such systems **(Figure 3)**.

The SOA ports addressed above are typed by a service interface that is described unambiguously by multiple aspects. It is first necessary to clarify what role a port plays – does it provide a service, or is it a consumer of one? The properties of the interface are described here syntactically: generally valid, technology-independent properties such as available methods, fields, and events – but also data types that are already known at this point and are supported by the middleware, for example the parameters of a method. This completely describes the static behavior of a service.

The semantics of the service contract between the SOA ports are frequently described in text form. This specifies time sequences between service roles and represents protocols, among other things. This design step is also called "service choreography". The collaboration or message sequence charts ('MSCs') proposed by the UML standard are usually used for visualizing sequences.

Figure 3: SOA diagram for the definition of logical service nodes (participants) in PREEvision

The formal description of a service, which is initially technology-independent, is typically transported via an interface description (Interface Description Language, 'IDL'). In modern system design tools like PREEvision, the external structure of the implementation can already be derived from this, for example for the application in the context of AUTOSAR Classic.

This creates the implementation bodies that are necessary for providing further detail about the design and implementing it. Furthermore, the role of the service and the description of the service interface are taken over. The advantage of this consistent approach: changes to the role of a service or the description of a service interface can be carried over to the technology-specific implementation automatically through synchronization methods. Software design interfaces are then derived from the IDL descriptions and represented in the AUTOSAR Classic platform as sender-receiver and client-server interfaces. If file types appropriate for the technology have already been selected, they can be taken over automatically.

The introduction of a special "service interface" is being discussed for the AUTOSAR Adaptive platform. This is intended to contain the properties described above directly and take full advantage of a technology-independent description of services. The goal is to derive implementation bodies for various technologies from the same service definition with the push of a button. The middleware-specific startup behavior should be specified and parameterized after this step. This process is called "service discovery".

Communication Design

During the further procedure, the data supply and structures of the communication description depend on the communication bus technology and middleware used. AUTOSAR uses Service Oriented Middleware over Internet Protocol ('SOME/IP') by default as the transformer for service discovery and data serialization.

However, AUTOSAR also provides for the use of other transformers and middleware. If Ethernet and Internet Protocol ('IP') are used as the communication channel, an essential work step is defining sockets in terms of their IP address, transport protocol, and port. And while a description of socket addresses suffices for AUTOSAR Adaptive, for AUTOSAR Classic the signal level must also be specified – since the AUTOSAR Classic basic software stack is designed for signal-based communication using, for example, FlexRay or the more widely-used Controller Area Network ('CAN') bus and Local Interconnect Network ('LIN') sub-bus. Here integrated system design tools offer a crucial advantage: they make it possible to derive the signal communication, based on the work products of the previous steps. The service description and information about where in the system a service is to be provided – i.e., which

Figure 4: From service design to communication design

specific ECU implements the provider and which one the consumer – is necessary for this purpose. At this point PREEvision also transfers further properties from the service description into the communication description, for example service and method IDs **(Figure 4)**.

Summary

The development tool PREEvision supports the methodical, consistent design of a service-oriented architecture. The user is guided from the definition of the service interfaces, through specification of the interaction of services, to an AUTOSAR compliant Ethernet design through an integrated workflow. If other bus technologies such as CAN, LIN, or FlexRay are also to be used in addition to Ethernet, mixed topologies can also be designed. Thus PREEvision accompanies the system designer through the challenging task of combining classic embedded design with modern service orientation and the necessary back-end communication – and thereby supports the transformation of the automobile into a data center on wheels.

Literature References

[1] Conway's law: http://en.wikipedia.org/wiki/Conway%27s_law (January 19, 2017)

Authors

Dipl.-Ing. (FH) Markus Helmling
Vector Informatik GmbH
Product Management Engineer PREEvision

Originally published in "Elektronik automotive" magazine Special issue "Automotive Ethernet" – March 2017

VECTOR

PREEvision
Model-Based Electric/Electronic Development

From Architecture Design to Series Production

PREEvision is the powerful premier tool for the model-based development of distributed, embedded systems in the automotive industry and related fields. This engineering environment supports the entire technical development process in a single integrated application. Your advantages:

> Design, evaluation and optimization of Electric/Electronic architectures
> Product line approach
> Integrated requirements management
> AUTOSAR system and software design
> AUTOSAR communication design
> Wiring harness development
> Test engineering and management
> Development of safety-related systems
> Collaboration and parallel multi-user operation

See for yourself: **www.vector.com/PREEvision**

PREEvision – The E/E Engineering Solution.

Vector Informatik GmbH | Germany · Austria · Brazil · China · France · India · Italy · Japan · Korea · Sweden · UK · USA | www.vector.com

VECTOR

PREEvision
Technical Article

Model-based Solutions for Medical Systems:
An Integrated Approach for the E/E Development Process

In common with many other industries, medical technology is facing a profound upheaval. Medical devices are networked, collect data, adapt behavior and can learn independently. The days of developing a single isolated medical device are long gone – today, development focuses on medical system solutions with widely varying complexity. To keep increasing complexity manageable, model-based processes for electrical/electronic development are now available.

"How Apple wants to take over the medical market" – This announcement of Apple's development of a blood sugar meter[1] makes it abundantly clear that the medical industry (like many other industries) is either in, or soon to be faced with, a profound upheaval. Digitalization is having a lasting effect on more than 11,000 companies and over 190,000 employees in Germany.

This being the case, both products and their development need to be digitally updated. Instead of closed systems produced in small quantities, for example computer tomographs, or smaller sensor devices produced in large quantities, such as blood sugar meters, networked system solutions are increasingly in demand today. These interoperable systems collect data, place it in private and public cloud storage and are provided with over-the-air (OTA) software updates.

The development of these kinds of systems requires interdisciplinary cooperation between a variety of different experts, as well as accelerated and dynamic development processes. This is why established methods are being called into question and new tools are being used for development support.

The Challenge Polygon as a Decisive Hurdle

Risk management, approval periods, financial pressure, developed structures with established methods, their compliance, etc. – the challenges involved in electronics development for medical products vary greatly.[2] Add to this ever-shorter product introduction periods (time-to-market) and faster innovation cycles in relation to the gradually rising complexity of developments for new networked system solutions.

Naturally, the already established high level of product quality compliant with international standards must also be maintained. Increasingly, the systems within projects are themselves composed of systems networked with these systems, something which makes management of the developments more complex. As a result of this, the magic triangle of project management is turning into a polygon in which each triangle represents a system development sub-project **(Figure 1)**.

Coping with the diversity of these challenges in (sub-)projects is the decisive hurdle for large corporations, mid-sized firms and startups alike when it comes to existence in the market.[3] Breaking down the points listed above to a deeper level leads to specific tasks which need to be solved. These tasks are examined in more detail below:

Figure 1: Increasing complexity in medical product development.

- Transfer of the previous electronic development to the development of networked products or system solutions
- The systemic use of a modular development in electronics
- The ability to respond to accelerated changes in development due to globalization
- The creation of requirements for OTA updates and software versioning in the products
- The addressing of increasing safety requirements

From Isolated Devices to Modular and Networked System Solutions

Increasingly, networked products and system solutions are being desired, or even expected, by customers – the development of an isolated device for the measurement of specific vital signs such as blood pressure and blood sugar simply no longer fits in in today's world. The electronics of these devices must feature interfaces enabling networking with other devices, and the variety of networking technologies including Bluetooth Low Energy, WiFi and IoT narrowband, together with proprietary protocols, need to be taken into account when it comes to electronics development. In some cases, even on-demand solutions developed in a different way from the classical design and implementation of an embedded system are required within the system.[4]

Ever-shorter innovation cycles in medical-technology development require even greater reuse of specific hardware and software components in products, as full redevelopment from the ground up is often too time-consuming. Thus measuring instruments, for example, including the underlying electronics, are reused in multiple product generations. Software components such as the user interface, however, have to be replaceable or expandable in the device within a generation. One totally new challenge is the development of modular electronic components which enable parts to be extracted and replaced more easily. These devices might include sensors used in intensive care and medication dosing, for example.

Cross-Location Development for Dynamic Global Markets

Another challenge, in addition to accelerated innovation cycles, is the timely and need-based response to the changing market conditions of medical products sold internationally. From a development standpoint, specific devices for

different countries and markets are also the best solution due to different approval requirements, but this approach is not usually economical. As a result, medical products are divided up according to their functionality and matched to the individual requirements of the respective countries. This presents E/E development with the challenge of being able to evaluate the addressing of a market and its particular features in advance.

Increasing globalization also means that the development team has to change the way it thinks, as electronics development has not been carried out at a single location by a select team for a long time now. Agile development methods require flexible tools which support a variety of different teams in different languages and which possess a common database. Decision-making bodies should not be overlooked here, either. The creation of documentation and architecture-based decision criteria during the development process must be comprehensible to everyone involved.

Evolution and Functional Reliability of the E/E Architecture

As software is continually gaining in importance when it comes to medical products and is also primarily being created in a modular way, the architecture of the components already existing in the product, including those which must be replaced, also needs to be reconsidered. Increasingly, products require over-the-air software update capabilities, as well as the ability to remedy software errors within a very short period of time through the use of updates. In addition to the interfaces to be implemented in the electronics for this purpose, the development of appropriate software components for validating and verifying the new software parts in the device is required.

To know the current version of the software in the product at all times, software version management also has to be conducted synchronously with electronics development. However, the implementation of interfaces or exchangeable software components in the electronics may not influence the functional reliability of the systems at any time. This is why electronics development requires the use of methods which can address changes in standards with regard to the reliability of the devices in a simpler way. Accordingly, approaches such as security by design and the early integration of fail-safe functions are being used on an increasing basis. Overall, it is becoming clear that the combination of new and existing challenges in the field of E/E development for medical products requires new processes and methods in development.

Figure 2: Schematic representation of the levels of abstraction in the PREEvision model.

Figure 3: Reuse and version concepts in PREEvision already take country-specific versions into account during early stages of development.

A Model-Based Approach as a Solution

The challenges described above illustrate the increasing complexity in developing modern electrical/electronic systems in the field of medical technology. When designing and optimizing the architecture of distributed and networked systems, for example, it is important to not only consider a single sub-system, but to design the entire system as a whole. This is the only way that the optimization criteria of the individual sub-systems can be merged and a global optimum for the overall system architecture found.

Many design criteria need to be taken into account here, such as different versions, OTA update capabilities and aspects of functional reliability. The feasibility of new concepts also has to be evaluated early on to reduce risks in later series development. Coping with these complex tasks requires model-based development tools such as PREEvision from Vector Informatik with its variety of engineering functions and accompanying process support.

The model-based procedures of PREEvision take all the relevant aspects of developing complex E/E systems into account, an approach which has already been proven successful in the automotive industry and other industries over the past few years and which is able to be transferred to medical technology as well. To handle the complexity involved, PREEvision follows three proven system engineering principles – abstraction, decomposition and reuse – and consistently supports the modeling of product lines and product versions with different model layers.

System Engineering with Levels of Abstraction

The PREEvision data model **(Figure 2)** shows the various different abstraction levels vertically. The hardware can be modeled by the physical geometry with cabling via the electronics logic using system switching and circuit diagrams through to the network level. The system software architecture and communication details are modeled in parallel with it. When the software architecture is created, both function- and service-oriented approaches are supported, and may be used side-by-side in a single product.

In modeling the logical function architecture, an added benefit is created, as abstraction occurs through concrete implementation in hardware and software. This represents considerable added value, since this level remains stable for many system functions over a longer period of time as hardware, software and communication technologies

change from product generation to product generation. On the higher levels, the collected requirements and identified customer functions are encapsulated in the E/E systems to be created. This complete representation of all levels of abstraction enables all the effects of changes and developments in the model to be made comprehensible. To this end, PREEvision links all levels together using so-called "mappings," which connect the artifacts on the various different abstraction levels with one another.

The horizontal direction of the model supports decomposition. Hierarchy concepts which can be modeled bottom up or top-down are available on each level. The third direction is orthogonal to the other two directions and enables reuse and version concepts on each model and hierarchy level. This makes it possible to take country-specific versions of the medical products to be developed into account during early stages of development and identify reusable components **(Figure 3)**. With the help of mappings, valid versions can also be created automatically using propagation rules.

PREEvision supports teamwork with a collaboration platform, allowing teams of software and hardware developers can work in parallel on one product line. All architecture artifacts are subject to version management and are linked with tickets for coordinated change and release processes. Maturity level models can be depicted in PREEvision using life cycles in conjunction with consistency checks. In this way, greater degrees of freedom can be made possible during early stages of development – the closer the level of development is to the start of production (SOP), the greater the number of consistency checks which can be activated and conducted automatically.

Using this type of approach, risks within the project can be identified early on and reduced using appropriate methods. Safety aspects are supported by safety analyses such as HARA (Hazard and Risk Analysis), FMEA (Failure Mode and Effects Analysis), and FTA (Fault Tree Analysis) and performed on the existing product line model. PREEvision also supports test engineering, the control of tests and the management of accumulated test data across the entire E/E development process. Test cases can be assigned to requirements directly.

Conclusion

The model-based approached presented here, as implemented in PREEvision, can contribute considerably to the efficient circumvention of increasing complexity in the development of E/E system solutions. This makes key to successfully participating in the digitalization of industries in which electronics development is of ever-increasing importance. In addition to the automotive and manufacturing industries, this includes medical technology in particular.

References

[1] How Apple wants to take over the medical market: http://www.handelsblatt.com/unternehmen/it-medien/diabetes-therapie-wie-apple-den-medizin-markt-erobern-will/19668928.html (April 30, 2017)

[2] The seven biggest problems of medical technology: http://www.franklange.eu/die-7-groessten-herausforderungen-der-medizintechnik/ (April 30, 2017)

[3] Five hurdles for health care startups – and how to overcome them: https://www.deutsche-startups.de/2015/10/09/5-huerden-fuer-health-startups-und-wie-man-sie-bewaeltigt/ (May 10, 2017)

[4] Software-on-demand describes specific software operated accordingly by the manufacturer and which is scalable on request. This enables flexible forms of licensing which can also be charged according to the data volume or the number of users.

Authors

Dr. Stefan Hellfeld
B&W Embedded Solutions GmbH
CEO

Dr. Ulrich Berthold
Vector Informatik GmbH
PREEvision Business Development Manager

Originally published in "Elektronik Journal" magazine issue "Medical Electronics" – October 2017

Electric-electronic architectures (E/E) in vehicles – What's the first thing that comes to mind? Complex systems which have been becoming increasingly more extensive for years? Where an ever-greater number of components build upon one another and are dependent on one another? Where the individual components are subject to continual adjustment and expansion during development? And that it's becoming increasingly difficult to maintain an overview for recognizing correlations and dependencies? If so, you should take a closer look at the advantages of an integrated development tool and requirement management.

Whether it's networked systems, convenience electronics and multimedia, driver assistance systems, autonomous driving, connectivity or cloud applications, the number of communication systems in vehicles is increasing relentlessly and E/E architectures are becoming more complex and diversified. Despite this, dependencies need to be quickly and easily identifiable when developing E/E systems – everyone involved has to be certain at all times as to where the modification, adjustment or expansion of a system requirement needs to be taken into account. Questions like: "Which components are affected by modified requirements?" and "What test cases need to be adjusted or run again?" must be answered without delay. If safety requirements or legal regulations change, the places in the system where they must be adjusted have to be traceable. Classic requirement management just doesn't cut it here anymore.

The Limits of Classical Requirement Management

With classic requirement management, requirements were originally collected in text editing or spreadsheet programs without any help from special tools. Traceability was as good as impossible using this method. It wasn't until independent, self-contained tools for requirement management came along that it was possible to relate requirements in a structured way and with one another – something which ensures general traceability. By linking to external data or other content, the limits of tools can also be surpassed. The problem here is that these links often quickly become obsolete or invalid. This is especially the case when working across multiple tools or databases, where links cannot be defined on the atomic level and are difficult to maintain.

While a small team used to be sufficient for creating requirements, the complexity of today's systems necessitates the management of an interdisciplinary team – complete system or component specifications don't come about until system and function developers, architects responsible for components, communication designers, test managers and other areas of responsibility work together. Classic requirement management systems quickly reach their limits here as well.

Figure 1: A disadvantage of heterogeneous development environments without consistent data models: Noticeable limits to stand-alone authoring tools for E/E systems and requirement management.

Integration as the Key to System Mastery

Most vehicle functions provided by E/E systems concern control, regulation, monitoring and diagnostic functions. They interact with the mechanical vehicle components using sensors and actuators to be installed in various geometric locations in the vehicle. This means that E/E systems are also geometrically distributed systems which work closely together in a distributed function network, such as driver assistance functions with the functions of the drive, steering and brake systems.

This is why comprehensive E/E architectures with complex and intermeshed function and control unit networks can only be efficiently mastered in a computer-aided development environment with a central database. Without a central E/E database like this, the expenditure for internal alignment and maintenance of redundant and possibly inconsistent data and requirements quickly rises.

This results in high development costs, more time being required and, in the worst case scenario, errors which are first noticed out in the field. The only way this enormous task can be overcome is for an organization to maintain an overview of all development artifacts and their correlations through common storage of the artifacts, visualization, computer-aided evaluations and consistent traceability. With the combination of standalone authoring tools for the E/E system and requirement management tools which are independent of it still being widely used today, however, it can only be implemented with restrictions or not at all. This is because cross-tool and cross-division cooperation, as well as relation on the atomic level, are not possible with it. Point-to-point connections often arise here. Integration of the individual tools is partially implemented using standards such as OSLC (Open Services for Life cycle Collaboration); the integrated artifacts are then visible in the different tools and can also be edited to a degree there, but the relationships beyond database boundaries which have arisen in this way cannot be reliably versionized and archived **[Figure 1]**.

Redundant data management with substitution objects using cyclical import/export and update processes has the disadvantage of an inconsistent data model. This results in noticeable tool limits, a variety of different architectures (such as Rich Client and Web Client) and a variety of different concepts for release, versioning, backup and migration in the tools. A large integration and operation expenditure arises here, yet the solution only partially satisfies the actual requirements.

Integrating Requirements directly into the Development Process

The solution to this dilemma is provided by the next evolutionary stage in requirement engineering: integrated requirement management within the E/E development environment as implemented in the PREEvision E/E architecture tool. Here, requirement management is integrated directly into the development process **[Figure 2]**.

Involved persons from all areas work together on a database, which provides a seamless and transparent view of the overall system and enables structured cooperation. Thanks to the model-based approach of PREEvision, the various different E/E artifacts and their relationships are dissolved and managed in great detail. For a variety of different abstraction levels and roles, consistent and valid documents are created in this environment. During the development phase, changes can be responded to efficiently and quickly at any time as well. Fundamental requirement management functions are available for this purpose:

> Collecting requirements
> Hierarchical structuring
> Linking and tracking
> Validating
> Importing and exporting (ReqIF, RIF and Excel)
> Creating system specifications or performance specifications

In addition, integrated requirement management enables seamless interaction between the requirements and all development artifacts in the model. Not only the requirements, but the development artifacts themselves as well, serve as requirements in the integrated approach. These are linked with requirements or embedded directly in requirements using placeholders.

The integration of individual property values of an artifact is also possible. In E/E development, the concrete artifacts are often defined in much more detail than is common with conventional requirements. For example, the properties of the individual pins of a plug are seldom described in classic requirement management. With the integrative approach, however, they can be collected automatically and part of the requirement specification can be used. The placeholder principle, which works in a similar way to field functions in text editing programs, ensures that the current values are always accessed automatically in the dynamic development process here.

Figure 2: Modeling levels of the PREEvision E/E development tool.

Figure 3: Export of any desired model parts in the ReqIF format: A ReqIF object can be created for each node of the network and for the connections between the nodes for a selected hardware architecture. PREEvision then collects this data in a ReqIF file.

Developing Requirements Seamlessly, Consistently and Completely

All software and hardware components in the development model can be seen as requirements and used in system specifications or performance specifications. With integrated requirement management, you continually receive a current and detailed requirement specification, rather that a text-only description. Integrating and augmenting details from the E/E development data ensures consistency, completeness and comprehension.

The model-based tool environment with integrated requirement management permits top-down, bottom-up and middle-out approaches here, where the requirement specification is either classically based on requirements and carried over to concrete E/E artifacts or derived directly from the E/E artifacts. Mixed operation with requirements and specifications defined in concrete E/E artifacts is also possible, offering advantages from both worlds. For secure data exchange with suppliers, both requirements and any other desired E/E model artifacts are exported in the ReqIF exchange format over the standardized ReqIF interface of PREEvision. They are then available to other tools for integration **[Figure 3]**.

Developing Requirements Together and Simultaneously

Many users with different roles don't just work on a single component or a single system on a single team. This is why it must be possible to access, create and manage E/E artifacts, requirements and test cases across the organization.

In addition, a rights management system which provides monitored access and role-dependent editing rights is required. Ultimately, highly detailed version management is required in order to ensure unique and reproducible development statuses. Joint simultaneous work on the development and management of E/E systems in vehicle and automobile manufacturing is supported in PREEvision with a collaboration platform in a three-tier architecture.

Its functions are designed for structured creation and implementation of complex E/E projects in large organizations. The central database ensures a homogeneous dataset, which enables organization-wide multi-user operation. Project and development teams have shared access to the necessary data and work simultaneously without data conflicts. Expensive processes for merging datasets are thus eliminated. Versioning is carried out with functions for checking in and checking out. As a result, requirement development also remains traceable over time **[Figure 4]**.

Developing Requirements in a Transparent and Process-Reliable Way

Thanks to the model-based, integrated approach of PREEvision, persons responsible for components can also view the requirements of the system level and access a concrete implementation. As a result, the requirements which are implemented by a specific component remain identifiable. Conversely, requirements which have not yet been covered can be identified in this way. The integration of test data management also identifies the test cases

which cover requirements and those artifacts which have not yet been covered by tests. Changing or supplementing a requirement may, in turn, necessitate a change to a component or test case. Process reliability is ensured with the help of change identification and product life cycle management in PREEvision – life cycles enable management of work flows tuned to the specific organization. Within them, company-specific work flows or principles of operation such as agile processes are mapped and, using conditions and rules, safeguarded. In addition to change identification, change history and life cycle management, ticket management is also available for coordinated change and release processes.

Eliminating Disadvantages of the Integrative Approach

The increased complexity of the development process through the embedding of all data relevant to the E/E system and the high number of users and roles involved may be considered disadvantages of the integrative approach. User acceptance is also a challenge.

Requirement engineers, for example, can no longer work in a specialized tool which is purely custom tailored to their needs. These aspects were identified in PREEvision and solved using suitable mechanisms: Perspectives enable a special view of the system data for dedicated user groups and provide a user-specific interface (use case-driven GUI) which supports the respective work flow. Scopes reduce the visible data extract and only focus on the required artifacts – complexity is greatly reduced here **[Figure 5]**.

Visualizations ensure easy access to the system, facilitate comprehension and represent correlations in a transparent way. Examples here include technical diagrams for a wiring harness or network and behavioral diagrams like use case diagrams. The reuse of requirements, functions and other E/E artifacts accelerates development noticeably and ensures data consistency, so the recreation or copying of similar or identical artifacts is avoided. Last, but not least, validators and testing mechanisms ensure consistency and offer automated solutions for identified problem areas – as soon as data is entered.

Figure 4: PREEvision collaboration functions for developing requirements.

Figure 5: With the integrative approach of PREEvision, mechanisms like perspectives and scopes reduce complexity and increase both efficiency and user acceptance.

Summary and Outlook

A consistent tool in which users can access all the data relevant to them provides clarity and comprehension for all roles involved with system development. It makes the complexity of the overall E/E development process controllable and enables conflict-free and consistent cooperation – quickly, flexibly and efficiently. Current, detailed and always traceable requirements are an important key to the success of an E/E system, as their seamless integration prevents tool breakdown and enables the traceability of all data in an E/E system. PREEvision has been implementing this approach for years. It has proven itself in productive use and supports all E/E development in a consistent, model-based and future-proof way.

Author

Robert Rotter
Vector Informatik GmbH
Senior Product Management Engineer PREEvision

Originally published in "Elektronik automotive" magazine issue November 2017

Current Approaches for ECU Testing

What really matters

Further development of methods and tools for comprehensive and structured testing of ECUs is necessary not only for economic reasons. New highly complex technologies such as ADAS and autonomous driving also call for suitable testing strategies. This article presents current development activities and approaches as well as appropriate strategies by Vector.

A large portion of the value created in a modern vehicle is associated with the electronic and software systems. Malfunctions of electronics and software are accordingly high-risk and are not tolerable especially when used in safety-critical systems. That is why significant investments have been made in automated testing of these systems for many years. With each new technology, the question arises about the correct test methodology. Important factors here are the effectiveness, i.e., whether and how the test objectives are achieved, and the efficiency, i.e., optimization of costs and effort.

Model-based test design

In model-based testing, test sequences and data are generated from models. These models can either be a model of the ECU functionality or the test cases themselves. The first approach, thus the derivation of test cases from function models, should be viewed somewhat critically. If both the productive software and the corresponding test cases are generated from the same model, that promises large efficiency gains – the test practically drops out during development free of charge. But one should be aware of what is actually tested in such a constellation – in the extreme case, the code generator is merely being tested against the test case generator. Moreover, a function model largely contains the "good case" and thus describes how a device is supposed to function. The corresponding operating situations for the test of these functions can still be effectively derived from this model. However, it is difficult if not impossible to derive the many different error scenarios from the function model. But it is exactly these error scenarios that make the test so valuable since most problems don't arise in the "straightforward case".

For this reason, the Vector test solution works with a test model. This means that the test designer systematically models the situations in which the System Under Test must exhibit correct functioning. A graphical test notation enables a clear and transparent representation of the abstract test sequence **(Figure 1)**. The test cases are generated from this high-level description. Variants are visible in the diagram and are taken into account during test case generation. The test models connect the specification and implementation of the test in an abstract and easy-to-understand way. This will also facilitate the frequently neglected quality assurance measures for the tests themselves, such as reviews by developers. Abstraction enables use of the same test specifications on different test systems and the reuse of previously specified tests – both leads to significant savings.

Figure 1: vTESTstudio provides various graphical test notations for test modeling

Hardware-in-the-Loop Tests

Hardware-in-the-Loop (HiL) tests are rightly considered today as a key component in assuring fully integrated ECUs. During these tests, the ECU runs in a simulated environment that takes into account the activities of the ECU which forms a closed loop with the environment model. The necessary environment model and the real-time-capable simulation system often turn out to be correspondingly complex. However, unlike the engine and vehicle dynamics controllers, for which the HiL approach was originally developed, not all ECUs today are so complex and designed as closed-loop controllers. For testing most ECUs in the vehicle, ranging from external lights to power windows, it is fully sufficient to stimulate the inputs and observe the corresponding reactions at the outputs.

As we know, it is less expensive to eliminate errors the earlier they are found. Elaborate test automation, complex models, and expensive test systems generally tendto be performed only later in the development cycle.The necessary resources are available only during the explicitly identified test phases. An agile procedure during development would be desirable that achieves high quality gains with little investment.

The art in selecting test systems is in identifying the right degree of depth, effort, and usability. With the combination of CANoe software and VT System hardware, Vector offers a test system that is scalable from the simple test tool at the developer's work station to the highly-automated HiL environment in the test lab. The core idea of the VT System is that all hardware functions that are needed for the testing of ECUs are combined into a modular system that is seamlessly integrated in CANoe. The test hardware surrounds the inputs and outputs like an envelope and includes the power supply and network connections of an ECU or a subsystem **(Figure 2)**. The pin function corresponding to stimulation, measurement, load simulation, error injection, and switching between simulation and original sensors/actuators is possible at each pin. These functions are designed universally so that a test system that is set up once can be used for different ECUs.

Besides the network environment, the physical environment can also be simulated by corresponding MATLAB/Simulink models in CANoe. A closed hardware-in-the-loop simulation is just as possible as a simple manual stimulation without a complex model. CANoe offers the same flexibility for the test automation. The list of options range from the above-described graphically notated test models

Figure 2: The VT System forms an envelope around the ECU to be tested and operates all inputs and outputs.

and the programming in different languages (CAPL, .NET/C#) to the defining of simple test sequences in table form. With vTESTstudio, a modern authoring tool is available. The manual performance of tests using the graphical user interface of CANoe and user-created panels is also possible. It is crucial that a powerful test tool with manageable complexity is available for all phases and user groups (such as ECU and software developers, component, ECU, and system testers). The tool has the same basis in all cases but has a different expansion depth.

Virtualization and Software-in-the-Loop Tests

Tests in virtual environments promise an enormous saving potential for two reasons: First, tests are expensive in real-world conditions and with real ECUs, subsystems, and vehicles. Real prototypes are needed, which requires stimulation and measurement of all inputs and outputs and physical effects by corresponding hardware. Second, systems can be tested earlier in virtual environments because it is not necessary to wait on the corresponding hardware and the integration of the various components. In addition, it is often difficult to correct the hard-to-reproduce and thus often expensive errors in rare situations with real devices. However this can be easily done in a virtual environment.
Virtual systems that reproduce real-world conditions as exactly as possible are common. For example, Vector offers an ECU simulator for integration tests of software components (AUTOSAR SWC) that even works with original ECU code. For other applications, such as system tests, model-based approach is recommended. If the functionality of an SWC is being tested, the configuration of a complete AUTOSAR stack is cumbersome. Here, it should be possible to work at a higher abstraction level, that models only the time behavior of the data exchange between the components, for example. How the data are mapped onto specific networks is of no interest here. The concentration down to the essential aspects reduces the complexity and increases the efficiency.
Another aspect concerns the possible variations in the test. What good is it to have a close-to-reality virtual environment when this reality is always changing, such as when a newer basic software version is used. For effective protection, an environment that can be variable adjusted using many parameters can be much more valuable. This leads to robust software or helps to reliably adjust or identify a difficult constellation under error conditions. The Vector software vVIRTUALtarget pro uses this approach.

Testing of Service-Oriented Communication

In automotive networking and the communication between the vehicle and the environment, a change to a service-oriented communication must be realized. The driver for this was the introduction of Ethernet in the vehicle and the "over-the-air" interfaces to the server back-end. This represents a significant paradigm shift for the test systems even if vehicle diagnostics has long used service-oriented protocols. Thus, for example, a recorded communication cannot be simply played back to the system. In contrast to signal-oriented protocols, that generally transmit signals through a simple sequence of messages, an interaction between the ECU and test system is needed for service-oriented protocols, at a minimum for registering of services.
The vehicle systems used up to now are statically designed. This means that the communication structures are specified during development and no longer changed. Service-oriented architectures offer the option of registering services dynamically and distributing them among different nodes in the network. Test systems must stay abreast of this development. However, the dynamics should not have to be programmed in the test scripts. Nor should the complexity of the protocols be visible for the test creator. With parameterizable interaction layers and OEM-specific modules for transport protocols, network management, and the like, CANoe has long provided a basis for efficient simulation and test case development in the former, signal-oriented systems. A comparable abstraction layer for the service-oriented communication is an important part of the further-developed concept of the Vector tools.
Services will become an integral component of the simulation and test automation languages with these concepts. The tool takes care of the mapping of the services onto the network level, the various protocols, and transmission mechanisms. The test author can thus concentrate on the relevant elements; an explicit and thus laborious and error-prone programming of the intermediate levels is not needed. Possible interventions on the various levels are nevertheless possible in order to provoke errors in transmission behavior for the test, for example. If corresponding ECU interfaces are present, communication on the service level in the ECU can also be accessed using these concepts.

Software Test

Another paradigm shift that is partially associated with service-oriented architectures is currently taking place in the ECU software. Up to now, ECUs have been designed as embedded systems whose firmware is trimmed for efficient use of the hardware and that are flashed on the devices with a fixed functionality and as a complete program executable. Many functions will be implemented in central ECUs in the future that are heavily oriented to classic IT systems. Various programs that can be individually replaced work together on powerful hardware on an elaborate operating system like Linux. These central ECUs serve largely as compute nodes on which a large portion of the software runs. The control of peripherals, very time-critical routines, and probably also many safety-critical functions will still be performed by classic ECUs. AUTOSAR standardizes both systems and divides them into AUTOSAR classic and the new AUTOSAR adaptive, which is still in development.
These new software approaches for the automobile market raise many questions for the test systems and especially the test methodology. How can I ensure the quality of an ECU whose software is composed of different programs over its lifetime? The focus will shift in the future from the overall system of ECU hardware and software to the individual software functions. Where this is executed will play a subordinate role. Tests will therefore frequently take place where the software can be well observed – thus in virtual environments.

Dr. Stefan Krauß
Studied Computer Science and worked with Vector in Stuttgart/Germany since 2002 where he serves as Global Product Line Manager for the product line Tools for Networks and Distributed Systems.

Translation of a German publication in Automobil Elektronik issue 9-10/2016.

Image rights: Vector Informatik GmbH

Quickly Converting Test Benches Worldwide in Record Time

Record-Breaking

The numerous tests that are required before an ECU is released for production represent a considerable share of the development costs for the ECU. In addition, the effort required for conceptualizing test benches and programming test sequences has grown immensely in recent years. Of equal interest are the impressive cost and time savings that automotive supplier ZF TRW is realizing in its new generation of test benches. For example, the company was able to reduce its test bench setup times from many months to just a few weeks by using component-based test benches.

Years ago, anti-lock braking systems (ABS) were considered ambitious equipment for vehicles and were only offered beginning with upper mid-class vehicles. Today, complex extended traction control systems such as ESC (Electronic Stability Control) are now state-of-the-art. The evolution from ABS to ESC clearly illustrates how the complexity of automotive electronics continues to grow in all fields. In contrast to an ABS system, which is only active when the driver brakes, an ESC system operates practically autonomously. Accordingly, more sensors are needed to provide the system with acceleration data and information on transverse and longitudinal forces, torsional motion, etc. Therefore, newer ECUs must be able to process the data of increasing numbers of signals from their own sensors and data from connected bus systems.

Testing Effort Reached Pain Threshold

The products that automotive supplier ZF TRW develops at its Koblenz facility in Germany include controllers for ESC systems. The growing complexity of its developments is reflected not least of all in its increased testing effort. Both the numbers of ECUs and their extended functionality are resulting in test cost increases. Over the past ten years, the number of ECUs installed in a premium vehicle has grown from around 30 to as many as 200. The technical effort involved in the numerous test scenarios for testing every ECU type under all conceivable operating and environmental conditions is tremendous. A special ECU version is required for each vehicle model and each OEM. At ZF TRW, one ECU variant is implemented as an ESC system and one variant has an integrated parking brake as well.

Each ECU must pass numerous electrical, functional and mechanical, environmental in addition to EMC tests before product release. Simulations must recreate environmental parameters as well as mechanical and electromechanical stresses and in extreme situations. In temperature cycle durability tests, for example, the devices and PC-board assemblies are subjected to thermal shocks from -40 to +120 °C. Mechanical stresses can range from vibration tests with sinusoidal oscillations and noise characteristics to individual mechanical shocks with accelerations as high as 30 g. Moreover, sealing tests are performed in tests with salt spray fog, extremely fine sand and splash water, while high-pressure cleaning devices simulate engine washing.

Virtual ECU Environments

A precondition for these tests is that the ECU must be electrically and functionally active. However, representative tests and comprehensive diagnostics of ECU functionality are only possible if the test bench seamlessly emulates the electrical and electronic vehicle environment. This means that signals must be generated with realistic voltages and currents at all digital and analog inputs. This also applies to connected loads and to what is known as remaining bus simulation. That is because, in addition to CAN, other bus systems – such as FlexRay, LIN and MOST – have become well established in the automobile with numerous ECUs communicatinge simultaneously on multiple buses. These multibus systems further increase complexity and must also be simulated properly in the remaining bus simulations – and sometimes the functionality of gateways must be simulated as well.

Some time ago, the test bench engineers began their search for a more advanced and flexible test bench solution that could handle the growing complexity of ECU tests and available timeframes that would meet their future needs. The solution was to be implemented globally and uniformly at all ZF TRW business sites. Since then, the automotive supplier has started up a total of 32 of the new test benches in Germany, the USA, Czech Republic and China. Each test bench consists of a 19" cabinet with six rack-mounted units, so each stand can test six ECUs in parallel **(Figure 1)**. Each cabinet has a scanner, a touchscreen and a keyboard for user inputs, display of results and readout of status messages **(Figure 2)**. The individual rack-mounted units are supplied with energy from a central power supply that can supply electrical currents of up to 500 A for high-current for consumers and a nominal voltage of up to 30 V. These high power levels are required for tests of the ESC controllers, in which 580-Watt drive units are responsible for building up brake hydraulic pressure, and high ramp-up currents occur. Despite its high energy capabilities, it is neces-

Figure 1: Six rack-mounted units or subsystems, each of which represents a unit for testing one ECU. One test bench can test a total of six ECUs simultaneously.

Figure 2: User-friendly concept with touchscreen, scanner and keyboard simplifies operation of the test benches

Figure 3: The modular VN8900 network interface with integrated real-time computer plays a central role in the ECU's network communication.

sary to manage energy consumption during parallel tests to ensure that not all of the rack-mounted units demand maximum power simultaneously.

Focus on Intelligent Network Interface

Using the VN8900, a special hardware device from Vector, plays a central role in the solution used for the component-based test bench **(Figure 3)**. The ZF TRW test bench supplier, the company Smart Testsolutions GmbH – not to be confused with the manufacturer of small cars – integrates one VN8900 system per rack-mounted unit in its solution **(Figure 4)**. The VN8900 system is a modular network interface for FlexRay, CAN (FD), LIN, J1708 and K-Line bus systems that is equipped with a dedicated x86 real-time computer. Specially designed for applications with many parallel accesses to multiple bus channels, it offers high I/O performance with extra-ordinarily short reaction and response times and very minimal latencies. Along with variably configurable bus interfaces, the system also offers extension options for analog and digital inputs/outputs. This intelligent interface is easy to configure from a PC over USB or Ethernet.

Since the VN8900 systems was optimized for the use with Vector CANoe and CANalyzer simulation and analysis tools, the new test benches can easily and seamlessly be integrated into the tool chain at ZF TRW. This means that existing CANoe remaining bus simulations, which are available in the development departments, can be reused practically in tests with minimal modifications. These remaining bus simulations are already verified internally, so that no additional effort is needed for quality assurance when they are reused. This approach essentially eliminates all internal concerns from the outset, and it leads to noticeable workload relief for employees and especially for the project leader. The approach reduced changeover times drastically: before it could take up to eight months to retrofit an ECU test from OEM1 to OEM2, to a time period of just two to three weeks now. At the same time, there was a tremendous increase in flexibility in responding to late customer software changes shortly before the start of testing with modification times of less than one week.

Robust Test Sequences

Another special aspect of the VN8900 system is the option of automatic operation without requiring a connection to an operating or monitoring PC. The test engineers use CANoe at a normal PC workstation to create the remaining bus simulations and test sequences in the CAPL script language. They can then be simply loaded into the VN8900 system and executed there. Some of the benefits that ZF TRW realizes from this are greater stability and durability in long-term tests. Previously, the PC would sometimes crash after a couple of months in the long-term tests which took up to six months to complete. This was annoying, because the computer could then no longer provide any monitoring of the ongoing test process. Such problems have yet to occur so far with the new test benches.

Today, a strict separation is made between test controlling, visualization and communication with the ECU. The VN8900 system communicates with a Linux-based host computer from Smart Testsolutions via the LAN interface. In this case, a special real-time capable Ethernet/UDP protocol supplied by Vector, which is known as FDX (Fast Data Exchange), is used. By working closely together with Vector, ZF TRW was able to modify the FDX protocol to meet its own needs. In this framework, it implemented data exchange by the FIFO principle, for example. In turn, Vector got ideas from ZF TRW about ways to further develop the VN8900 system. FDX offers wide-ranging access to automatically running tests along with enabling starting and stopping of applications. It also offers such capabilities as reading out and clearing error codes, reading and saving XCP variables, influencing remaining bus simulations and much more.

Figure 4:
Block circuit diagram of a stand-alone subsystem; six of these units are installed per test bench.

Budget-Friendly Solution

For ZF TRW, extensive support of the necessary protocols was a key argument in favor of the Vector solution. Vector has specific solutions that support any of the commonly used protocols and OEM-specific automotive networks and diagnostic systems on the market. When customers turn to lower cost hardware components from other manufacturers, they often experience higher subsequent costs that offset any hardware savings when protocols need to be redeveloped for individual buses or for the XCP support for various OEMs that is often lacking.

Vector also advantageously resolved a licensing issue for customers, because users like ZF TRW essentially do not incur any additional costs for updates by using what is known as the CANoe stand-alone extended license. Any number of test bench applications can be created using just one CANoe license on a development PC. Updates of the VN8900 systems to newer CANoe stand-alone extended versions are always included. Even towards the end of the life cycle of the new test benches, the customer can continue to use the VN8900 devices meaningfully at the workplace instead of having them become unused dead capital.

Development Continues

With its new test benches, which were first used to test ESC controllers, the automotive supplier ZF TRW has a component-based test system that consists of six to seven modules which may be combined in various configurations. The company can rapidly adapt the test bench concept to tests for other types of ECUs, e.g. for airbag controllers or driver assistance systems. The drastically shortened changeover times and the speed with which managers can now respond to changes in requirements are very impressive. Future systems for autonomous driving are already envisioned. When testing the radar and camera systems, which are indispen-sible for these future systems, the new test bench concept will offer many opportunities for further exploiting its capabilities and streamlining potential.

Translation of a German publication in Elektronik automotive, issue 9-8/2015.

Image rights:
Figures 0–2: ZF TRW
Figures: 3–4 Vector Informatik GmbH

Links:
Homepage ZF TRW: www.trw.com
Homepage Vector: www.vector.com

Stefan Siefert-Gäde, ZF TRW
studied electrical engineering at the Technical College of Koblenz. In 2006, he began work at TRW Automotive in Koblenz. Since 2014 he has been coordinator for the global, strategic development of test equipment in the Global Electronics area.

Katja Hahmann, Vector
studied Electrical Engineering at the Technical University of Chemnitz. In 1997, she joined Vector Informatik GmbH in Stuttgart, where she is now group leader for CANoe application development in the Networks and Distributed Systems product line.

Network Tests for Everyone

With Minimum Configuration Effort for Maximum Testing Depth

Every development department of automotive electrical control units (ECUs) naturally thinks about how they can test their units quickly and efficiently. Since 2012, Daimler has been successfully using a special network test generator to create test configurations for network tests of engine ECUs for passenger cars among others. The tool, which was developed externally by Vector, is based on the simulation and test tool, CANoe. It covers comprehensive test cases from network management tests and gateway tests to monitoring of message counters and cycle times. Tedious adaptations and the programming or editing of test scripts are now a thing of the past.

The testing of ECU software always takes up a significant portion of the overall development process. Each new software version must run through a defined release procedure that includes numerous individual tests with different focal points. At the same time, the results must be thoroughly documented as proof of the software's degree of maturity and bug-free condition. The complexity of today's ECUs makes this process unmanageable without test automation. Still, test configurations and test scripts do not come into being on their own.

To make this process more efficient, Daimler developed the CANoe application CANSpector years ago for network tests of their passenger car engine ECUs. This application supported all tests for the network management (NM) of the Daimler network technology at the time with OSEK NM. Employees had to handle all other test scopes manually with the CANoe software, requiring them to create scripts, configurations, and the like and to perform measurements, generate reports and so forth.

In the current powertrain architecture, multiple engine ECUs and gateways are in development concurrently at Daimler. Because the automobile manufacturer uses platform software, the engine ECUs are similar in their software components for networking. The two development cycles based on the V-model per year with four software releases per V-cycle entail a considerable testing effort. The first software release is typically characterized by the most innovations so that the largest test scope also occurs then. If one considers the overall test effort for the networking software, approximately 40 percent is allotted to the tests of the network changes. The rest is allotted to general software tests based on requirements. The time needed for testing is significantly less for subsequent software releases.

CANSpector becomes COMSpector

With its new network topology, Daimler introduced an AUTOSAR-compliant network management. The lack of support for AUTOSAR standards, FlexRay, and new database formats were some of the reasons that the in-house tool underwent further development in 2011. In view of the extensive adaptation work and maintenance effort, the development department at the German Sindelfingen site decided to outsource the project and hired Vector to undertake the development and further development of the new tool. The new "COMSpector" network generator has a significantly expanded scope of functions and thus covers considerably more tests case and – like its predecessor tool – is based on CANoe. The latter provides numerous functions for automatic testing of ECUs via the TFS (Test Feature Set). The tool also handles residual bus simulations, plays back recorded trace data, and has a report generator for creating detailed test reports **(Figure 1)**. Automatic test sequences and network node simulations can be coded in the CAPL script language with C-like syntax. Because CANoe has long been used successfully within Daimler, COMSpector is integrated seamlessly into the existing tool chain.

In the current topology, the engine ECU and the central powertrain controller are connected to one another via a CAN direct connection. Furthermore, the engine ECU has a CAN bus for connecting sensors as well as the powertrain CAN bus that leads to, among other things, the transmission control module (TCM). The central powertrain controller (CPC) forms the interface between the powertrain and the vehicle and performs functions including gateway functions. Besides multiple CAN buses, there is also a FlexRay bus leading from the CPC. It serves, for example, the ESP, electric driving steering response, electric ignition switch, and radar ECUs for various assistance systems. The COMSpector tests cover all seven current engine types of Mercedes Benz passenger cars.

Sophisticated tests with a mouse

The basis process for creating a test case with the COMSpector tool looks something like this: The first step is to select the communication matrix. That can be an AUTOSAR ECU extract, an AUTOSAR system description, or a CANdb database (network or routing database). Using selection trees, lists, and buttons, the test engineer can now easily select the test nodes, scopes, and options needed for his test case using a mouse. At the end, the program generates an XML test module and a CAPL library. Up to this point, COMSpector works as a standalone tool fully independent of other software tools so that any employee can use it on his PC. The XML test module and CAPL library are then used on a computer with installed CANoe for creating the actual test configuration for the test system and can also be executed there **(Figure 1)**.

Figure 1: Structure of the Daimler test execution: process from the network test generation by COMSpector up to execution by CANoe

COMSpector supports both passive and active tests. Active tests use the CANoe functions for the remaining bus simulation and are always intended for an individual ECU. Through a remaining bus simulation, it is possible to simulate the missing environment almost completely for the isolated test object. This is the only way to realistically test the engine ECU or gateway in an early development stage before other network components exist. The simulation system simulates the send behavior of receive messages with the help of information from the communication matrices. Based on the interaction layer implemented in CANoe, the software is able to exactly reproduce the Daimler send model needed here.

Passive and active testing

In passive tests the test system assumes the role of an observer. In other words, it does not intervene in the events on the bus systems for the entire test period. Passive tests are thus typically well-suited for actual test drives. Because every network engineer at Daimler has an ECU at his desk, these tests are also possible directly at each workstation. There is constant testing there of the network traffic or certain messages for specific rules of behavior. In the first step, it is usually tested whether an ECU sends all messages in the correct time pattern and with the correct checksum and correct message counter. Missing messages may indicate an overload of the ECU itself or that the bus load is too high in general. Passive tests are possible both for individual buses and across buses.

For testing the routing behavior of the central powertrain controller, for example, the test system monitors the routing times, data consistency, and data sequence for all cyclic and spontaneous messages. Every error that occurs triggers an entry in the test report, such as when messages are lost or the permitted routing times (MaxRoutingTime) are exceeded. The update bit routing test follows a similar process. The gateway must always set the update bit when a certain signal has been received on the source bus so that the destination ECU can detect that the original ECU is sending. Signal changes are thus detected correctly and not concealed by the gateway under any circumstances. The test system documents every violation of these rules in the error log.

In further tests, message counters, checksums, and cycle times can be monitored in a similar way **(Figure 2)**. The parameters defined by the test engineer in COMSpector, such as MaxRoutingTime for gateway tests and MaxCycleTime/MinCycleTime for cycle time monitoring, are of vital importance for the evaluations of the test run. The last two parameters mentioned define the upper and lower time-related tolerance limits within which the relevant cyclic message must arrive in each case. Each violation of the cycle time tolerance limits triggers an error entry in the test log. Passive tests also generally function with logging data in offline mode. This means that recorded network communication from test drives can be subsequently tested at the workstation at any time.

Figure 2: COMSpector screenshot with passive tests

Figure 3: COMSpector screenshot with active gateway tests

In active tests, CANoe simulates the network communication of the network involved except for the ECU to be tested. The test system enters actively into the bus communication in order to test the defined rules of behavior **(Figure 3)**. The gateway routing tests take up a large test scope here. The message and signal contents are stimulated with different values (minimum, average, and maximum values). Each stimulation is tested one after the other in a separate test case. The routing behavior of the ECU is monitored during each test case. The following functions are tested here in particular: routing times, data consistency, and data sequence. The gateway routing tests also include the simulation of message failures and the transport protocol routing.

In the active tests of message counters, checksums, cycle times, and update bits, the corresponding signals of the test system are manipulated in such a way that an error entry is generated in the diagnostics error memory. The user then verifies the corresponding error entry via a diagnostic tool.

In addition to the predefined test sequences, interactive tests enable the engineer to select certain messages during operation and to manipulate them by interrupting the sending of messages or by automatic signal value changes, for example. The manipulations are made via control panels that are automatically available for each ECU.

Life tests in the vehicle

A special use case that is just as important as tests at the workstation are the tests with signal manipulations directly in the vehicle. CANoe is connected between the vehicle and the ECU to be tested and acts as a gateway by routing the data through one-to-one in both directions. The tester is then able to interactively select and manipulate particular messages during operation. He can block messages, change signal values, falsify checksums, freeze message counters, and much more. Because the test system maps all messages in the form of system variables, these can be easily displayed using the Symbol Explorer. A click in the context menu of a system variable opens the related control panel that permits the indicated manipulations.

Network management tests

The network management (NM) controls the network traffic on the bus and is mainly responsible for the waking up and shutting down of ECUs. An ECU with faulty network management could, for example, keep the network traffic in the vehicle permanently awake causing the car battery to discharge after a short time. This test should discover any problems of this type well in advance. The NM is tested for the various Daimler network topologies in predefined test sequences. Besides the ECU to be tested, other nodes are simulated on the NM for this. The Daimler NM tests currently support the CAN and FlexRay bus systems.

Small test setup with major impact

Daimler is especially pleased that the network tests can be implemented with a surprisingly small test setup. Besides a PC or laptop, a test engineer needs only a break-out box and one or two CAN or FlexRay interfaces from Vector. The break-out box with its handful of separated connections and cables represents, in effect, the minimum necessary

hardware configuration. Among other things, it generates not only a 12 V steady plus but also a switchable voltage from a 12 V infeed for simulating an ignition switch. One of the requirements of the automobile manufacturer was that the Vector tool would work without expensive additional hardware. In particular, Daimler wanted, if possible, to replace the very expensive and complex Hardware-in-the-Loop systems used for network testing, as the effort and costs associated with these were often disproportionate to the results achieved.

In a given time period and with the available personnel, COMSpector allows the creation and run through of many more ECM and CPC test configurations. This increases the test depth and, above all, benefits product quality. For example, while the engineers previously had to determine the cycle times for each message type, a passive COMSpector test configuration now covers that completely for all messages simultaneously. In general, numerous manual adaptations were previously required, and complicated cases meant that constant changes in the CAPL code were inevitable. Only well-trained testers were able to accomplish these tasks but even then only with significant preparation work. Today, even employees who are not proven test or network specialists can perform may simple testing tasks.

Daimler is now using COMSpector worldwide, including in the USA and in China. The master distribution list of the Daimler employee in charge contains approximately 20 recipients. Recently, an American colleague reiterated that he found the tool very helpful for his work. So it is no wonder that the development department at Sindelfingen, Germany, is receiving a lot of requests for extending the tool. Ideas and suggestions for the next COMSpector version are being collected and reviewed there.

Development continues

The network test generator COMSpector has become an indispensable component for testing worldwide at Daimler. It allows test configurations for passive and active network tests in the powertrain to be created quickly and without special expertise. In particular, the tool makes the laborious and time-consuming programming and adapting of test scripts unnecessary. The employees can concentrate their efforts instead on the actual development work and on quality improvements. It is now mandatory for many ECU and routing tests to be performed with COMSpector, according to the Daimler release procedure. The tester can easily store the finished report supplied by CANoe as proof of compliance with this requirement.

Daimler and Vector are currently working on integrating test capabilities for CAN FD and Automotive Ethernet in COMSpector.

B.Eng. Viola Misch
studied electrical engineering with emphasis on vehicle electronics and mechatronic systems at the German Ravensburg University of Cooperative Education. Since 2014 she has worked in passenger car development at Daimler AG. She is active in the field of powertrain topology and networking and oversees the COMSpector tool on the Daimler side.

Dipl.-Ing. Katja Hahmann
studied electrical engineering at the German Chemnitz University of Technology. She joined Vector Informatik in Stuttgart in 1997 where she serves as Group Leader of customer-specific CANoe applications and test systems in the Networks and Distributed Systems product line.

Translation of a German publication in Elektronik automotive, issue September 2016

Image rights:
Figures 0-3: Daimler AG

Integration of Virtual Hardware Prototypes in CANoe

Case Study
Fraunhofer Institute for Integrated Circuits (IIS)

The Customer

The Fraunhofer Institute for Integrated Circuits (IIS) is one of the most important German research institutes for the development of microelectronic systems and software. The scientists in the institute's department for Design Automation (EAS) in Dresden develop reliable methods and tools for designing complex electronic and mechatronic systems.

The Challenge

To master the complex design process for electronic systems in automotive technology.

In developing electronic components for automotive technology, many very diverse partners must work together – from the semiconductor manufacturer to the TIER1 and finally the OEM. In this work, the specifications of individual components pose an especially large challenge: The system supplier must create its system specification according to these requirements, and it needs – for validation – a hardware prototype that is not always available at this point in time.

The Solution

Integration of virtual prototypes based on SystemC/ SystemC-AMS in the CANoe platform

SystemC/SystemC-AMS is a system modeling language that is widely used in the semiconductor industry, which makes it possible to create abstract models of a system's various hardware and software components. Because these models are abstract they can be simulated very quickly. Such models can be compiled and supplied to partners and customers without revealing internal know-how. The AutoSUN project supported by the German Federal Ministry of Education and Research demonstrates how models supplied by TIER1 or TIER2 suppliers can be integrated in the CANoe platform used at the OEM. This involves using the CAPL interface and implementing time synchronization between the SystemC-AMS simulation time axis and the CANoe time. For this purpose, the SystemC-AMS model is linked together with the synchronization library to form a DLL. This enables sharing of the model after compiling, which protects IP.

The Advantages

Component specifications without hardware prototypes

> Very early on testing with the ability to integrate in the system
> Cost and time savings for hardware prototyping
> Control of the virtual prototype using CANoe is identical to control of the real hardware
> Connection of the virtual prototype with real hardware; for example, a virtual ECU can be connected to the bus system and tested early in the specification phase
> IP-protected model exchange
> Detection of concept or architecture problems

Figure: Exemplary simulation setup

Tips and tricks for the use of CAPL

CAPL is a programming language available in the software tools CANoe and CANalyzer. In three consecutive articles, CAPL fundamentals will be discussed as well as tips for all levels of user knowledge.

Authors

Marc Lobmeyer

Roman Marktl

Vector Informatik GmbH
Ingersheimer Str. 24
DE-70499 Stuttgart
Tel.: +49-711-80670-0
Fax: +49-711-80670-111

Links
www.vector.com

This first part focuses on the basics of CAPL. It is primarily intended for those who are new to this language; however, it also offers a few insights for well-informed users into the motivation behind individual CAPL constructs. The second part will discuss advanced functionalities of CAPL. Finally, the third part will address performance and memory needs and offers tips and tricks on using databases and associative arrays.

For over 20 years – it was initially used in CANalyzer for DOS – CAPL has been used to implement a broad bandwidth of tasks: from simple stimuli to the simulation of complex bus nodes. In the following, CANoe is illustratively named for the two products CANoe and CANalyzer. The goal of CAPL has always been to solve specific tasks as simply as possible. Typical tasks are reacting to received messages, checking and setting signal values and sending messages. A program should restrict itself to precisely these things and not require any additional overhead.

Many programming tasks that CANoe users typically perform might actually be as brief and trivial as the example presented below – of course many other tasks are not so trivial. That is why CAPL has been continually extended over the years to be a tool that can also solve complex tasks according to the principle "as simple as possible".

"CAPL" is an acronym for Communication Access Programming Language. The original focus on CAN has long been extended to all automotive bus systems such as LIN, Flexray, Most, J1587, as well as a few others like Arinc and CANopen.

As in many other languages, the syntax of CAPL is closely based on the syntax of the C language. Those who are familiar with C, C#, or various modern script languages will quickly be quite comfortable using CAPL. However, a few unique aspects distinguish a CAPL program from a C program:

CAPL programs are event-driven. This means that they consist of individual functions, each of which reacts to an event within the system under analysis: receipt of a message, change of a signal, expiration of a timer, or even a change in the environment. To react to the message "EngineState", for example, you would use: "On message EngineState" (Figure 1).

CAPL programs use specific databases for the concepts of the system under analysis. Messages and signals get their names there, and these names can be used directly in the program code. In Figure 1, they are the names "EngineState" for a message and "EngineSpeed" for a signal in this message.

CAPL programs do not give the user any pointer types to use. Right from the outset this excludes numerous potential programming errors and causes of program crashes, such as those that frequently happen in C programming. Nonetheless, since pointers – aside from their susceptibility to errors – also represent a very powerful concept, CAPL provides a substitute for some things, e.g. associative arrays as a substitute for dynamic memory.

One important property that CAPL shares with C should be mentioned: CAPL is always compiled, i.e. it is converted to efficiently executable and flexible machine code.

Example: a simple CAPL program

In Figure 1 a simple CAPL program is presented, which ▷

CAPL

CAPL is a procedural programming language similar to C, which was developed by Vector Informatik. The execution of program blocks is controlled by events. CAPL programs are developed and compiled in a dedicated browser. This makes it possible to access all of the objects contained in the database (messages, signals, environment variables) as well as system variables. In addition, CAPL provides many predefined functions that support working with the CANoe and CANalyzer development, testing and simulation tools.

```
1.    variables
2.    {
3.      const long kOFF   0;
4.      const long kON   1;
5.    }
6.
7.    on message EngineState {
8.      $sysvar::Engine::EngineSpeedDsp@this = this.EngineSpeed / 1000.0;
9.    }
10.
11.   on message LightState {
12.     if (this.dir == RX) {
13.       SetLightDsp(this.HeadLight, this.FlashLight);
14.     } else {
15.       write("Error: LightState TX received by node %NODE_NAME%");
16.     }
17.   }
18.
19.   SetLightDsp (long headLight, long hazardFlasher) {
20.     long tmpLightDsp;
21.
22.     tmpLightDsp   0;
23.     if(headLight == kON)
24.       tmpLightDsp = 4;
25.     if(hazardFlasher == kON)
26.       tmp  gbLDsp += 3;
27.     $sysvar::Lights::LightDisplay = tmpLightDsp;
28.   }
```

Figure 1: A simple example of a CAPL program

performs one of the basic tasks of a bus monitoring tool: it listens to traffic on the bus and prepares a couple of events on the bus for observation/monitoring by the user. This is a shortened, sample CANoe program: "Display.can" from the sample "Easy.cfg". In the following, first the overall functionality is briefly summarized, and then the individual sections are described in more detail.

Task description

The task is to observe a CAN network whose elements – e.g. bus nodes, messages and transported signals – are described in a database. When the "EngineState" message is received, then the Engine-Speed signal it contains is conditioned for display on a display panel, and it is routed to the panel. When the "LightState" message is received, the "HeadLight" and "FlashLight" signals it contains are conditioned for display on a panel, and they are routed to the panel for graphic display.

Description of the program

The line numbers are not part of the CAPL program and are only inserted here to make it easier to reference individual lines or sections. To achieve the most compact representation possible, opening brackets were not placed on a separate line. In a CAPL program, it is possible to define global variables and constants. This is done in the "variables" section (lines 1 to 5). These constants and variables are globally defined for this program: they can be used anywhere in the program, but not in other programs within the same CANoe application. The other sections define reactions to events (lines 7 to 17) and an auxiliary function (lines 19 to 28).

Lines 7 to 9 show a minimal form of a message event procedure. This function is called whenever this message has been transmitted on the bus. In reference to CAN, this means that the precise time point is the TX or RX interrupt of the CAN controller, i.e. immediately after correct transmission of the message. The message that triggers the call of the particular function is referenced by "this" syntax.

In line 8, the value of the "EngineSpeed" signal is read out from the received message ("this") and is assigned to a system variable with a conversion (/1000.0).

Lines 11 to 17 show a message event procedure for the "LightState" message, which transmits the information for a turn signal flasher. Its processing is similar to that of the "EngineState" message with the following unique aspects: In line 12, the direction flag (.dir) is now checked in the message ("this") that has been just transmitted. Only received messages should be considered in this program (value RX), because a message sent by the node itself would also trigger an event procedure (value TX). In this case, an error message would be output in line 15.

Since conditioning of the signal for display on the user interface (a panel on which different states are shown by different bitmaps) is somewhat more complex, its implementation is outsourced to a separate function: In line 13, "SetLightDsp" is called with the two message signals that are needed as parameters.

Finally, lines 19 to 28 define a separate function, which writes different values to the system variable "LightDisplay" in the "Lights" name space according to the value of the transmitted signal. In this demo configuration, this variable then selects the appropriate bitmap on a display panel. ◀

Tips and tricks for the use of CAPL (part 2)

The first part of this series of articles addressed fundamental concepts of the CAPL programming language. This second part explains the time behavior of event procedures.

Authors

Marc Lobmeyer

Roman Marktl

Vector Informatik GmbH
Ingersheimer Str. 24
DE-70499 Stuttgart
Tel.: +49-711-80670-0
Fax: +49-711-80670-111

Link
www.vector.com

CAN Newsletter (print)
Tips and tricks for the use of CAPL (part 1)

The second part also offers tips for all types of users so that they can work more effectively with CAPL in the areas of "generic programming" and "conditional compiling."

Execution model

A key difference between CAPL and C or C++ relates to when and how program elements are called. In C, for example, all processing sequences begin with the central start function *main()*. In CAPL, on the other hand, a program contains an entire assortment of procedures of equal standing, each of which reacts to external events:

- Triggered by the system: These events include those that are useful for initializing and post-processing the measurement run: on *preStart, on start, on preStop* and *on stopMeasurement*, as well as the time control and keyboard events *on timer* and *on key*.
- Triggered by bus communication: There are many different types of event procedures that react to bus events such as those related to communication or error handling, and they are very dependent on the bus type. Examples of these are *on message* and *on busOff* in CAN and *on frFrame* and *on frStartCycle* in FlexRay.
- Triggered by access to a *Value Object*: Such objects include system and environment variables that are globally available in CANoe and CANalyzer as well as signal values that represent a data interpretation of the bus communication. Special databases perform the interpretation. Part 3 of this series will address this concept.

Event procedures are atomic: The simulation model of CANoe is event oriented. In event procedures, CANoe executes all actions simultane ously from the model perspective, namely at the point of in time of the triggering event. The actual computation time on a real PC is ignored.

Simulation time and time stamp: However, a real event generated by the PC, such as a bus output by *output()*, gets a time stamp of the real-time clock. The sequence and time points of these events can be influenced by bus protocols, driver, and hardware properties.

On a simulated bus, some of the mentioned influencing parameters are eliminated. In this case, bus events are initiated simultaneously; in the case of CAN, for example, this leads to a dependable arbitration of multiple messages that are output by *output()*.

Updating system variables: Users can also use CAPL to modify environment or system variables that are visible outside of the program. CAPL does not propagate value changes to a variable until after the current event processing is finished, but with the same time of the just handled event. A read access within the current procedure always returns the old value even if the variable appears to be set to a new value within the same procedure. The advantage is that only one value change occurs at a single point in time.

The execution model is situation dependent: There are many ways to use CAPL in CANoe and CANalyzer, and so the execution model varies somewhat, too: The simulation nodes of a CANoe simulation are in parallel on the bus. Hence, they are completely independent from each other. Triggered events are always dispatched to all programs. In contrast, nodes in the measurement setup and in CANalyzer are processed in sequential order: Each node passes its output to the next. Incoming events must be passed to the next node explicitly for further processing. The procedures *on ** and *on [*]* are provided for this purpose.

Another type of program is a test program whose test procedures can wait for external events. CAPL resumes execution with the simulation time of such events. In contrast, waiting in normal event procedures stalls the entire simulation system. This is a frequent source of errors when CAPL is used. It is therefore inadvisable to use a busy-wait or wait command in an external DLL.

Efficient programming in CAPL

The preprocessor is a powerful tool in the C language, but it can also lead to confusion and consequently to errors. Therefore, only a subset of the well-known preprocessor directives in C is offered in CAPL with comparable semantics.

#include: Include files contain arbitrary but complete sections of a CAPL program: *includes*, *variables* and *procedures*. In contrast to C, the text of an include file is not simply inserted into the CAPL file, rather the sections. All sections of the included file apply to the entire parent CAPL file "as if" they were contained in that file. The sequence of sections is irrelevant in CAPL anyways. This means the compiler reports any duplicate symbols as an error. Moreover, code and data from included and parent files may use each other mutually. One exception to the just stated prohibition of duplicate symbols is that *on start, on preStart, on preStop* and *on stopMeasurement* may coexist in both the included file and the parent file. In these functions, the code is executed sequentially: first the code from the included file and then the code from the parent file. This means that the Include files are used to perform three tasks: declare data types, define variables and provide an (inline) function library.

#pragma library: CAPL programs can use Windows DLLs created in other languages, as long as they implement a suitable CAPL DLL interface. These DLLs can be directly linked with the directive *#pragma library("capldll.dll")*.

Macros: In CAPL, there are a number of predefined macros that are available to users for use in the code or for conditional compiling. Macros for use in the code can be used anywhere in the code without restriction. In contrast to C, macros may be used freely within string constants, identifiers for variables, and function names. They always begin and end with a % character, and they are primarily used to write generic programs.

Available code macros include the node name, index of the current channel, name of the current network and the type of bus being used. The code can access the name of the containing file with *%FILE_NAME%*, or it can access the name of the program file currently being compiled with *%BASE_FILE_NAME%*. In the case of Include files, the latter is the parent file. Here are two simple examples:

```
write("The node name"
" is %NODE_NAME%");
@Ch%CHANNEL% = 1;
```

There is a separate set of predefined macros for the conditional compiling of code sections. They are *#if, #else, #elif* or *#endif*. Within a program, they allow distinguishing between the program types simulation node, measurement node and test program as well as the CANoe version that is used. Here is an example that uses a *#pragma message*:

```
#if (TOOL_MAJOR_VERSION
== 7 && TOOL_MINOR_VERSION
== 5 && TOOL_SERVICE_PACK
< 2) || CANALYZER
#pragma message("This
program needs at least
CANoe 7.5 SP 3")
#endif
```

#pragma message: The *#pragma* message directive lets users output their own message during the compiling process, e.g. the version number of the currently compiling CAPL program. It appears together with the other messages, warnings, errors, and general messages of the compiler.

Tips and tricks for the use of CAPL (part 3)

The third and final part of this series presents tips and tricks for advanced users. Topics include associative arrays, performance, memory needs, and other database access options.

Authors

Marc Lobmeyer

Roman Marktl

Vector Informatik GmbH
Ingersheimer Str. 24
DE-70499 Stuttgart
Tel.: +49-711-80670-0
Fax: +49-711-80670-111

Link
www.vector.com

CAN Newsletter (print)
Tips and tricks for the use of CAPL (part 1)

Tips and tricks for the use of CAPL (part 2)

Unlike languages such as C, CAPL does not support any pointer objects as a reference data type and therefore has no dynamic memory management. This makes CAPL very robust, and therefore well-suited for runtime environments that are short on memory and difficult to debug. In particular, CANoe's "CAPL-on-Board" feature benefits from this; in order to improve real-time behavior, it executes programs directly on certain hardware bus interfaces. Having said that, memory is seldom in short supply in the Windows' runtime environment. Therefore in this runtime environment CAPL offers associative arrays that can be used to store data even if the amount of data to be stored is unknown at the program start. Associative arrays are containers which are equivalent to maps or dynamic arrays of other programming languages. Internally, CAPL uses an efficient hash table for these arrays. Consequently, these special arrays enable saving bus messages or measurement values, even if it is unknown in advance which messages or how many measurement values will occur.

In CAPL, associative arrays are declared as simple arrays, but with a key type instead of the otherwise usual size entry. Two examples of associative arrays:

```
long lastTime [long];
char[30] translate[ char[] ];
```

The variable *lastTime* is an array that maps *long* keys to *long* values, while *translate* maps *string* keys (without length limitation!) to *string* values up to 30 characters. The following example uses *lastTime* to store a time value for each message ID occurring on the CAN network:

```
on message CAN1.*{
    lastTime [this.id]
      = this.time;
  }
```

To enhance the user's experience, CAPL provides the following list of methods for associative array variables using dot notation:

- *ContainsKey* queries whether a specific key is already contained;
- *Size* returns the number of contained keys;
- *Remove* removes one key from the associative array;
- *Clear* fully empties an associative array.

In fact, *Remove* and *Clear* free up memory.

Finally, there is a special form of the *for* instruction for associative arrays. This form iterates over all keys actually contained in *lastTime:*

```
for (long key: lastTime)
  {[…]} …
```

Access to databases

Part 1 of this article series already illustrated the primary use of bus-specific databases in CAPL: they make it possible to introduce names for messages and signals. From a programming perspective, the complicated aspect of signals is that they are usually tightly packed in the data payload of messages for efficiency reasons. Therefore, signals generally exhibit arbitrary bit lengths and positions within the data payload of a message. They can also be stored in either Intel or Motorola format.

Symbol-based access via a signal name relieves the CAPL user of all of these details. In the case of reading or setting a signal value, the CAPL compiler automatically accounts for the signal's precise bit pattern that may include masking, swapping and shifting the bits.

To enhance user friendliness, other definable objects in the database may improve the linguistics of CAPL programming. For example, symbolic value tables may be associated with signals to use plain text names for signal value states. Furthermore, authors of a database have the freedom to define other attribute objects and to use them in the program code.

CAPL is able to use database objects directly based on their symbolic names. However, sometimes the potential objects of interest are not known at the time of program implementation. Therefore, the CAPL user may dynamically access the symbolic names and properties such as message names and identifiers transmitted by a network node. A brief example:

```
message * m;
int i, mx;
mx=elcount(aNet::aNode.Tx);
for (i = 0; i < mx; ++i)
{
  m.id=aNet::aNode.TX[i];
  write(DBLookup(m).Name);
}
```


These symbolic access methods allow the user to implement generic programs – together with the previously introduced associative arrays.

Performance

Most CAPL programs must meet non-trivial real-time conditions. The execution model of a node simulated with CAPL even follows the model concept that CAPL programs can be executed at any speed (see part 2 of this series of articles). To adequately approach this ideal, CAPL programs are compiled, i.e. they are compiled into the machine language of the specific executing microprocessor. Moreover, optimized code sequences are used for the often complex access to signals. Below are a few tips on how the user can affect performance.

writeEx(): the *write* function is used to output specific texts to the Write window in CANoe and CANalyzer. As an alternative, the *writeEx* function is available for outputting larger quantities of data. For one, it can be used to write directly to the Trace window or to a log file. The text output generated by *writeEx* is in all respects treated like a bus event, including the high priority processing and synchronizing the time stamps with real bus events.

Event procedures: a CAPL program consists of a combination of procedures that react to events. Some of these events may occur very frequently. Therefore, a program's performance is significantly better if only those events get processed, which are concerned. For example, if the user is only interested in those Flexray *slots* that contain a specific signal, it is more efficient to define `on frSlot signalname` than `on frSlot *`.

Signal edges: there are two event procedure versions for signals and system variables. *on signal_update* and *on sysvar_update* are called with each write access to the specific data objects, even if the object's value has not changed at all. By contrast, *on signal_change* (*on signal* in short) and *on sysvar_change* (*on sysvar* in short) offer a performance advantage if only signal edges are to be handled. Those event procedures are optimized to trigger on value changes only.

Memory needs

Unlike most block-oriented languages, such as C, all locally defined variables in CAPL are static by default. This means that they are all created at the program start, and memory used to store these variables is not freed until the end of the program. Consequently, CAPL may require a surprisingly large amount of memory if many event procedures define the same type of large variables, which they could actually share. An example:

```
testcase test789()
{
  char outBuffer[1024];
  [..]
```

There are CAPL programs with thousands of such test procedures, of which only one may be executed at any given time. Rather than defining a large local variable of the same type in each event procedure, defining the large variable once globally in the Variables section utilizes a lot less memory.

Another inadvisable practice is to create very large arrays, e.g. to store event data under the respective message IDs. An extended ID in CAN comprises 29 bits, so it can assume over 500 million values. To define an array for this purpose would be a waste of memory. In such cases, it is better to use associative arrays as described above. Although associative arrays need somewhat more memory for each key that is actually used, they do not need any memory for keys that are not used.

Useful, relatively unknown features

CAPL offers a number of less familiar and mainly newer features:

Structs can be used to define structures, similar to the approach in C. Together with copying operations, which can also convert Intel and Motorola formats within a *struct*, they represent a flexible method for data conversion.

When CAPL functions are called, the user has the option of passing reference parameters in addition to value parameters. Reference parameters make it possible to return more than one result value from a function. Reference parameters can also be used within CAPL–DLLs.

CAPL programs should also not crash in case of faulty usage. On one hand, this robustness is attained by the language structure, since there are no general pointers. On the other hand, stability is improved by automatic runtime checks of array limits, stack limits and the necessary computing time.

A separate command-line version of the compiler is available. This version is very helpful in automating sequences in script languages.

Concluding Remarks

This series of articles has introduced CAPL as an example of a problem-oriented programming language. The familiar C language syntax of CAPL simplifies the user's learning curve. Specific symbolic databases and concepts for using CAPL in simulation, emulation, and testing of fieldbus nodes support the application domains. Vector is carefully and continually extending the language in a way that maintains compatibility with previous versions while cultivating new application areas.

CAPL

CAPL is a procedural programming language similar to C, which was developed by Vector Informatik. The execution of program blocks is controlled by events. CAPL programs are developed and compiled in a dedicated browser. This makes it possible to access all of the objects contained in the database (messages, signals, environment variables) as well as system variables. In addition, CAPL provides many predefined functions that support working with the CANoe and CANalyzer development, testing and simulation tools.

Quick Paths to a Comprehensive Remaining Bus Simulation

Create Virtual Networks without Programming Expertise

A key task in developing ECUs is to create the remaining bus simulation. It ensures that a functional environment is available to the ECU, without which comprehensive tests could hardly be realized. The challenge for developers is to quickly generate a realistic remaining bus simulation that considers relevant constraints. With the right tools, it is possible to create remaining bus simulations without programming – essentially by drag & drop operations (Figure 1).

The goals of a remaining bus simulation range from simulating individual network nodes to simulating entire networks. From a technical perspective, the goal is to simulate an ECU's infrastructure over all layers of the OSI model. This requires modeling from the communication layer to the network management layer, transport layer and finally the application layer. The aspects and task areas to be considered in the implementation are just as varied. They might include selecting suitable hardware for the CAN, LIN, FlexRay and MOST bus interfaces and simulating ECU functionalities via MATLAB/Simulink models on the upper levels. The simulated remaining bus must receive messages from the system under test, dynamically react to them and send results in the form of messages back to the ECU (stimulation). In the final analysis, the remaining bus simulation is simply a means to an end. It is always embedded in a high-performance development and test system, which evaluates the obtained test results and prepares them for documentation. An example of this is the CANoe simulation and test system from Vector.

Configuring instead of Programming

Modern test and simulation systems should support different types of generation (automatic, manual, programmed) for the remaining bus as well as differing complexities of individual tests. For example, it makes a large difference whether a developer needs a small simulation for testing just one or a few system functions, or whether comprehensive and detailed full tests are needed in the final phase of a product development. The primary source of data for remaining bus simulations is the communication data of the various bus systems that is stored in DBC (CAN), LDF (LIN) or FIBEX (FlexRay) databases or in function catalogs (MOST). Diagnostic descriptions such as CDD and ODX are also incorporated. It is also possible to source all data from a single database such as the AUTOSAR system description.

Figure 1: Remaining bus simulation enables testing of simulated nodes in a network with real nodes.

With the help of a suitable configuration assistant and these databases, developers today are able to create remaining bus simulations in just a few configuration steps or even generate them automatically. Programming know-how is no longer an absolute necessity. Project participants simply add communication databases to the simulation setup by drag and drop operation. After this is done, the detailed behavior of individual network nodes can be modified. Multiple bus systems may be assigned to a network node, which is important when comes to simulating gateways and more challenging ECUs.

In remaining bus simulation, key roles are played by consideration of OEM-specific metadata and send models. The interaction layer comes into play here, since it represents the defined sending behavior for the relevant OEM. The interaction layer correctly represents protection mechanisms – such as a message counter and CRC computations – for each message type according to the send model. The interaction layer converts the signal-based access of higher applications to the message-based transmission method of the bus system. Another important aspect is support of different types of network management (NM) such as OSEK-NM or the bus-specific AUTOSAR-NM.

Figure 2: Standard panels let users interactively modify signal values.

Stimulation by Control Panels and Run Sequences

Applications on the upper OSI layer include signal panels and signal generators, for example. They are indispensible aids in simulating user activities and dynamic processes. They contain virtual switches, buttons and display instruments that can be used to conveniently input spontaneous operating actions on the computer screen such as activation of a turn signal, windshield wiper or window lift. They also show user system parameters and enable specific modification of signals and variables during the test runs. Therefore, a state-of-the-art test environment offers a suitable panel editor as a standard feature to create customized panels with user control and to display instruments or standard panels **(Figure 2)**, which are configured from database information. In automatically generating simulations and panels or in making signal assignments smoothly, the following is true: The more detailed the relevant networks, messages and attributes are described in the associated database, the more precise are the models created by the generators. Preconfigured panels benefit from all available information, such as signal descriptions, specified value ranges or symbolic identifiers.

Panels do indeed permit spontaneous and flexible manipulation of signals, but they are not capable of executing test sequences, user inputs and other events reproducibly. This is the domain of classic programming. Nonetheless, users often do not want to have to utilize powerful programming tools for every project. Therefore, a welcome alternative for test engineers is the option of creating test sequences in the form of a table **(Figure 3)**. This table simply consists of commands to be selected, which perfectly fulfill their tasks in many different test applications. They are used to set values, define wait times, modify signals and trigger events. This approach does not require any special knowledge, nor does it require great training effort, and it quickly leads to the desired results with small reproducible tests.

A Precise Fit for Challenging Tests and Simulations

Even large test scenarios no longer need to be programmed by a classic approach today, if a simulation and test system offers power-ful support for them. In standard cases, for example, there are prepared test patterns that offer support for all steps: from setting up the stimulation and evaluation functions to results analysis, HTML documentation and preparation of graphic output. Details of interest can be shown in graphic evaluation windows. For example, color highlighting in CANoe's State Tracker lets users quickly

Figure 3: Graphic creation of command sequences for stimulation and testing with the Visual Sequencer does not require any programming knowledge.

detect states, events and state changes **(Figure 4)**. This type of representation is realized by an Art Logic Analyzer for automotive networks or ECUs.

As already suggested, there are hardly any upper limits to the conceivable complexity that can be handled in a remaining bus simulation. To fulfill the many different requirements individually, a test and simulation system should exhibit such properties as scalability and modularity. In addition to broad support of bus interfaces, intelligent hardware interfaces – e.g. those with their own computing power – are increasingly gaining in importance, because of growing real-time requirements. They are capable of executing time-critical processes. Another challenge is to generate specific errors in the form of invalid messages, corrupt useful data or disturbances to the bus physics, with the goal of studying ECU reactions to such situations. A modern test and simulation tool offers these capabilities via special stress functions implemented in software and hardware.

Remaining Bus Simulation with CANoe

A comprehensive test and simulation tool, such as CANoe from Vector, masters all aspects of the remaining bus simulation for test setups ranging from simple to comprehensive. In particular, it lets users create the necessary simulation models rapidly, either manually or automatically, in just a few configuration steps and without the need for programming. The tool should be able to process all commonly used data formats, diagnostic descriptions and metadata and support the automotive industry with numerous OEM-specific packages. Numerous options for accessing messages, signals and symbolic identifiers simplify the work of ECU developers. Stimulation or flow control can be implemented by using signal panels, the Visual Sequencer **(Figure 3)**, higher-level programming languages via the .NET API or the signal-based CAPL language. For complex test setups, the integrated Test Feature Set provides prepared test patterns for stimulation, evaluation, results analysis and HTML documentation. High-level features, such as integration of MATLAB/Simulink models or expandability with supplemental hardware that simulates connected sensors and actuators, enable functionality that extends to the area of mid-size HiLs (Hardware-in-the-Loop).

Outlook: Remaining Bus Simulation and Electric Mobility

In the future, the theme of remaining bus simulation will also gain in significance in the area of electric mobility. Fueled by the dynamic development in this area in recent years, efforts here are focused on network management with simulation of partial network operation. That is because parking periods for electric vehicles are of course simultaneously charging phases, in which certain ECUs need to be reliably switched off, while other components should remain active for charging and monitoring. The fact that the topic of partial network operation was a priority in AUTOSAR 4.0 compared to AUTOSAR 3.2.1 underscores the current importance of electric mobility.

Figure 4:
The State Tracker graphically depicts system states and discrete values over a time axis.

Translation of a German publication in Automobil Elektronik, issue 3/2012

Image rights:
Vector Informatik GmbH

Links:
Home page Vector: www.vector.com
Product Information CANoe: www.vector.com/canoe

Stefan Albrecht
has been employed at Vector Informatik since 2003 and is currently working as a product manager in the central product management area for CANoe/CANalyzer.

Peter Decker
has been employed at Vector Informatik since 2002 and is currently working as a product manager in the multibus development area for CANoe/CANalyzer.

Hardware Simulation for the Challenging Unimog Tire Pressure Control System

Time Savings and New Options by ECU Tests on the Model

Increased traction, minimized fuel consumption and avoidance of site damage: These are good reasons to always drive off-road utility vehicles with tire pressure adjusted for specific conditions. Developers at Daimler AG used a new type of test system for realistic tests of the next generation tire pressure control system for off-road Unimog vehicles. The test system simulates all sensors and actuators of an ECU's environment and enables comprehensive HIL tests, including simulation of fault situations in the environment's hardware and wiring.

The business site at Wörth am Rhein is home to one of the largest commercial truck manufacturers in Europe. Besides producing the well-known Mercedes-Benz Atego, Axor and Actros trucks, Daimler AG also develops and manufactures special purpose vehicles such as the Unimog, Econic and Zetros.

The Unimog U3000/4000/U5000 product line achieves extreme off-road mobility with portal axles, coil springs, manually selectable all-wheel drive and thrust-tube technology; its performance capabilities include a maximum fording depth of 1.20 m **(Figure 1)**. Low-emitting diesel engines with power outputs ranging from 110 kW (150 PS) to 160 kW (218 PS) are used for the drives. One of the special features Mercedes Benz offers in its program is the "tire control" option. This is an electro-pneumatic system for inflating and deflating the tires that lets the driver modify tire pressure right from the cab.

While high tire pressure ensures low rolling resistance and fuel consumption on pavement, other properties are preferred in off-road driving. For example, when driving on a wet field reducing the tire pressure also reduces tire contact pressure, so that no appreciable site damage occurs **(Figure 2)**. When it comes to available traction, there is a direct relationship between soil composition and the tire pressure used. By reducing tire pressure off-road you can easily double available traction. This could be a crucial factor in determining a task's success or failure. In addition, low tire pressure can contribute to better self-cleaning of the tires, because of the tire's greater deformation.

Model Based Development of the Tire Pressure Control System

The electronic concept that was successfully applied to the Unimog series is currently undergoing advanced development to meet performance and operating convenience requirements. Advanced development of the next generation tire pressure control system involves a model-based development application on a new high-performance hardware platform. For the driver, this means increased operating convenience: In Automatic mode, the operator simply selects the terrain type, and the system automatically ensures correct pressure in all tires via a 4-channel pneumatic system.

PC-Based MIL Tests

In developing the new tire pressure control system, first a model of the new pneumatic system ("plant model"), including the tires, was created and verified based on measured data and design documents. This was followed by initial implementations of the application model. Daimler hired the company ITK Engineering to create the MATLAB/Simulink models. The models realistically simulate the dynamic behavior of the real system. Pressure sensors continually acquire the pressure of individual tires, which are always interconnected with the pneumatic system via a central air channel in the portal axles. Another sensor serves as a reference for the system and automatically verifies the sensors that have fixed assignments. Also included in the model are momentary pressures, compressor power and channel cross-sectional areas, etc. The relationships, including effective time constants when increasing and reducing tire pressure, are computed precisely in the model. As a result, PC-based Model-in-the-Loop tests (MIL tests) are already possible in this development phase.

Simulink Models in SIL Tests

The model-based approach is also taken in subsequent development phases. By integrating the models in the CANoe simulation and test tool from Vector, model behavior is tested in conjunction with bus communication. Using the CANoe blockset for MATLAB/Simulink, the model interfaces are connected directly to bus signals, system variables or environment variables. The Simulink® Real-Time Workshop® generates a Windows® DLL that is loaded in the CANoe simulation environment. The CANoe Interaction Layer handles sending of messages according to the send behavior stored in the database. Finally, the test sequences are created and automatically executed with the CANoe Test Feature Set.

This is followed by initial tests in the vehicle. Previously, these tests were not possible until the availability of prototype ECUs or so-called Rapid Prototyping Platforms. When CANoe is used with the integrated application model, this enables in-vehicle testing of the operation and display concept without requiring a special hardware setup. CANoe is responsible for CAN communication with the vehicle. A sensor/actuator module, in this case the current level ECU, makes I/O functionality available in the vehicle over CAN. Test sequences that have already been created can be reused for this purpose. The first Software-in-the-Loop tests (SIL tests) are executed early in a phase in which requirements have not all been fully described yet.

The Path to the Component HIL Test Bench

Since the developers would not always have access to a test vehicle, Daimler decided to procure a component test bench based on the VT System test hardware from Vector. The system is tailored to the needs of the automotive industry. It is a modularly configurable and powerful test system for uses ranging from a small bench-top setup to a large HIL system in the laboratory. In the VT System, the focus is on simulating the sensors and actuators connected to the ECUs as well as simulating potential error situations such as short circuits, overvoltages and undervoltages. The system is set up modularly with various VT modules for load simulation, measurement and stimulation. Another important factor in the decision to use this hardware was that it would be optimally integrated in the CANoe software system already in use at Daimler for many years. In particular, direct integration of these modules in CANoe

Figure 1: All-terrain mobility is the primary feature of the Unimog

Figure 2: Tire pressure control while driving is possible, e.g. to avoid site damage

would make it easy to perform later extensions or modifications to the system for new projects. The test system is connected to the test computer via its Ethernet port, utilizing the real-time capable Ethernet protocol EtherCAT®. CANoe permits access to all parameters of the VT System and is the tool for test automation **(Figure 3)**.
For HIL tests on the real ECU, the plant model is integrated in CANoe. Another advantage of using models is revealed here: Any desired initial states may be produced essentially "at the press of a button". In the plant model, this relates in particular to the pressure in the tires and in the pressurized reservoir. That is because in tests on real vehicles it may take 20 minutes or so until 4 flat tires are filled back to their specified operating pressure when a vehicle air compressor is used. The laboratory test system, on the other hand, is capable of doing this immediately. It addresses parameters of the model directly and represents values such as pressure curves graphically.

VT System as HIL Test Bench Component

The plant model is integrated in the CANoe Simulation via the CANoe Blockset - the real ECU is being tested here. All ECU inputs and outputs are connected to the relevant bus signals or hardware channels of the VT System. Communication between the CANoe Simulation and the Simulink models occurs directly via a signal interface or via CANoe environment and system variables. As a standard feature, the ECUs check their inputs and outputs to verify that the relevant components are actually connected and that sensor data or termination resistors are plausible. Therefore,

Figure 3: Layout of the component HIL test bench

Figure 4: Modular setup of the VT System as a component HIL system

for meaningful tests, either the original sensors and actuators are necessary, or else the ECU's environment must be simulated as realistically as possible. However, because reasonable test automation with original components involves enormous cost and effort, e.g. when controls are operated by robotics, simulation of the environment is generally preferable. The VT System is now responsible for simulating the environment hardware. The VT2004 stimulation modules apply the relevant voltage values to ECU inputs to which the tire pressure sensors are connected in the real vehicle. The switches for inflating and deflating the tires can be simulated as well. On the other hand, the VT1004 load and measurement modules are used to simulate valves by electronic loads or measure the values at the ECU outputs. CANoe also handles the network simulation and test automation. This so-called Component HIL Test bench **(Figure 4)** for testing an individual ECU is re-used later for tests in the real vehicle as well.
The focus of the new test system is to simulate error situations caused by incorrect wiring, defective sensors, actuators, etc. These components do not supply any values or they supply incorrect values, or they may have internal resistances and current consumptions that deviate from specification. In the case of pressure sensors, for example, voltages outside of the measurement range are generated, electronic loads simulate the valves and current consumptions deviating from normal values can be parameterized. The ECU's reactions to such unpredictable events are of tremendous interest to Unimog experts in their efforts to further develop the system, as is information on whether the ECU generates correct entries in error memory.

Outlook

In testing the tire pressure system at Daimler, it no longer makes any difference whether a real or simulated ECU is tested. Developers also enjoy independence from the availability of real test vehicles. As a result, the test system can be used universally – from the design phase to functional tests. This is what is required if test results for the ECU are to be directly comparable.
Motivated by the success of the VT System in the tire pressure control system, developers in Wörth are already planning their next projects. They include advanced development of the hydrostatic traction drive, in which the system is extended by 16-channel digital modules and the power supply module.

Translation of a German publication in Elektronik automotive, June 2010

Image rights:
Cover picture: Daimler AG
Figures 1 - 2: Daimler AG
Figures 3 - 4: Vector Informatik GmbH

Links:
Homepage Daimler: www.daimler.com
Homepage Vector: www.vector.com
Product Information VT System: www.vector.com/vt-system
Product Information CANoe: www.vector.com/canoe

Mario Wirmel, Daimler AG
initially worked in Production Planning at Mercedes-Benz Trucks in Wörth; since 2003 he has been working in Electrical/Electronic development in the Special Vehicles product area at Mercedes-Benz.

Katja Hahmann, Vector
since 1997 she has been employed at Vector in Stuttgart where she is group manager for CANoe Application Development in the Networks and Distributed Systems product line.

ECU Testing with XCP Support

A Look behind the Scenes

Blackbox tests are typically conducted in the framework of ECU development or in analyzing faulty ECU behavior. This involves connecting an ECU's inputs and outputs to a test system for stimulation and measurement. Although this method lets the test engineer perform extensive analysis, certain tests require looking directly into the ECU. This is the only way to obtain meaningful test results or reduce testing effort.

In most cases, it is actually sufficient to look at the ECU's inputs and outputs to functionally test a component **(Figure 1)**. However, this becomes difficult when state machines are used in the ECU. Their current states can only be derived indirectly by their effects at the ECU's outputs. In the case of sensors whose values are not transmitted over the bus system, it is also very difficult for the test engineer to localize errors to the software interface. From outside the ECU, it is not clear exactly where the sensor value was incorrectly processed.

Different methods that offer access to internal ECU data are used, depending on the phase of ECU development. In early phases, for example, internal ECU values are often output in so-called "reserved development messages" **(Figure 1)**. For the functional developer at a supplier, this is an effective and quick method that precisely targets a specific objective. However, these supplemental messages must be removed for later development phases, especially for system integration and series production. They induce additional bus load, and in the worst case they might even collide with messages of other system components.

Another way to access internal values is through diagnostics **(Figure 1)**. Some information is available directly via diagnostics, e.g. diagnostics offers access to fault memory. Special diagnostic services are also provided to read the required values from memory. The advantage here is that a standardized access method is used. The only precondition is full integration of the diagnostic driver; this is generally provided in today's ECUs. The disadvantage of this method is that a lot of diagnostic protocol information is transmitted along with the actual measured values, and this adds load to the bus system interface.

XCP for Test Access

If bus load needs to be kept low, an alternative is to use a calibration protocol. Originally, such protocols were developed for the ECU calibrator. They let calibrators modify parameters or characteristic maps in the ECU to optimize their algorithms. With the XCP protocol standardized by ASAM, the user can read individual values directly from the ECU's memory as needed. The protocol can also supply a defined set of measured values synchronously to the ECU's application via so-called Data Acquisition (DAQ) lists. The XCP protocol was defined for efficient provision of data over the bus medium. As an example, after configuration the DAQ lists can be transmitted in response to a single identifier from the test system. In addition, measurement times of the DAQ lists can be optimized to internal ECU processes. Automated test systems place similar requirements on the system. Use of the XCP protocol makes it possible to integrate internal values in test sequences without excessive loading of the ECU or the bus system used.

Another reason that a widely used standard like XCP is ideal is that it is very easy to configure in the tool chain. All necessary information is already in the A2L file such as internal program memory locations with their names and communication parameters. Depending on the development environment, the A2L file is either automatically generated. As a second step, after compilation memory addresses are taken over into the A2L file from the linker-map-file. In the test tool, the user only has to configure this file once for each ECU used in the test. In a second step, the user selects the symbols needed for the test sequences from the A2L file.

CANoe Option .XCP

Option .XCP supplements the CANoe test tool from Vector with the convenient option of reading or writing internal ECU values. Besides supporting the standards XCP in CAN/CAN FD, XCP on FlexRay, and XCP on Ethernet, it also supports the previous protocol CCP. Once the A2L file has been configured and the necessary values selected, CANoe automatically acquires them and maps them as system variables. The user can then use these variables in any of the testing tasks. Besides offering access to ECU inputs and outputs, they also provide an in-depth look into the ECU's memory **(Figure 2)**.

In simple analysis tasks, users can display the data in the Trace or Graphic window and use panels to evaluate the results. For more complex test sequences, the tool vTESTstudio offers extensive options for creating CANoe test cases that perform automatic evaluations. For example, this enables checking of the Network Management state machine for correct functionality. The necessary stimulation is performed in the CANoe rest-of-bus simulation, and the ECU's reaction is not just measurable on the bus; it is directly measurable in the ECU over XCP.

The effort required to execute test cases is also significantly reduced, e.g. for test cases that require sensors. The test system writes the sensor values directly to memory cells in the ECU over XCP. This eliminates the need to connect and control original sensors at the ECU inputs – a demanding task. The ECU is notified that the sensor and associated hardware driver have measured the values correctly. The same approach can be used in the other direction. Here it is assumed that the output stage and actuator have been tested and accepted. In this case, the test system measures the value that the application prescribes to the driver stage over XCP.

Access with Large Quantities of Data

If large quantities of data need to be exchanged between the test system and the ECU in a test case, or if especially quick processes need to be monitored, an XCP connection over the CAN or FlexRay bus is no longer effective. In such cases, direct access to the ECU's debug interfaces is recommended. This could be implemented via a NEXUS or JTAG interface, for example. These protocols access the ECU memory directly – in part by using the microcontroller's debug-trace. Taking this approach, the user can quickly read out very large quantities of data from the system without loading the bus or the ECU.

Figure 1: Conventional test system for ECU testing with limited access to an ECU's internal values via diagnostic functionality or special messages created by the developer

Figure 2: Option .XCP is used with a CANoe test system for direct access to an ECU's internal values.

Vector VX hardware, for example, offers direct access to an ECU's NEXUS or JTAG interface **(Figure 2)**. Since this hardware communicates with the test system via XCP-on-Ethernet, integration in CANoe is as easy as integration for XCP access over CAN. Combining VX hardware with the CANoe test system further improves test system performance, without any negative effects on the communication medium.

Updated article – translation of a German publication in Automobil Elektronik, June 2010

Image rights:
Vector Informatik GmbH

Links:
Homepage Vector: www.vector.com

Oliver Falkner
studied Electrical Engineering at the University of Stuttgart. After graduating, he joined Vector Informatik GmbH in Stuttgart in 1999 where he is currently Manager in product management for the Networks and Distributed Systems product line.

Rugged Data Logger Endures Tough 24h Race

Case Study GETRAG

The Customer

With a history of producing 2.6 million transmissions, the GETRAG Corporate Group is one of the largest transmission manufacturers in the world. In 2008, the company successfully launched the GETRAG PowerShift® dual-clutch transmission on the market, which is installed in such cars as the Mitsubishi Lancer Evolution X.

The Challenge

Reliable acquisition and quick evaluation of measured data in a challenging car racing application

The objective was to put the GETRAG PowerShift® transmission to the test in a Mitsubishi Lancer Evolution X during the ADAC Zurich 24h car race to evaluate its durability and racing performance. Six different CAN control modules were used to reliably and quickly log measured data throughout the race for performance analysis.

The Solution

The rugged GL1000 fleet logger

The GL1000 was used as an autonomously operating test instrument, which proved to be rugged, and reliable in processing all trouble-free measured data. This demonstrated that the GL1000 operates flawlessly, even in a tough car racing environment that places especially severe demands on mechanical toughness. Because of its toughness and ease of configuration, the logger fully satisfied all of the requirements placed on it.

The Advantages

A compact device reliably acquires all required bus and measured data

> All bus data is acquired from the CAN channels. In addition, the user may choose to have the measured data logged over the current CAN protocols CCP or XCP.
> Measurement signal lists are individually configurable
> The rugged design and strong aluminum housing protect the device when used in challenging environments.
> Parallel logging of GPS data
> Data exchange is simple and fast, even during bus operation; data is exchanged by SD card with an easy-to-access card slot.
> Offline evaluations are possible with any tool, without requiring connection to the data logger.

Flexible Test Systems for Efficient ECU Testing

Functional Testing with Error Simulation at the Developer's Bench

Functional testing of a vehicle ECU requires testing of the most significant error conditions as well as actual functionality in the vehicle. Systems used for this type of testing must fulfill stringent testing requirements. Vector's VT System is a modular test system tailored specifically to meet the needs of the passenger, commercial and agricultural vehicle industries. It allows the engineer to perform effective functional testing during the early development phases of the vehicle.

The complexity of electronic and software systems installed in today's automobiles requires comprehensive testing in the developmental phases of the ECU. Generally, errors detected early on are easier and cheaper to correct than errors found in the later phases of development. In this process, ECUs are rigorously tested individually in functional tests with special attention given to the numerous error cases. Faulty behavior detected in rare cases or in situations that are impossible to reproduce in normal operation represent a tremendous problem for manufacturers if those errors are not discovered before the ECU has reached the field.

Functional Testing of ECUs

To test its functionality, the ECU is stimulated via its hardware and software interfaces and its reactions are evaluated. It is important to present the ECU with a test environment that matches the environment in the real vehicle as closely as possible. This can be accomplished in a number of different ways. What is most important here is that the ECU should not be able to perceive any difference between the simulated environment on the test bench and the actual environment in the vehicle.

In many cases, ECUs automatically check sensors and actuators so it is imperative that they are connected during the test. If these external components are missing, the ECU will generate faults or deactivate certain functions. As a result, real actuators and sensors are usually connected to the ECU for testing. An alternative would be to simulate the loads and sensors. The great advantage of sensor and actuator simulation lies in the potential for automating test flows with suitable models, a Hardware-In-The-Loop (HIL) test is also possible.

ECU Testing in Error Situations

Additional devices are necessary to simulate error conditions during an ECU test; like the VT System from Vector. They are inserted in the circuit between the ECU pin and the sensors and/or actuators to which it is connected **(Figure 1)**. Specifically, these test components enable testing of the following error conditions:

> Damage to the electrical wiring: Line breaks, short circuits to ground or battery voltage, short circuits between connection lines

> Sensors or actuators are damaged: Sensors do not output any values, the values lie outside of the acceptable value range, electrical properties of the components – such as internal resistance or current consumption – do not conform to specification

> Incorrect input values, especially incorrect sensor data: From the ECU's perspective the sensor is working properly and measured values lie within the allowable range. However, they are implausible or contradict other sensor values.

Figure 1: The VT System is placed between the ECU and the actuators/sensors for testing.

The ECU must react in a defined way in these cases and generate appropriate diagnostic entries. In turn, these entries can be checked by the test system – in this case over the diagnostic interface.

Compact Test Systems with the VT System

Systems based on CANoe and the VT System demonstrate that the stringent requirements of high-performance test systems – with regard to their interfaces and test hardware, test automation, user control of software interfaces and options for rest-of-bus simulation – can also be implemented in a compact test system for the bench.

With CANoe, the user gets a mature and widely used tool for analysis, simulation and test automation. Vector hardware interfaces provide reliable bus interfaces to CAN, LIN, FlexRay and MOST. External measurement and test hardware from various manufacturers may also be connected via GPIB, the serial port or Ethernet.

The VT System is a modular I/O system that controls the ECU's inputs and outputs for functional testing with CANoe. It allows users to set up compact test benches of widely varying complexity **(Figure 2)**.

The PC with CANoe is connected via the computer's Ethernet port using the real-time capable EtherCAT protocol. This means that flexible test systems can be constructed with minimal integration and wiring effort.

Different modules are available for driving the various ECU inputs and outputs. However, all modules share these properties:

> The ECU's I/O lines are connected directly to the VT System. Each module provides all circuits for the I/O channel, thus substantially simplifying the test bench setup **(Figure 3)**.

Figure 2: VT System modules enable setup of very compact test systems as well.

Figure 3: Modules of the VT System integrate all of the I/O channels needed for the test.

> If necessary, real sensors and actuators can be connected to the module. Using relays on the VT module, users can toggle between the use of externally connected original equipment components and the simulation residing in the module.
> On each ECU pin, relays can be used to generate errors such as line break and short circuits between lines, to ground or to supply voltage.
> All modules and measurement devices are designed for voltages and currents typically used in vehicle electronics. Additional signal conditioning hardware is not required.
> VT System modules are automatically registered onto CANoe and are ready for use after a minimal configuration effort. All measurement, output and control signals can be accessed through CANoe and may be used in test scripts together with bus signals, or evaluated in analysis windows.

The VT1004 load and measurement module is connected to the outputs of an ECU. In the vehicle, these outputs are normally connected to actuators such as positioning motors and lamps. The module contains an electronic load for each channel to simulate these actuators. The voltage at the ECU output is measured at a sampling rate of 250 kSample/s; it is then processed in the module and the results are transmitted to CANoe as momentary, mean and effective values. The module can also measure PWM signal parameters. It can handle high continuous load currents of up to 16 A, such as those occurring when lamps and motors are driven.
The VT2004 stimulation module is connected to the inputs of an ECU. To simulate sensors, it has a decade resistor on each channel that can be controlled by a test script. Alternatively, the sensor can be stimulated by voltages; voltage curves or sequences stored on the module can be reproduced with high precision. The module can also generate PWM signals and simulate a potentiometer on a channel how it is used for fuel level sensors, for example.
The VT2516 digital module controls ECU inputs or outputs that utilize digital signals, e.g. lines in the vehicle that are connected to switches, encoding jumpers, small indicator lamps or LEDs. ECU inputs are stimulated by digital signals with configurable levels; besides bit sequences stored on the module, PWM signals may also be output here. In the opposite direction, the module can measure digital or PWM signals and voltages output by the ECU at each channel. Using selectable resistors, loads can be simulated or pull-up or pull-down circuits implemented.
An ECU's power supply lines are connected to a power source through the VT7001 module **(Figure 4)**. Precise current measurements are acquired over a wide range making it possible to check quiescent states, for example, or analyze the power consumption of different software variants. The module also generates control voltages for external power supplies thus allowing simulation of defined disturbances on the power supply lines. Rated for continuous currents of up to 70 A, the VT7001 can also supply ECUs with high power consumptions. An ECU's two supply current inputs, Terminal 15 and Terminal 30, can be controlled separately.
Furthermore, there are modules for bus connections, for the execution of simulation and test in real time on the VT hardware as well as for the integration of custom I/O circuits in the VT System. Other modules are in development. The ECU's I/O channels do not need to share these test setups in all cases, since the VT modules are provided separately for each channel. This simplifies test automation and programming and facilitates clear representation of multiple errors and more complex user operations.
Based on its modular organization, the VT System is ideally suited for both small test setups at the developer's bench and comprehensive test benches in the test laboratory. Together with CANoe, the test engineer has a flexible and high-performance solution for automotive compact test systems. Test automation is implemented in an efficient and seamlessly integrated package using CANoe and VT System.

Figure 4: The new VT7001 module supplies an ECU's power supply inputs.

Translation of a German publication in Elektronik Praxis, Special Edition 'Automotive', issue September 2011

Image rights:
Vector Informatik GmbH

Links:
Homepage Vector: www.vector.com
Product information VT System: www.vector.com/vt-system
Product information CANoe: www.vector.com/canoe

Stefan Krauß
studied Computer Science at the University of Stuttgart from 1990 to 1995. After graduating, he worked as a scientific assistant in the Software Engineering department of the university's Institute for Computer Science until 2001. Since 2002, he has been employed at Vector Informatik GmbH in Stuttgart where he is currently Product Manager for the VT System.

Functional GUI Testing of In-Vehicle Infotainment Systems in Virtual and Real Environments

Combined Testing with CANoe and Eggplant Functional

In-vehicle infotainment (IVI) systems are part and parcel of today's automotive sector. Their graphical user interfaces (GUIs) serve as the "face to the customer," and externally represent a variety of embedded vehicle applications. Organizations need to adopt and apply measures for quality assurance to the distribution of software-based interfaces, as complex logic (which poses new challenges to the development process) is required to operate embedded systems. Trends such as the expansion of functionality using new software components and flexible adaptation of the user interface can further increase this complexity. As a result, the testing phase of user interface development continues to take on greater importance over time. Verifying IVI systems requires an interdisciplinary approach: one that's capable of handling the complexity resulting from the interplay between software-based user interfaces and embedded systems. To enable efficient testing of the overall system for this purpose, Testplant and Vector have combined their domain tools — Eggplant Functional for GUI tests and CANoe for ECU testing.

Embedded systems are ubiquitous in everyday life today, and are spreading especially quickly in the automotive sector. To operate vehicles, users rely on the functions and data of these systems, which increases the importance of GUIs as the main element of interaction between human and machine.

In vehicles, these interfaces used to be primarily hardware switches but are now being replaced with software implementations because they can be used in a wider variety of ways and can even be easily changed and expanded at runtime. These IVI systems are largely being developed to be the main interface in the vehicle, as they cover an increasing number of functions and access the vehicle on a deep level **(Figure 1)**.

To cope with this increasing importance, functionality must already be ensured during the early stages of development. Frequently, the target hardware is not yet available, or is at least incomplete at that point in time. So, validation must often occur in a purely virtualized environment or on an isolated ECU for each remaining bus simulation.

The interfaces, users want, are characterized by clear and comprehensible visualization of the underlying vehicle applications—despite their complexity—as well as by intuitive and easy operation (i.e., usability). Ultimately, the goal is to make IVI systems as smart as smartphones, which involves highly responsive, touch-based infotainment interfaces with a broad spectrum of functions that also need to be ensured. Customers always expect the technology of the main unit to be state of the art, both in terms of the operating concept and the supported

Figure 1: In-vehicle infotainment: the main interface in the vehicle

peripherals and audio/video codecs used. The result is a high update frequency which peaks when the car is connected and the software is updated in the field. These framework conditions highlight the requirements for repeatable, automated, and maintainable tests.

An Interdisciplinary Approach

Satisfying these requirements requires an interdisciplinary approach that combines the tools from both classic GUI testing and ECU testing to ensure comprehensive verification in every stage of development.

Eggplant Functional, the software from Testplant for functional GUI testing, enables embedded software applications to be tested based on their user interfaces.

With respect to testing on the program code or function level, a greater depth of testing is ensured here. Test automation draws on image and text recognition algorithms to detect buttons and indicators. Should a button be activated, for example, its position is identified through image recognition, and a button press initiated. The transfer of screen content and the initiation of user interactions occur on the system level through remote-control mechanisms such as Virtual Network Computing (VNC), Remote Desktop Protocol (RDP) and Keyboard-Video-Mouse (KVM) over IP. VNC and RDP are protocols common in the PC domain for transferring screen content and keyboard and mouse input. KVM switches typically rely on the hardware level for implementation of the same functionality in order to transfer the input and output information. Changes to the software to be tested, which are made to ensure testability, are thus unnecessary.

A noninvasive approach like this ensures that changes, which are otherwise only carried out to test the software, have no effect on the actual test. Since remote-control mechanisms can now be found in every commonly available operating system (Android, iOS, Linux, QNX, VxWorks, Windows, etc.), any ECUs can be tested using this approach. Eggplant Functional runs as a distributed system, and the host application with the test sequence control is executed on a conventional PC for this purpose. Only the remote-control component is required on the ECU to be tested. Using various interfaces, Eggplant Functional can interact with test automation systems of other manufacturers and is thus designed as an open system.

The CANoe software from Vector enables remaining bus simulation for the ECU to be tested and the analysis of bus sizes. It also provides a testing environment which includes test sequencers and test reporting. The tests themselves are created by Vector in vTESTstudio, the authoring tool for editing test sequences for embedded systems. Tabular, graphical, and programming language-based (CAPL/C#) test notations and design methodologies are available for this purpose. CANoe (CAN Open Environment) was designed explicitly as an "open environment" and offers a host of interfaces for linking external programs **(Figure 2)**. This includes, among others, Vector's own UDP-based Fast Data Exchange (FDX) interface, the ASAM XIL API standard, the Functional Mockup Interface (FMI) and MATLAB. External DLLs can also be integrated, an option used to connect Eggplant Functional to CANoe.

Eggplant Functional provides an access option for external applications over an XML RPC interface and is thus run as eggDrive in a mode without a GUI. The interface makes it possible to call functions and test scripts, as well as to read back partial, individual results or the entire test report. Testplant already offers an implementation in the form of a .NET assembly for use of the XML RPC interface. This enables linking to CANoe/vTESTstudio **(Figure 3)**. Encapsulating the necessary calls in a C# test library and reproducing the test results in the CANoe test report yields an easy-to-use interface for the combination of CANoe and Eggplant Functional. The test designer creates the ECU tests in the vTESTstudio environment they are familiar with and draw on the functions of Eggplant Functional in the same way as the scope of functions belonging to CANoe.

The Benefits

The integrated approach from Vector and Testplant provides the user with a testing environment in which to monitor and stimulate the GUI of the ECU, as well as to stimulate and analyze the values transferred on the vehicle bus. From the point of view of the test developer, the level on which they access the information does not play a role.

Because it is also possible to initiate existing Eggplant Functional test scripts, the scripts available from the pure

Figure 2: CANoe as an open environment

user interface test can be reused in an integrated, functional GUI test. In the combined use case, domain specialists can also use the tools already familiar and available to them. Such an approach increases the testing depth and degree of test automation, and also ensures deep and comprehensive functional testing of the ECU, even in the event of more frequent software releases. Such a test setup makes it possible to already ensure an isolated ECU, including the user interface during the early stages of development. ECUs which communicate with the ECU to be tested are simulated by CANoe and do not actually need to be present here. Functional GUI tests thus enable functionality to be ensured with a transparent hardware expenditure from very early on, which is also ideal for creating a development environment with continuous testing. It also provides the option for daily builds, for example, and enables early responses to problems in new software releases—facilitating frequent software releases and short development cycles, and ensuring a timely response to new requirements or framework conditions.

Conclusion

Software-based embedded systems operated via GUIs are playing an ever more important role in automotive development. To satisfy quality assurance, testing both individual software components and the user interface is required.

Thanks to their open interfaces, Eggplant Functional and CANoe test automation tools are ideal to automatically verify the functionality of embedded systems—completely and reliably. Domain specialists can implement tests in their preferred environment and integrate them into an overall test suite—eliminating expenses associated with creating additional tests. This, in turn, enables shorter development cycles with frequent software releases, and a competitive advantage for the OEM or supplier.

Figure 3: A chain of tools for integrated, functional GUI testing

Authors:

Sebastian Mezger
Team leader and project manager for customer-specific testing systems at Vector Informatik.

Mingzhi Deng
Technical consultant at Testplant.

Translation of a German publication in Hanser Automotive, issue 5-6/2017

Image rights: Vector Informatik GmbH

Testing In Heterogeneous Tool Landscapes

What Kinds of Interfaces Are Available?

Many automotive software tools provide outstanding services custom-tailored to their specific fields. A common assumption would be that coupling multiple expert tools together would create a valuable development environment for ECUs, but the tools often just end up being isolated applications which can't interact with one another or only in a limited capacity. The simple fact is that every modern tool integrated into a larger system requires extensive options for interacting with its environment.

During development and, more importantly, testing of E/E systems in the automotive sector, the industry is being confronted with a growing number of different software tools which have to work together in some way. In the area of autonomous driving and the associated ADAS-based systems **(Figure 1)**, for example, you can quickly end up using three, four or more tools, possibly from different manufacturers. Each one of these tools is highly specialized in its own field: one covers remaining bus simulation of the communication networks, another contributes vehicle dynamics or environmental models, a third feeds virtual driving scenarios into the simulation and, last but not least, test automation is also carried out – and this may also include a link to a test data management system. What's more, they all need to be coupled together to form an executable network of tools **(Text box** *Tool coupling relevance***)**.

Connecting different tools using your own proprietary link is one obvious solution. This has the advantage of complete flexibility, as the implementation is totally in your hands. However, proprietary solutions also have crucial disadvantages: one such disadvantage is that these implementations are extremely project specific, which is an obstacle to reuse in other projects – if not being impossible. Such solutions are also tied to specific individuals or teams – if a key person leaves the company or changes departments, the learning curve for their successor is correspondingly high due to the resulting lack of expert knowledge. If one manufacturer changes the behavior at one interface or the interface itself, an enormous effort is required to keep the links functional across different versions. This is why it makes sense to rely on standardized interfaces.

Tool coupling relevance

In the context of this article tool coupling is understood to mean the following:

> In a co-simulation scenario at runtime, possibly as a closed loop.
> A tool generates, exports or compiles models which are integrated into other tools.
> Tools can be integrated into higher-level systems, e.g. for test data management or requirement management. This article does not address this type of connection, however.

As standard interfaces don't change, they exhibit high reuse potential by their very definition. They may be expanded later on, but backwards compatibility is usually ensured. This keeps maintenance costs in check. Another advantage is that linking standard interfaces is frequently achieved by configuration, whereas linking proprietary interfaces usually requires programming. Standard interfaces aren't perfect, though, as they typically don't cover the full range of desired functions. As a result, mixed solutions where a large portion is covered by standard interfaces. The remaining requirements are implemented through additional proprietary development are frequently encountered.

Figure 1: Questions regarding the right interfaces for the integration of different tools

Regardless of the solution, the following generally applies: to ensure the efficient implementation of projects, any software integrated into a larger system requires comprehensive options, i.e. interfaces, for interacting with its environment **(Figure 1)**.

Functional Mock-up Interface

One example of a standard interface is the Functional Mock-up Interface (FMI). Generally speaking, FMI is a software-independent standard for model exchange, where the models can originate from different sources. One instance containing an FMI-compatible model is called a Functional Mock-up Unit (FMU). In this case, the description of the interface data is separated from its function. This means that models can be shared while maintaining intellectual property protection – the model is available as a black box. This standard supports two modes. In "Model Exchange" (ME) mode, the FMU provides inputs, outputs and internal states. Software which integrates the FMU ME provides the inputs, simulation time and new states. The algorithm for calculating the model, the so-called solver, must be provided from outside, however. For dynamic models such as controllers, these algorithms typically consist of approximation methods for solving differential equations. In the other mode, "Co-Simulation" (CS), the FMU already contains the solver algorithm – the software which integrates the FMU merely calls a step function at intervals corresponding to the simulation increments. As a result, the FMU-importing software doesn't need to know anything about the internal details of the model - the outputs of the model are calculated as a function of the inputs and the underlying solver.

The standard thus describes the inputs, outputs, internal parameters and, if applicable, a step function for calculating a simulation step in the model. In addition to pure model exchange between different tools using FMI, tools can also be coupled with one another. The FMI standard does not describe an infrastructure or transfer protocol for the actual exchange of data, though, so how can a tool like the widely used CANoe development and testing tool from Vector Informatik be linked to other systems?

A CANoe excursion

A CANoe configuration usually consists of a remaining bus simulation and, if applicable, executable test automation. Both the simulation and test automation are executed in the same process and are therefore synchronized by a common time base. Test automation implements function-based tests on the network level and/or of the inputs and outputs of an ECU. The remaining bus simulation, on the other hand, consists of simulated network nodes containing the communication model of the entire network in the so-called interaction layer. The remaining bus simulation can contain multiple networks of the same type, optionally linked using gateway nodes, or multiple networks of different types such as CAN, LIN, FlexRay and Ethernet. This multi-bus capability, including time synchronization across all channels of all network types, represents a special feature of CANoe. An FMU defining the relevant data to be exchanged can be exported on this basis **(Figure 2)**. In the process, a corresponding selection containing only this subset should be made. Signals from cyclical messages, for example, which merely keep the ECU active, but are not

Figure 2: FMI/FMU workflow in CANoe

necessarily related to the application, don't have to be transferred across tool boundaries. Also network management related messages typically don´t play a role in this context. FMUs are exported solely through the configuration, i.e. no manual adaptation or programming work is necessary on the CANoe side.

FMI link with CANoe

As mentioned above, the FMI standard does not describe an infrastructure or transfer protocol for the actual exchange of data. On the technical level, CANoe provides an interface for the fast exchange of data over Ethernet – which is open, but proprietary in this case. This Fast Data Exchange (FDX) basically specifies the sending of User Datagram Protocol (UDP) data over Ethernet using a defined protocol. This interface is public and used very frequently by users to link different components and subsystems, such as with test benches. As UDP is very widespread, CANoe can be coupled with a large number of different systems using FDX. The drawback of full flexibility is that the users have to implement the remote node themselves. The integration of FMI in CANoe uses FDX solely as a transport layer to implement the exchange of data for the tool connection **(Figure 3)**. The FDX transport layer remains hidden from the user, and an FMU is conveniently created through configuration, rather than programming. Since

Figure 3: FMU export from CANoe

FDX is based on UDP, the importing of a CANoe FMU also works on systems not running under Windows. In addition to FDX for the pure exchange of data, a separate TCP/IP connection is established by the FMU and CANoe. This connection functions as a control channel through which the simulation is started and stopped.

As the FMU-loading system and CANoe run on different time bases, the simulation times have to be synchronized. For CANoe, this only works in simulated mode. The simulation time can only be influenced from outside in this case. In contrast, the CANoe simulation time continues to run independent of the simulation time in the FMU tool when CANoe is working in real mode.

In addition to the remaining bus simulation, CANoe is also used to perform automated tests. When used in combination with other tools and test platforms, it will be necessary to consider which tool will act as the test master. If CANoe is operated in real mode, i.e. with access to the communication networks, it is typically also the master for the simulation time.

ASAM XIL API

The XIL API offers an additional interface especially in the environment of test automation. The interface is standardized by the ASAM [2] and abstracts simulation tools as well as test tools. This allows for the separation of test run control from test execution. In the CANoe example, separation means both that it can be remote-controlled as an instance for test execution and that CANoe can remote-control other test execution tools **(Figure 4)**. Coupling tools from different manufacturers is also possible using the API. The interface addresses various different ports, with the most important one in this context being the Model Access Port (MA), which provides write and read access to the model. This includes the recording of simulation data, the creation

Figure 4: Decoupling test sequence control and test execution with the XIL API

of signal generators for the simulation, the definition of triggers, error handling and methods for controlling the simulation. The Electrical Error Stimulation (EES) Port is an additional port which serves to control special hardware to influence electrical fault conditions.

The XIL API is based on .NET; as such the ability to meet certain real-time requirements, for example a closed-loop scenario, would depend on the specific implementation. For this data transfer, CANoe uses the UDP-based FDX interface described above and thus is real-time capable – at least as capable as an exclusive Ethernet link between two systems allows. If gigabit Ethernet in a 1-to-1 connection is assumed, performance is also within a range satisfactory for most applications.

Component Object Model (COM)

Finally, Microsoft's COM interface should be mentioned. Although it's not suitable for the exchange of data during the time of simulation due to Windows scheduling, COM is still very powerful in terms of configuration tasks and sequence control. This means that COM provides a control channel not available with FMI, for example. In this way, a large number of the functions available for interactive handling through the user interface can be remote-controlled and automated via COM. For this reason, users will also frequently need a combination of different interfaces for a fully automated test or simulation sequence ? COM to prepare the configuration and sequence control and FDX, for example, for the exchange of simulation data at runtime.

Outlook

Unfortunately, even standard interfaces frequently don't offer a complete approach. Although models can be incorporated using FMI and open tools such as CANoe can be connected to other tools using FMU export, there is no option for controlling and configuring the overall system – the control channel mentioned in the article is missing. Another issue is time synchronization. Who provides the time basis in a distributed system? What is it derived from? How are individual subscribers synchronized to this global time? How can simulated parts and real-time parts be coupled? The ACOSAR project financed by the ITEA3 European research program is addressing these issues. The project is currently in the specification phase, and the API specification should be available by July 2018 [2].

Jochen Neuffer
He studied communications engineering at the Esslingen University of Applied Sciences in Germany. He has worked at Vector Informatik since 2002, where he is a product management engineer in the Tools for Network and Distributed Systems Department.

Translation of a German publication in Elektronik automotive, issue 6/2017

Image rights: Vector Informatik GmbH

Literature References:

[1] ASAM. https://wiki.asam.net/display/STANDARDS/ASAM+XIL. [Online]
[2] ACOS. http://acosar.eu/index.php. [Online]

Further approaches to tool coupling

The "oe" in CANoe stands for the "open environment" which natively supports a large number of different application options and interfaces. The examples presented in this article represent just a small selection. Shared Variables from National Instruments, which can be used to exchange data with NI-based systems over Ethernet, for example, was not mentioned. Nor was MATLAB®/Simulink® Integration, which makes it possible to integrate Simulink models into an MIL or HIL scenario in CANoe. This is why it's worth looking at the CANoe documentation before users begin their own comprehensive development. Chances are good CANoe already provides appropriate tools for solving the problem at hand. By using standardized interfaces, users increase both the stability of new versions of the tools involved and the degree of reusability in other projects.

Vector Global Services Implements VectorCAST Test Framework for IEC 61508 Compliance

Case Study Moore Industries

The Customer

Established in 1968 and privately held, Moore Industries is a world leader in the design and manufacture of interface instruments for industrial process control, system integration and factory automation. Functional safety products include temperature transmitters, alarm trips, isolators, relay repeaters and other components for use within the process industry. Moore's typical customers include oil refineries and manufacturing plants. Most of the company's software development is conducted at its headquarters in North Hills, California.

The Challenge

Moore was faced with a functional-safety test mandate that required them to test the embedded software for a dual-input temperature transmitter. This component required intense unit and integration testing to comply with the functional safety standard IEC 61508. The company had very limited resources to conduct the testing in the desired amount of time. Additionally, safety requirements demanded that an independent third party conduct the testing.

Moore researched outsourcing options for the test project. They discovered Vector. Interested initially in Vector's unit test and code coverage tools to automate testing, Moore was introduced to the Vector Global Services team who proposed to do all of the required testing as a service. Vector was bought on-board to test setup, development, execution and reporting.

The Solution

Working with its services partner CRITICAL Software, the Vector Global Services team put together a Project Management Plan (PMP) during the first week of engagement. A PMP establishes the working relationship, objectives, communication protocols, status reporting and detailed completion criteria, so that each project maximizes success probability as defined by Moore. The PMP also ensures Vector has a good arrangement with any providers or partners, such as CRITICAL Software, who also follow the same PMP.

For this project, Moore provided approximately 25,000 lines of executable code in groups of 5,000 LOC for each two-to-three week period.

Following successful completion of testing and baselining the initial project, Moore now uses the VectorCAST test automation platform to test all new functionality and augment the original testing done by the Vector Global Services team proving that 100% requirement and decision-condition coverage metrics are achieved. Any coverage not achieved because of anomalous conditions is satisfied with a unique rationale.

The company also uses VectorCAST/C++ for unit testing and VectorCAST/Manage to provide aggregated analytics on all regression tests being performed. Tests are being run via PC due to speed and simplicity, as well as on target boards to get binary validation. VectorCAST/RSP is being used for on-target/simulator unit testing of the PIC 24FJ256GA106/Microchip C30 v3.31 compiler.

The Advantages

When Moore initially considered undertaking this project using internal resources, they quickly realized it would take at least eighteen months to complete. By utilizing the Vector Global Services team, Moore Industries reduced this to a total project time of just four months.

"Even if we hired people, we wouldn't have been able to hire enough people to provide this in the time that we wanted to be able to produce it", stated Charles Larson, Director of Technology and Software Development at Moore Industries.

Utilizing Vector's Professional Services allowed Moore's software development team to focus their efforts on developing application code and systems/hardware-level testing instead of writing, managing, and reporting on software unit tests. The Vector Services team was able to "baseline test" the current application and provide Moore with direction on how to use the VectorCAST tools to become self-sufficient for future testing requirements.

"All the way down the line [the testing project] was on time and on budget. We were really impressed with how it was managed", added Larson.

Automated Test Solutions Simplifies Implementing a Software Quality Process

Case Study SsangYong Motor Company

The Customer

SsangYong Motor Company is the fourth largest South Korea-based automobile manufacturer. Headquartered in Seoul, SsangYong Motor got its start building jeeps for the US Army. Since 1963, the company has supplied automobiles and commercial vehicles to the Asian and global automotive market.

The Challenge

In 2011, SsangYong Motor Company began managing a functional safety response taskforce and needed to adopt a test solution that would easily integrate with their existing QA•C static analysis tool from PRQA and help them meet ISO 26262 compliance.

The Solution

SsangYong chose the VectorCAST platform for its complete fidelity with the PRQA tool and its excellent reputation for unit testing, system level testing, and structural coverage analysis.

The Advantages

VectorCAST gives SsangYong developers the ability to verify ISO 26262 compliance for software delivered from suppliers as well as assuring compliance as components are integrated into the complete automobile.

> Reliable and Seamless Test Environment

Automobile manufacturers seeking compliance with ISO 26262 requirements face a daunting challenge. Part 6 of the standard addresses the requirements for software development and required organizations to create tests to prove the correct implementation of low-level requirements, as well as measure code coverage to ensure testing completeness.

SsangYong Motor is able to meet ISO 26262 standards leveraging a combined VectorCAST and QA•C solution. They use PRQA QA•C for static analysis, including MISRA compliance and use VectorCAST to meet the testing and code coverage requirements. The integrated solution improves developer efficiency and results in higher quality software applications.

A key feature of VectoCAST is its perfect fidelity with the compiler and target environments being used. With VectorCAST SsangYong Motor is able to test using the production hardware ensuring that the test results reflect the real-world environment where the applications will be deployed.

> VectorCAST Certification Kit Eases the Path to IEC Compliance

In order to demonstrate compliance with ISO 26262, SsangYong Motor uses the VectorCAST IEC Certification Kit.

The VectorCAST IEC Certification Kit is a package of certification artifacts and tools. It provides documentation, certificates, templates, and tool qualification documents that help organizations use the VectorCAST automated test platform on projects based on the IEC 61508, EN 50128, and ISO 26262 standards. The kit includes certificates and reports from certification authority TÜV SÜD Rail GmbH, that are based on documented, application-specific verification workflows.

> Complete System for Testing Automotive Functional Safety

"Our testing system is now complete and possesses all the elements needed to comply with the functional safety international standards", stated Mr. Jeong Jae-wook, Functional Safety Taskforce Team Head at SsangYong Motor Company.

Mr. Jeong continued, "[With VectorCAST] we will develop components that are ISO 26262 compliant. We are now able to verify the compliance status of suppliers by applying the ISO 26262 verification standard".

VECTOR

Autec Meets EN ISO 13849-1 and EN IEC 62061 Standards with VectorCAST

Case Study Autec s.r.l.

The Customer

Autec s.r.l., produces radio remote controls for industrial cranes, lifting machinery, logistics and construction. Since 1986, Autec has designed, manufactured and delivered wireless control systems specifically for safety applications. The company's radio remote controls are used in industrial overhead cranes, truck mounted and construction cranes, as well as elevating work platforms.

The Challenge

In order to ensure safety and quality and obtain functional safety certifications, Autec must meet EN ISO 13849-1 and EN IEC 62061 (derived from EN IEC 61508) standards. To fulfill these requirements, Autec engineers must unit test complex code, including legacy code, and provide documented evidence regarding the coverage level achieved in testing.

The Solution

Following an extensive evaluation, Autec determined that VectorCAST's embedded test solution could not only help them meet their software standards challenge, but also operates seamlessly with their preferred development suite. Because of the VectorCAST tool's ease-of-use and tight integration with their development platform the Autec engineering team selected VectorCAST/C++ as their software test solution.

The Advantages

By employing VectorCAST, Autec engineers are able to reduce the time needed to test their designs, enabling them to gain greater productivity and higher levels of software quality. Autec is now able to rapidly deliver results in an comprehensible form for the validation and certification process.

> **Meeting Functional Safety Standards**

The engineers at Autec face a unique challenge as they develop cutting-edge radio remote controls; while they do not manufacture the heavy equipment their devices are used in, they literally "take control" over them. Typically used in hazardous operational environments, these wireless control systems must perform flawlessly in the field. In order to ensure that the embedded software used in the control systems is up to the task, they must meet tough EN ISO 13849-1 and EN IEC 62061 (a subset of EN IEC 61508) safety-critical standards.

To meet safety-critical software Verification and Validation, Autec needed an automated test solution that would integrate with their existing development environment, and provide proof that the C code passed stringent target and host tests. Autec engineers use their preferred development suite to develop the firmware used in the company's radio remote controls for industrial applications; complex embedded software applications are required to control this equipment as intended.

> **Increased Efficiency through Automation**

After conducting an initial web search for unit testing tools, Autec made a short list and began evaluating them. The team's goal was to gain confidence that the test tool could meet their requirement to document adherence to relevant standards, and show that VectorCAST could provide traceability to these standards.

Prior to using an automated testing tool, the team had a process by which they would perform unit testing on both the assembler and C code. The tests were built manually using the simulator included in the IAR Embedded Workbench development suite. While evaluating the tool, the Autec team approached Vector's support and made a request that was quickly sorted out by releasing an ad-interim version of the VectorCAST tool that helped meet their testing needs. Impressed by the fast start-up process, and rapid testing times, the team selected VectorCAST as their test environment.

> **Standardizing on VectorCAST**

Due to the quick ramp-up time, the rapid and responsive support services, the reduced testing time necessary, and the improved quality of the software, Autec is implementing VectorCAST as their standardized testing tool. Although the organization has only used the tool for a few months, management estimates that Autec has saved hundreds of man-hours in development time.

Easy access to CAN network analysis

Jochen Neuffer

Author
Jochen Neuffer
Product Management Engineer
Vector Informatik GmbH
Tools for Network and Distributed Systems
Ingersheimer Str. 24
DE-70499 Stuttgart
Tel.: +49-711-80670-4808
Fax: +49-711-80670-584808
jochen.neuffer@vector.com

Link
www.vector.com

Changing requirements are continually in flux when it comes to networking ECUs (electronic control units). The general trend for tasks is becoming increasingly more complex, and so they require even more complex tools. However, there are often simple tasks whose quick handling is actually hindered by this complexity when the user is confronted with a multitude of features. For such tasks, the user wants an easy-to-operate tool. However, if the task requires it, the user also wants to be able to access an extensive set of features.

These conflicting interests occur in typical tasks such as network monitoring, stimulation or data logging. In the case of monitoring, for example, different perspectives are often of interest in observing the data traffic on the network. Here, the Trace function shows the time sequence of all network events. It is also possible to graphically display individual parameters. Moreover, the user typically wants an overview of the network statistics. In stimulation, on the other hand, specific messages need to be sent on the network either spontaneously or periodically. And of course data needs to be logged for later off-line analysis.

These three core tasks are the domains of CANalyzer Beginner, a special execution mode of CANalyzer from Vector Informatik. The mode that focuses on these core tasks is easy to operate even for new users. The individual task areas may be combined, and each may be added or removed whenever the user wishes. The full range of CANalyzer features is at first not visible to the user, but it can be called up at any time.

CANalyzer Beginner can be immediately used as part of any CANalyzer installation (CAN and LIN networks). This saves the user time and money, because there is no need to purchase or install a separate tool. The Beginner mode exploits the advantages of the revised window layout in CANalyzer. The individual tasks are organized on fixed desktops, which do not need to be modified, and which already contain pre-configured windows (see Figure 1). This eliminates the need for time-consuming manual configuration. Since the windows have fixed positions, it is easy to focus on the essentials. Furthermore, the windows cannot be closed, which eliminates searching for inadvertently closed windows. The windows are configured by drag-and-drop operation or with the help of functions on the toolbar.

Figure 1: Desktop with fixed configuration for network monitoring task (Source: Vector Informatik)

Figure 2: Creating a new configuration with CANalyzer Beginner (Source: Vector Informatik)

One can create his own configuration with a few mouse clicks. To do this, the user needs to add for each network a channel and a suitable network description file (DBC for CAN or LDF for LIN) to the central configuration window (see Figure 2). If applicable, the bit-rate is also configured, and the user then selects the tasks that need to be performed. During the measurement, for example, the Trace window offers many different options for filtering specific events, such as blocking and passing filters for messages or channels. Furthermore, the Trace window offers a long data history, so that long-term measurements over several days can be preserved. The Statistics window offers a detailed summary of the current situation on the network and can prepare statistical information on the node level or the message level.

Also complex tasks may be performed. A configuration created with the Beginner mode can be loaded in its full form in CANalyzer. It is also possible to use CANalyzer to perform further off-line analysis of logged data. There is a seamless migration path from CANalyzer Beginner to CANalyzer. Configurations that have been created with CANalyzer Beginner can be loaded in CANoe as well.

Future planning for CANalyzer Beginner calls for supporting additional tasks and possibly adopting concepts into CANalyzer.

Automatic Validation of Diagnostic Services by Use of a Diagnostic Integration and Validation Assistant at Opel

For the first time, a fully automated test case generator has been introduced in diagnostics validation at General Motors Europe (GME) Development. This article describes the introduction of this automated testing of diagnostic implementations based on the example of the new Opel Insignia. An electronically readable diagnostic specification forms the basis for test generation. The article describes how the tool used – CANoe.DiVa (Diagnostic Integration and Validation Assistant) from Vector Informatik – was integrated in the existing tool environment, and it addresses cost and time savings as well as improvements to technical processes that were realized compared to conventional, manual validation at the Opel Corsa.

Introduction

One consequence of strong competition in the global automotive market is that it is forcing a shortening of development cycles. Another is that the complexity of the electronic networking architecture is continually increasing. Key goals in replacing conventional systems by electronically controlled systems relate to cost reductions, a high level of safety and reliability as well as better manageability. Despite all of the benefits, it must not be forgotten that increased numbers of electronic components in vehicles can increase the probability of electronics-related faults. Since reliability is an important criterion for customers when purchasing a new vehicle, it is essential to introduce new methods that enable mastery of this complexity, accelerate the development process and guarantee proper operation of the installed ECUs. Particularly in the area of diagnostic functionality provided by the ECU, it is crucial that diagnostic services are correct. They transport information that helps mechanics in the service garage to quickly determine the cause of a fault and correct it. This information must make it possible for the mechanic to decide which component is the source of the problem and what needs to be replaced to restore full operational readiness. If this is not assured, the result may be erroneous replacement of properly operating units [1], which causes a rise in warranty costs and a decline in customer satisfaction.

The E/E architecture of the Opel Insignia consists of several Controller Area Network (CAN) and Local Interconnect Network (LIN) bus systems [2, 3]. All bus systems are accessed via a central diagnostic port (DLC), see **Figure 1**. Communication is defined by a GM-specific protocol. This GM diagnostic specification is based on KWP2000 [4] and the CAN 2.0A standard. It contains all diagnostic services allowed for addressing an ECU's diagnostic system to obtain diagnostic information. These services are then output by the diagnostic tester to establish diagnostic communication. As soon as a request is sent, the addressed ECU(s) react with either a positive or negative response:

> Positive responses contain the diagnostic information requested by the diagnostic device. If there is a lot of diagnostic information, the response may include multiple message frames.

> Negative responses contain a clearly defined Negative Response Code, which gives information indicating the reason for the negative response. Negative Response Codes are given in accordance with the GM Diagnostic Specification.

The received responses must enable technicians to determine the cause for a fault, so that they can perform the right tasks to solve the problem.

Therefore, the success of a fault correction in the service garage depends considerably on the accuracy and precision of the data output by the diagnostic system. Proper implementation of diagnostic services is essential in performing quick and professional service or maintenance to the satisfaction of customers. Diagnostics also plays an important role in end-of-line testing: it is used to program ECUs and assure product quality. That is why comprehensive validation of diagnostic functionality is absolutely necessary.

Validation Process and Tool Environment at General Motors Europe

In development of the Opel Insignia, GME introduced the DiVa tool from Vector Informatik for the first time. DiVa automates generation and execution of diagnostic tests. **Figure 2** shows the tool environments for the Opel Corsa and Opel Insignia. In both cases, CANoe [5] is used as a test tool. While validation is largely performed manually in development of the Corsa, in development of the Insignia the vast majority of testing is covered by fully automated tests. **Figure 3** shows a typical diagnostic validation process for an ECU performed by a test engineer at GME. Development of the ECU software is subdivided into several phases. At the beginning of an ECU development, the focus is more on implementation of ECU functionality than on diagnostic services. The latter are then elaborated and developed in subsequent software versions. As shown in **Figure 3**, with introduction of the Phase 1 (SWR 1) software version, only a small number of diagnostic services are implemented. The use of diagnostic software components at GME (CANdesc) has made it possible to implement a portion of the diagnostic content early at the start of development, and as a result it is integrated in the ECU earlier **(Figure 3)**.

The number of diagnostic functions to be tested grows with each development cycle. Once all diagnostic services have been implemented, regression tests are performed (SWR 7). If no more faults are reported in diagnostic services at that development stage, the ECU is production mature in the execution of diagnostic services.

Figure 1:
E/E architecture and diagnostic communication on the Opel Insignia

Since a test engineer normally tests a number of different ECUs simultaneously, without adequate tool support it is impossible for the engineer to perform the large number of tests necessary to cover all of the implemented diagnostic services of the individual software versions. As a result, only newly implemented diagnostic services are tested in-depth, and test engineers perform representative regression tests for previously integrated individual services based on their experience. By using a suitable automation tool, more tests may be performed in validation while simultaneously reducing effort.

Requirements for the Validation Tool

A tool for automated diagnostic validation must satisfy the following requirements:

> Seamless integration in the existing tool chain
> Transparency and reproducibility: The test engineer must be able to track the executed tests and repeat them.
> Conformity to existing testing methods at General Motors: The tool must support existing test methods. In the diagnostic area, the GM Diagnostic Specification already defines mandatory test procedures for GMLAN Diagnostic Services of the ECUs.
> Expandability by the test engineer
> Automatic generation of test cases: The specification must exist in a machine-readable format to enable this.

From Specification to Test Execution and Report Evaluation

As shown in **Figure 2**, DiVa represents the link between CANdelaStudio (diagnostic specification) and the proven validation tool (CANoe). DiVa can be seamlessly integrated in the existing and established GME tool chain. Test cases for checking the individual services are automatically derived from the CANdela diagnostic specification (CDD file). The generated code is based on the CANoe programming language CAPL (Communication Access Programming Language) and can therefore be examined at any time. If problems occur, the test engineer can intervene in the automated test sequence and troubleshoot their causes (transparency). Furthermore, CANoe's logging functions enable traceability and evaluation of the diagnostic data flow on the CAN communication level.

The following steps are necessary to conduct a test with DiVa:

> Select the ECU and its variant
> Configure the test
> Generate the test
> Add the generated test module to the CANoe test environment
> Execute the tests
> Evaluate the test report

The user can modify test constraints in DiVa at any time. Among other things, the "Intensity" parameter is used to configure the test contents, e.g. "full test", "quick test" or "good case test". In addition, under "Supported services" the user can exclude certain services from the test or modify data contents of the services under "Data customization" (**see Figure 4**).

In updating the diagnostic specification, i.e. the CDD file, DiVa enables synchronization to the new specification while preserving previously defined settings. From a technical perspective, DiVa generates CAPL code for the CANoe test module in order to test all diagnostic services supported by the ECU. To assure conformity to the GM diagnostic specification, the DiVa extension maps the test procedures of the GM standard. The test generation process produces a detailed description of the generated test cases, CAPL test codes for the CANoe test module and the associated CANoe test environment.

Figure 2: Comparison of diagnostic validation and tool environment on the Opel Corsa and Opel Insignia

Insignia ECU	Generated Test Cases	Execution Time (w/o User Interaction for DTC Checking)
Instrument Cluster	1700	13:25 min
Climate Control	3350	39:10 min
Airbag	4630	1:19:32 min
Mobile Phone	1120	11:05 min

Table 1: Test execution times for Opel Insignia ECUs

Test execution and Report Evaluation

After the test has been generated, the user opens the generated test environment in CANoe and starts the test. The test duration depends on the complexity of the diagnostic specification and the user-defined test scope that is selected, and it may vary from just a few minutes to several hours **(Table 1)**. At General Motors, the CANoe test environment serves as a joint platform for test automation and simplifies reuse of existing GM test programs. For example, end-of-line flash test procedures are also programmed in the CANoe programming language CAPL. To simplify analysis by the test engineer, test reports are structured according to the GM diagnostic specification. **Figure 5** shows a typical test report.

Test Coverage

Automating the tests extends test coverage and simultaneously shortens the time needed for test execution. The extent to which DiVa covers the test procedures described in the GM Diagnostic Specification is described below. The quality and number of generated test cases depend in large part on the completeness of the machine-readable diagnostic specification (CDD file). All generated tests are derived from it.

A total of about 350 test sequences are defined in the GM Diagnostic Specification. The test sequences cover both "good case" and "bad case" tests. A large share (approx. 80 %) of the test procedures are covered by fully automated tests in DiVa. An application-specific user input is required for 45 (15 %) of the test procedures defined in the GM Diagnostic Specification. In such cases, DiVa pauses test execution and asks the user to put the ECU in the required state. The remaining 5 % of test procedures are not supported by DiVa and must be tested either manually or by other means. This includes tests that would put the rest of the test procedure at risk (e.g. generate EEPROM errors and detect them) or would cause long-term changes to the ECU (e.g. an ECU without calibration data).

Figure 3: Scope of diagnostic functions in various phases of ECU development at GME

Testing depth is further enhanced by including execution of additional non-GM-specific test cases.
Comparisons made at GME between validation for the Opel Corsa and for the Insignia conclude that DiVa shortens test execution time enormously by predominantly automated execution of all generated test cases, **Figure 6**. **Table 1** shows a summary of execution times and the number of generated test cases for ECUs in the Opel Insignia. Often, manual tests can only be performed sporadically due to time demands. Therefore, test results largely depend on the experience of the test engineer and the amount of time available. At GME, DiVa enables both complete testing of ECUs per diagnostic specifications and greater test coverage in all development stages.

Economic Aspects and Efficiency Increases

When a tool is introduced, its economic benefit is a primary consideration. The new Opel Corsa is very successful on the market, and there are no negative reports of diagnostically-related electronic problems. That is why the manually performed validation process on the Opel Corsa was selected as a reference project. In contrast, on the new Opel Insignia, DiVa was being used as the primary tool for validation of diagnostic services. It was used to automate a large share of validation tests for the first time. For comparison purposes, the study evaluated the time required for test execution and evaluation in the validation phase, based on representative ECUs. The values given are based on implementation level SWR 5, **Figure 3**. Most services have already been implemented at that point, and a large number of failed test cases had already been captured. **Figure 6** shows validation effort in hours for manual testing on the Opel Corsa and automated testing on the Opel Insignia.
By using DiVa, execution and evaluation times were shortened considerably on the Opel Insignia compared to the Corsa. In the studied case, 3- to 5-fold improvement was attained **(Figure 6)**. In particular, the time savings was enormous for ECUs with a large number of diagnostic services. If one considers later development phases such as SWR 6 or SWR 7, the time needed for evaluating test results is reduced even further. This can be traced back to the smaller number of failed test cases in the more mature implementation. This trend continues in each new phase up to the production launch. The production ready ECU must not exhibit any defects; consequently, the evaluation time is equal to the execution time. In this stage of Opel Insignia development, depending on the complexity of the ECU, efficiency might be increased by a factor of 20–40.
The cost of the new solution is low, since all that is needed are licenses for DiVa. A user at GME who is familiar with CANoe can perform DiVa tests – without prior training. Additional hardware is not required for test execution, since DiVa utilizes the available CAN infrastructure via CANoe.

Limitations on Automatic of Test Case Generation and Test Execution

Even if automated tools are better than manual test strategies in terms of test scope and time effort, automatic test generation does run into limitations:

> Quality of the specification: Since the specification represents the basis for generating test cases, completeness and accuracy of the specifications are

Figure 4: Setting test constraints in DiVa (here: configuration of services)

Figure 5: Automatically generated test report in DiVa

Figure 6: Test effort per ECU on the Opel Corsa with manual validation, compared to automated validation of diagnostic services on the Opel Insignia (execution and evaluation time)

essential, i.e. a test is only as good as its specification. Furthermore, there must be conformity to the requirements of the General Motors diagnostic infrastructure (GGSE-I) [6]

> Reproducibility: Due to the non-deterministic properties of CAN communication in a vehicle, certain error situations are very difficult to reproduce in testing.
> Secondary fault: In case of error, the automated test tool – in contrast to a test engineer – cannot distinguish between an initial fault and a secondary fault.
> User interaction: In application-specific tests it may be necessary to put the ECU in a state where additional hardware is necessary. These cases cannot be handled fully automatically in the approach described.

Summary

Without the use of test automation tools, it is hardly possible to achieve the desired coverage in validation of the diagnostic functionality of modern vehicles any longer. CANoe.DiVa from Vector Informatik has been adapted to GM requirements to support all established test processes, and it fits seamlessly in General Motors Europe's existing tool chain. It is used as an automated test tool for validation of diagnostic services on the new Opel Insignia.
With DiVa, GME is not only shortening test duration, but is simultaneously increasing intensity of testing by its ability to perform regression tests more frequently. Furthermore, the scope of test coverage is extended by executing additional non-GM-specific test cases. In direct comparison to manual validation on prior successful projects, both technical and economical efficiency have been increased significantly. Depending on the development phase and quality of implementation, efficiency increases by a factor of 4 to 20 are realistic. At the same time, it is possible to satisfy the high expectations of customers in terms of quality.

Literature references:

[1] Thomas, D.; Ayers, K.; Pecht, M.: The "trouble not identified" phenomenon in automotive electronics. In: Microelectronics reliability, Vol. 42, P. 641-651, 2002
[2] LIN Consortium: LIN Specification Package Revision 2.1, OV. 2006
[3] Robert Bosch GmbH: CAN-Spezifikation 2.0, 1991
[4] International Organization for Standardization: Keyword Protocol 2000, ISO 14230, 1999
[5] Krauss, S.: Testing with CANoe, Application Note AN-IND-1-002. Vector Informatik, 2005
[6] General Motors. GGSE ECU Diagnostic Infrastructure Requirements, Version 1.07, 2007

Dr. Philipp Peti
studied Computer Science at the Vienna University of Technology. He earned his doctorate degree in Computer Science, also at TU Vienna. Dr. Peti is a development engineer in the "Global Systems Engineering" group at General Motors Europe, located in Rüsselsheim, Germany.

Armin Timmerberg
studied Electrical Engineering at the University of Applied Sciences at Wiesbaden. After his studies, he was first employed as a systems engineer in the aftersales area at General Motors Europe. His primary job was to implement ECU diagnostics in the GM Service Tester TECH2. Since 2004, Mr. Timmerberg has been working as a development engineer in the "Global Systems Engineering" group at General Motors Europe, where he is responsible for diagnostic validation.

Simon Müller
studied Software Technology at the University of Stuttgart. As a product manager he is responsible for CANoe.DiVa on the Vehicle Diagnostics product line division at Vector Informatik GmbH in Stuttgart.

Thomas Pfeffer
studied Electrical Engineering at the Darmstadt University of Technology. Mr. Pfeffer is group manager for Diagnostics and Test Automation in the "Global Systems Engineering" group at General Motors Europe, located in Rüsselsheim, Germany.

Christoph Rätz
is the Director of the Diagnostics product line at the company Vector Informatik GmbH in Stuttgart.

VECTOR

Remote Diagnostics

Diagnose Vehicles Worldwide!

You are looking for a remote diagnostics solution that lets you access vehicles directly and interactively worldwide? With Indigo Remote you identify systematically the cause of a problem on test drives, at production or in the service shop despite being very far from the site.

The advantages of Indigo Remote:
> Worldwide remote diagnostics, available at any time
> Quick and easy setup
> Best data security: Diagnostic data, processes and security algorithms are not needed on site, so they are not transmitted
> Fast data transfer and very short reaction times
> Comprehensive access through hardware interfaces — even from third-party interfaces (PassThru API)

More information: **www.vector.com/indigoremote**

Remote diagnostics with Indigo Remote quickly gives you an expert opinion and shortens diagnostic and repair times. This reduces costs and increases customer satisfaction.

Vector Informatik GmbH | Stuttgart · Braunschweig · Hamburg · Karlsruhe · München · Regensburg | www.vector.com

© Claas, Vector Informatik

Automated Data-driven Validation of the Diagnostic Implementation at Claas

Using automatically generated tests for the validation of diagnostic protocol implementation has been practiced for many years. This process is based on diagnostic description files from the automotive OEM. The ECU's diagnostic interface can be efficiently tested in a verifiable way, and this leads to an improvement in product quality. In addition, automated validation of diagnostic parameters and error codes is also possible – depending on how complete the diagnostic description is – as shown in a joint project by Claas and Vector.

AUTHORS

Dipl.-Ing. (FH) Nils Niedermark
works as Development Engineer in the Development Electronic Integration Department at the Claas Self-propelled Harvesters GmbH in Harsewinkel (Germany).

Friedemann Löw, B. Sc.
is Developer for the Automotive Diagnostics Product Line at the Vector Informatik GmbH in Stuttgart (Germany).

Dipl.-Inf. Simon Müller
works as Product Manager for the Automotive Diagnostics Product Line at the Vector Informatik GmbH in Stuttgart (Germany).

TASKS

Claas, an agricultural machine manufacturer, describes the interrelationships between diagnostic parameters and ECU inputs and outputs in its diagnostic data. The description of error setting criteria is formally documented for new implementations. In the past, this information was used to perform manual tests, or test engineers implemented special test cases. However, it was still not possible to attain broad test coverage.

In cooperation with Vector, this information relating diagnostic parameters and ECU I/Os is being automatically linked to the existing network (communication matrix) and hardware description. Fully automated diagnostic implementation tests are generated and executed in a test environment based on existing specification data such as CDD (CANdela Diagnostic Data) or ODX (Open Diagnostic Data Exchange). Correct diagnostic hardware and software integration is tested by automated stimulation of the ECU environment. This is done by modifying signals in the bus simulation or driving specific hardware I/O. So it is possible to test the correct setting of diagnostic parameters, or generate error states and test for their proper memory storage.

AUTOMATED TEST GENERATION

To attain automated test generation and test execution of implementation tests, the diagnostic parameters and ECU I/O must be correlated to each other. Besides the diagnostic data (ODX, CDD), specification data that indirectly references diagnostics is also used. This might include network descriptions (dbc, arxml) or environment configurations such as the interface description of an HiL configuration. This system information can be used to stimulate and measure an ECU's inputs and outputs in a test environment.

Today, the interdependencies between the diagnostic and ECU environments are typically not described formally in the diagnostic data – if they are described at all, they are typically described in natural language. This generally makes it impossible to further process it by automated methods. Heuristics can be of help here and at least enable some use of this information for test automation. If additional detailed OEM-specific knowledge about the type and scope of the ECU I/O description exists, they can be used to derive tests of the diagnostic implementation. In particular, this enables automated diagnostic parameter tests and fault memory tests.

FAULT MEMORY TESTS

The structure and formal contents of the fault memory data are taken from the diagnostic data, such as the layout of the environment data from DIDs (Data Identifiers) and conditions for setting DTCs (Diagnostic Trouble Codes). The latter, however, usually exist in a non-formal description. If the diagnostic description can be interrelated to the ECU's periphery, then it is possible to test whether a DTC is stored correctly in fault memory and under the proper conditions. In addition, it is possible to validate DTC status transitions and proper clearing of DTCs.

The specific setting condition must be known for each DTC. This consists of at least the following:

- I/O type (input or output, network or sensor/actuator)
- I/O name (message name, channel name)
- error pattern (e.g. short circuit to ground).

The error pattern can often be derived directly from the standardised Failure Type Byte (FTB) of the DTC (SAE J2012). Depending on the error pattern, additional information might be needed: threshold values, setting times and information on error monitoring.

DIAGNOSTIC PARAMETER TESTS

Analogous to fault memory tests, diagnostic parameter tests need to know the relationship between diagnostic parameters and the ECU pins. A diagnostic value can be validated by comparing it to a:

- measured value at an ECU pin
- Bus signal
- CCP/XCP signal.

In addition to the I/O type and name, conversions are also necessary to enable comparisons – such as conversion of a resistance value at a sensor to a temperature value in the diagnostic parameter. The updating frequency of the diagnostic parameter or ECU output must also be considered. Furthermore, test values are needed for the I/O stimulation, because they are seldom documented in the diagnostic description, and suitable values generally cannot be derived from specification data.

For both fault memory tests and parameter tests there may be preconditions for the availability of a function to be tested. For instance, grinding the blades of a field chopper requires that the main drive is activated. These interdependencies must be known and considered at test execution.

GOALS AND IMPLEMENTATION AT CLAAS

At Claas, much of the data required for implementation tests are described formally. Therefore, the goal is to automate test generation and execution based on this information. To accomplish this, Claas uses the CANoe.DiVa tool from Vector. The I/O information in the diagnostic description (CDD) and from other sources is imported into CANoe.DiVa to parameterise a test generator. Afterwards, the generated implementation tests are automatically executed in the CANoe test environment that already exists for the ECU under test, **FIGURE 1**.

Claas manages all data relevant to the implementation tests in a development database. Since the data is also imported into the diagnostic description file, this file is usually sufficient to generate the parameter tests. Additional conversion information is only needed for I/O's which use different units in the diagnostic parameter than at the ECU pin. They are contributed in the form of a CANoe mapping that maps signal values to CANoe system variables via a conversion rule.

This data can be used to automatically parameterise and generate three different parameter tests:

- Input tests: The test environment stimulates a sensor pin of the ECU. The associated diagnostic parameter is read out, and it is compared to the value at the pin.
- Output tests: A diagnostic service (I/O Control) writes a new value to the ECU, which causes it to drive an actuator. Afterwards, the value at the output is measured and compared to the value of the written diagnostic parameter.

FIGURE 1 System architecture: automated diagnostic implementation tests at Claas (© Vector Informatik)

FIGURE 2 Test setup at Claas for measuring and stimulating voltages at ECU pins with the VT System from Vector (© Claas)

– Passive tests: Some signals cannot be controlled by a diagnostic service; rather they can only be read out. They are controlled solely in the ECU application bypassing the diagnostic layer. In this case, a test can be generated which reads the value at the diagnostic service and compares it to the value at the ECU input or output.

In contrast to parameter data, some of the data needed for fault memory tests at Claas do not exist fully in the diagnostic description. They are therefore exported from the development database as an Excel file and are read in for test parameterisation.

In the test derived from this data, the I/O of an ECU is stimulated to create an error situation. Afterwards, the test verifies whether the correct DTC was stored in fault memory. DTC status transitions and DTC environmental data stored for the error can also be verified by having the test correct the error situation, implement wait times and set the error state again. The fault memory tests are rounded out by checking the clearing of fault memory on different safety levels.

To measure and stimulate voltages and currents at the ECU pins, Claas uses a HiL system from Vector– the VT System, **FIGURE 2**. In order to automatically drive the VT System using the data of the diagnostic description and the development database, name conventions were defined for the VT System configuration.

INCREASING TEST COVERAGE

At Claas, diagnostics-capable ECUs are not only found in combine harvesters and tractors, but also in such machines as mowers and baling presses. In the largest Claas machines, up to 40 ECUs may be installed depending on equipment options. Implementation tests must be conducted for all of these model series. This verifies that the ECU modules can be controlled by the Claas Diagnostic System (CDS). The largest ECU installed by Claas has more than 75 I/Os, with up to 15 different machine functions implemented, depending on the installed machine options. Associated with these I/Os are over 200 DTCs that must be checked in testing.

Automating the generation and execution of more implementation tests reduces the effort required for validation of the ECUs tremendously. OEM-specific extension of the test tool at Claas enables greater test coverage by automated tests of fault memory and diagnostic parameters: from a previous 55 to 95 % currently. Claas has set a goal of subjecting all of its ECUs to automated implementation test with CANoe.DiVa.

The effort required for diagnostic tests of hardware I/O is high, despite automated test generation and test execution. Setting up the test environment is especially time-consuming. Depending on the specific ECU, it can be very complex to drive individual I/Os, and individual access must be implemented. By contrast, validating diagnostic parameters with reference to network signals can be implemented with low initial effort: The necessary infrastructure can be automatically generated by generating a remaining bus simulation from the communication matrix, which can then be used by the test.

SUMMARY AND OUTLOOK

Automated generation and execution of tests – such as those performed at Claas with the CANoe.DiVa tool from Vector – offer great potential for increasing depth of testing while reducing test effort. Currently, for full test coverage with CANoe. DiVa, some functionalities must still be manually configured at Claas, because no formal description is available that could be used for automated parameterisation. For instance, there are I/O's whose control is very complex in parameter tests. Here, some preconditions in the ECU environment must be satisfied before it is possible to use the I/O as intended. Similarly, there are also preconditions for some fault memory tests that need to be met before an error monitor becomes active and a DTC can then be recognised and stored in fault memory. The currently implemented tests can still be supplemented by further details such as testing of self-healing functionality, prioritisation of fault memory storage and to test different error situations that lead to the same DTC. The test solution will be continually developed further in these areas.

In the automotive industry, there is also a noticeable trend toward merging development data for electrical and electronic architectures. Databases and tools thereby obtain information, either directly or indirectly, that can be used to automatically test electrical installation and electronic integration with diagnostic software.

Further formalisation of data is associated with greater effort initially. Above all, formal description leads to a rewarding high optimisation potential for automated tests which is well worth the effort. Progress in standardisation (e.g. Autosar) and the integration and interoperability of development tools are leading to a wide variety of new approaches in the field of automated testability. CANoe. DiVa is following this trend and will make use of these new opportunities for deriving further automated tests of the diagnostic implementation.

The possibilities for automated, data-driven tests have hardly been exhausted, and this field will remain exciting.

REFERENCE

[1] Peti, P.; Timmerberg, A.; Pfeffer, T.; Müller, S.; Rätz, C.: Automatic validation of diagnostic services. In: ATZelektronik worldwide 3 (2008), No. 6, pp. 30-35

Automatic Diagnostic Validation is not Rocket Science

... Just a Matter of Consequent Exploitation of Existing Possibilities

Technical articles on diagnostic validation often begin as follows: Along the complexity of the ECUs in vehicles increases, the scope of diagnostics is also increasing, and with it the time and effort for diagnostic validation. However, thanks to current testing tools, the time and effort is no longer increasing linearly – fortunately. In particular, it has actually been possible for several years to generate and execute diagnostic protocol tests fully automatically. Despite a further increase in the functional scope, the time and effort for creating and executing tests remains fairly constant. With the established standards for unified diagnostic communication (UDS) and diagnostic description formats (ODX), it is possible to achieve a comparatively high degree of test automation, and high diagnostic quality as a result. However, testing of the diagnostic application and the update software can also be automated. These days there are few areas in the development of automotive electronics with a similarly high potential for test automation.

The release of the Unified Diagnostic Services Standard (ISO14229) [1] in 2006 represented a significant step for the automation of diagnostic tests: with the harmonization of the diagnostic services, it became possible for the first time to implement cross-manufacturer diagnostic protocol tests in greater depth. The previous protocol KWP2000 (ISO14230) [2] still allowed a great deal of freedom and thus must be made more concrete in an OEM-specific way, that makes generally valid test implementation difficult. A revised version of the UDS was published in 2013, which makes it possible to formulate even more detailed protocol tests.

Besides the protocol definition, formally described diagnostic data represents the second important prerequisite for test generation. The functional scope of a power window differs essentially from that of an engine ECU – and the implemented diagnostic functionality also differs accordingly. In order to be able to generate a test automatically, the diagnostic services implemented in the ECU must be known. The CANdelaStudio tool from Vector Informatik GmbH is globally established for specifying diagnostics. Among other things, it supports the international Open Diagnostic data eXchange standard (ODX, ISO 22901) [3], that was released in 2008 and is now used worldwide by most vehicle manufacturers.

Thus the technical prerequisites for automatic generation of diagnostic protocols have been met for a long time already. Therefore, it is no wonder that there have been solutions for automatic diagnostic test generation and execution for over 10 years already, such as the CANoe.DiVa tool from Vector.

Very comprehensive tests are generated automatically on the basis of diagnostic information in CANdela or ODX format and the diagnostic protocol (e.g. UDS, KWP2000, OBD, WWH-OBD, ...). These tests cover both valid and invalid requests that put the error handling in the ECU to the test. The tests are loaded into the test environment (CANoe) and executed automatically. The test results are documented in a detailed report **[Figure 1]**. For larger ECUs over 10,000 test cases are generated quickly without redundant testing.

Figure 1: Function blocks of the CANoe.DiVa automated test environment

In practice there is now a formal diagnostic description in the CANdela and/or ODX format for nearly every ECU. Therefore, it is a small step from the diagnostic specification to a comprehensive, automatic protocol test. Protocol errors can be detected for a long time simply with little initial effort.

AUTOSAR Increases Diagnostic Protocol Quality

When AUTOSAR software components are used for diagnostics, a series of protocol violations by the tester (e.g., format violations) are handled by the basic software already. Since the basic software is usually parameterized directly with the available diagnostic data, the diagnostic responses of an ECU are usually also protocol-compliant. The proportion of simple protocol errors falls noticeably when AUTOSAR components are used. The time and effort for test evaluation and troubleshooting fall accordingly. Nonetheless, the protocol test remains important for the following reasons among others:

> Errors can occur during configuration of the AUTOSAR components, integration, and application development
> If the diagnostic implementation in the ECU does not match with the diagnostic data used by the diagnostic tester, correct diagnostics in the ECU is not possible despite "correct" implementation
> Gaps in the error handling of the application implementation can potentially be used to install malicious code

Since, in times of DoIP and OTA ("over the air"), no direct physical connection with the diagnostic tester is required, a protocol error may have a negative effect on these services for an entire fleet of vehicles, for example, due to limited options during remote maintenance, because the required diagnostic functions do not work as desired, so time in the workshop is necessary (after all). In the worst case, a protocol error could permit a safety-critical attack on the vehicle. Due to continuous availability of the vehicle, new use cases that the customer can experience will develop based on diagnostic functionality, which will presumably lead to more frequent and broader use of the diagnostic functions. It is thus foreseeable that the quality requirements on the diagnostic implementation will continue to increase.

Software Update Validation

Despite all the diagnostic standardization measures, the processes for updating ECU software (flashing) remain OEM-specific. Although the services used are standardized, functions such as incremental update or mechanisms for protecting the transmitted data (identification data, checksums, signatures, etc.) lead to quite different flash procedures in practice. Furthermore, different flash container formats are used in order to meet the process requirements of the respective manufacturers. In practice that means that there is also a specific software update tool for nearly every OEM. Therefore a supplier must not only work with various tools but often also develop and maintain a specific test environment for each OEM. A functioning flash procedure has always been indispensable in production and service. Taking future software updates "over the air" (SOTA) into consideration, a reliable software update function in vehicles is more important than ever. With the new possibilities, new software will presumably be loaded more frequently – possibly at a frequency similar to what we are already familiar with for mobile devices (e.g., to fix security vulnerabilities). Situations in which an ECU no longer functions after a failed update should definitely be avoided, of course. But even for established smartphone manufacturers, software updates do not always run smoothly – despite sophisticated protective measures.

Cross-manufacturer software updates have been performed for several years already with the reprogramming tool vFlash from Vector, currently for over 90 different boot loader specifications of all relevant vehicle manufacturers. The test automation tool CANoe.DiVa uses the vFlash automation interface and thus additionally offers automated software update tests for all boot loaders supported by vFlash. Besides the variables (timing, format, ...) in the valid flash procedure, classical error situations are also tested automatically, such as undervoltage and interruptions at various points in the flash procedure. This allows the robustness of the ECU software to be tested easily and comprehensively.

The tests sketched are essentially black box tests. Tests beyond these, e.g. plausibility checks for process-relevant

data such as identification data or signatures, require manufacturer-specific detailed knowledge. There are already specific extensions for well-known manufacturer for this type of test.
The software update tests ensure the essential update features such as reliable data transfer and robust behavior in the event of errors. The convenience of automation for them is similar to that for the diagnostic protocol tests.

Validation of the Diagnostic Application

The diagnostic functions in the ECU must not only satisfy formal aspects but also have correct content, of course. This is the only way to ensure sensible vehicle diagnostics. The current diagnostic formats (e.g., ODX) focus especially on the definition of the protocol aspects. Nonetheless, application-relevant information can be extracted from them for automatic test generation: for example, an automatic test of the plausibility of the transferred values can be derived from the description of the specified value ranges. Specific error causes can be derived from informal error codes or diagnostic parameters with heuristic methods and/or manufacturer-specific knowledge. In order to be able to put the insights obtained in specific automatic test runs into practice, the options for access to the (test) environment, and thus the options for simulation of the ECU, must be known. However, this information is usually available during ECU development in any case:

> The network architecture is described in AUTOSAR (.arxml) or CANdb (.dbc). Signals can be read from the bus easily and manipulated through a remaining bus simulation.
> The memory layout of an ECU is described in a2l files. Parameters can be read and manipulated easily via XCP.
> The electrical inputs and outputs and the test configuration are usually described in machine-readable formats. These can be measured and triggered, e.g. through a Hardware-in-the-Loop (HiL) test in combination with the Vector VT system **[Figure 2]**.

Figure 2: CANoe.DiVa: Software and hardware in-the-Loop

Furthermore, CANoe.DiVa supports the coupling of proprietary parameters and DTC descriptions through an Excel exchange format. In an integrated test environment like Vector CANoe, all these sources can be combined and used in one test.
When a virtualized ECU is used in development (e.g., Vector vVIRTUALtarget), the test setup for the application test is comparatively simple, since hardware I/Os can be simulated easily in software.
The degree of possible automation during application tests certainly does not reach that of protocol tests; however, numerous options arise for semiautomatic or even fully automatic test generation, which it pays to take advantage of.

Test Scope, Test Evaluation, and Further Processing

The high degree of automation for diagnostic protocol validation achieved through corresponding tools permits a greater test depth through further automated tests or additional tests of the user's own – with constant time and effort: more and more OEMs are taking advantage of the automation options in order to systematically safeguard the diagnostic application cases, e.g. for production and service, that are especially important to them. Correspondingly, the diagnostic function can be demonstrably safeguarded in early development phases already by the supplier.
Many of the largest OEMs around the world now rely on the test support of CANoe.DiVa. In the process they also benefit from the comprehensive support for further processing of the test results, which in practice saves a great deal of time **[Figure 3]**:

Figure 3: Practical experience: testing time and effort for manual validation compared to automated validation [5]

> Sorting and filtering, for a targeted and needs-based view of the test resultsThe memory layout of an ECU is described in a2l files. Parameters can be read and manipulated easily via XCP.
> Linking errors with the same cause
> Adding comments to individual test results in order to classify the error and its cause, documenting notes on corrective measures, and controlling troubleshooting.
> Transferring the error analysis results from one test run to another
> Connecting the test solution to existing test data management or requirement systems for integration into existing processes

Conclusion/outlook

The diagnostic scope of ECUs and the quality requirements will continue to increase. However, automated solutions already exist today that significantly reduce the time and effort for diagnostic validation and, at the same time, significantly increase the test breadth and depth. The formally described data required for this is usually available in any case. The reuse of this data for test purposes as well is simply logical.
With the increase in networking beyond the boundaries of the vehicle, the visibility of the topic of automotive cyber security is also increasing rapidly. "Testing security" and "testing despite security" [4] have now become topics of focus in many areas of development. Expanded vehicle access "over the air" opens up new application domains for vehicle diagnostics – which must be safeguarded in turn. Many innovations can only be brought into serial production if they are supported in software and the central tools from early on in an user-friendly way. This is not rocket science either – just an exciting challenge for software vendors.

Literature References:

[1] - [3] International Organization for Standardization: ISO 14229 (UDS), ISO 14230 (KWP2000), ISO 22901 (ODX)
[4] Metzger E.; 2016: Presentation: The Vector Security Solution, Vector Cyber Security Symposium 2016
[5] Peti P., Timmerberg A., Pfeffer T., Müller S., Rätz C.; 2008: Automatic validation of diagnostic services, ATZ elektronik 06I2008

Christph Rätz
Christoph Rätz is the head of the Diagnostics product line at Vector Informatik GmbH in Stuttgart.

Simon Müller
Simon Müller studied software engineering at the University of Stuttgart. He is the product manager responsible for CANoe. DiVa at Vector Informatik GmbH in Stuttgart in the Diagnostics product line.

Translation of a German publication in Hanser Automotive issue 3-4/2017

Image rights: Vector Informatik GmbH

The Standard Mix does it:

Diagnostics with AUTOSAR and ODX – Part 1: Diagnostics with AUTOSAR

AUTOSAR is the future-oriented reference architecture for ECU software. Clearly specified interfaces, standardized behavior and XML-based data formats are the key features of this standard. In AUTOSAR, diagnostics is handled in the modules DCM (communications) and DEM (fault memory). This article first addresses diagnostics in AUTOSAR and related data formats. Description data in ODX format (Open Diagnostic Data Exchange) represents an alternative to configuring the diagnostic software. Part 2 will address the topic of "ODX in the AUTOSAR development process".

Standardization is very much the trend in the development of automotive electronics. The use of open architectures and configurable components are intended to let developers focus more on the innovative and differentiating aspects of the development process. In addition, standardization is intended to be a central measure that contributes towards reducing costs. In the past, automotive ECU software architectures were not standardized. For the supplier, this resulted in different, OEM-specific software architectures that required different development processes, development tools and data exchange formats. AUTOSAR (AUTomotive Open System ARchitecture) has the stated goal of standardizing a common, open automotive software architecture. The primary goals of the AUTOSAR architecture are:

> Hardware abstraction
> Clearly specified interfaces
> Standardized behavior of the basic software
> Standardized data exchange formats between OEM and suppliers
> Definition of a harmonized methodology for developing software
> Support of model-based functional development
> Scalability over all ECU and vehicle classes
> Considers safety requirements per ISO 26262

Today, AUTOSAR is the reference architecture for ECU software. The first production launches with complete AUTOSAR software will occur in the near future. The number of development projects utilizing AUTOSAR methodology is continually growing.
The AUTOSAR Consortium is currently working on versions 3.2 and 4.x. Versions 2.x, 3.x and 4.0, which were released prior to this, have already been used as a basis for implementing vehicle projects. Today, most vehicle producers work to versions 3.x.

Functional orientation is increasingly gaining in significance in electronic development. AUTOSAR standardizes the description of individual component or vehicle functions and the description of the overall system in what is known as the System Configuration Description. The methodology for distributing vehicle functions to ECUs is also standardized. As a result, developed functions can be reused in other vehicle projects without changes.
The example in **Figure 1** illustrates this: The Lighting vehicle function from the Function Library is subdivided into three subfunctions. In vehicle A, the subfunctions are distributed to two network ECUs, while in vehicle B they are distributed unchanged to three ECUs. The communication between the subfunctions is defined in the System Configuration Description.
In AUTOSAR, there is an ECU Extract of the System Configuration for each ECU that covers the system content relevant to the ECU – and often also relevant to a supplier. The elementary components of AUTOSAR architecture for ECU software are:

> Functional software (SWC)
> Run-Time environment (RTE)
> Basic software (BSW)

The high level of reusability of the functional software is due to the abstraction of communication by the Virtual Function Bus (VFB). The application can be developed and tested without knowledge of the underlying communication mechanisms. It does not matter here whether communication occurs within the ECU or over a network (CAN, FlexRay, etc.). The Run-Time-Environment (RTE) serves as the runtime environment for the functional software, and it implements the Virtual Function Bus for a specific ECU. The basic software is developed as a component kit and is commercially available (off-the-shelf software). It contains fundamental system functions and abstracts the functional software from the hardware. It is subdivided into three areas **(Figure 2)**:

> The Service Layer provides basic services for the functional software and other basic software modules.
> The ECU Abstraction Layer abstracts higher layers from the ECU hardware
> The Microcontroller Abstraction Layer abstracts higher layers from the specific microcontroller device

The ECU Configuration Description is used to configure the basic software and the RTE. Initially, this configuration is generated from the ECU Extract of the System Configuration Description (e.g. communication over the network). The ECU Configuration Description plays a central role for the behavior of the entire ECU software and is extended and adapted, step by step, over the course of further development.

Diagnostics with AUTOSAR

The diagnostic software in AUTOSAR consists of three modules: DCM, DEM and FIM. The DCM (Diagnostic Communication Manager) implements the diagnostic communication per ISO 14229-1 (UDS) and SAE J1979 (OBDII). All diagnostic requests are first preprocessed by the DCM. One of the tasks of the DCM is comprehensive handling of invalid diagnostic requests. The DCM can already fully process a majority of valid requests; it routes other requests to the functional software. Each AUTOSAR release has in-

Figure 1: Functional distribution in AUTOSAR

Figure 2: Structure of the basic software (BSW)

creased the functional range of the DCM, while continually decreasing the remaining diagnostic content of the functional software. Handling of a DID **(Figure 3)** illustrates this development. Up to Version 3, signal structures had to be resolved in the functional software. In Version 4, this task can also be handled by the DCM.

The DCM is configured based on a ECU Configuration Description. This includes the service identifiers, subfunctions, data identifiers (with associated signal structure) and routine identifiers (with parameter lists). In addition, execution of diagnostic requests can be made dependent on the current ECU state (session and security level).

The DEM (Diagnostic Event Manager) implements an error memory. Up to (and including) AUTOSAR Version 3.x, the DEM is only specified as a facade, because details of error memory behavior are OEM-specific. Since Version 4, the goal has been to standardize an OEM-independent error memory, so that its behavior can be defined in AUTOSAR.

The DEM has the following primary tasks:

> Administer the DTC status bit
> Organize error storage, including NVRAM
> Organize snapshot data (freeze frame)
> Administer extended data records
> Provide for unlearning of errors
> Provide an interface for error readout for the DCM

Figure 3: DCM in different AUTOSAR versions

A standardized interface and various debounce algorithms for diagnostic monitors (error path) enable uniform and cross-project development of the functional software. One or more error paths can be mapped to a diagnostic trouble code (DTC). The DEM is also configured from the ECU Configuration Description. It contains information related to error paths, DTC numbers and the structure of extended error data (snapshot and extended data records).

The FIM (Function Inhibition Manger) makes it possible to inhibit the execution of certain functions in case of active errors, start substitute functions and suppress secondary errors. The FIM is also configured from the ECU Configuration Description.

Basic Software Modules for Diagnostics with AUTOSAR

Vector's MICROSAR product line provides an AUTOSAR solution for ECU software consisting of the RTE and basic software modules that cover the entire scope of the AUTOSAR standard. Each AUTOSAR BSW module is assigned to a MICROSAR package. The MICROSAR DIAG package is specially available for diagnostics. It contains the three BSW modules DCM, DEM and FIM from the AUTOSAR architecture. MICROSAR DIAG as the diagnostic software provides vehicle projects with an AUTOSAR-compatible implementation of the UDS protocol ISO 14229-1:2006.

Note: Part 2 "ODX in the AUTOSAR development process" is also available for download at www.vector.com/downloads/.

Translation of a German publication in Hanser Automotive, 10/2011

Literature:

[1] AUTOSAR specifications: www.autosar.org
[2] Pascale Morizur, Matthias Wernicke, Justus Maier: Neue Wege zur Steuergeräte-Software Teil 1, Elektronik automotive 11.2009
[3] Pascale Morizur, Matthias Wernicke, Justus Maier: Neue Wege zur Steuergeräte-Software Teil 2, Elektronik automotive 12.2009
[4] ISO 14229: Road vehicles – Unified diagnostic services (UDS)
[5] ISO 26262: Road vehicles – Functional safety
[6] ISO 22901: Road vehicles – Open diagnostic data exchange (ODX)
[7] Klaus Beiter, Oliver Garnatz, Christoph Rätz: Gesetzliche On-Board-Diagnose und ODX, Diagnose in mechatronischen Fahrzeugsystemen III S. 44 ff., Expert-Verlag 2010

Dr. Klaus Beiter
leads a development team for the Automotive Diagnostics product line at the company Vector Informatik GmbH in Stuttgart. He is a member of the ASAM/ISO ODX working group.

Oliver Garnatz
is employed at Vector Informatik GmbH as a product manager in the Embedded Software Components area. He is a member of the Automotive Diagnostics area of ISO and the AUTOSAR area.

Christoph Rätz
is the Director of the Diagnostics product line at the company Vector Informatik GmbH in Stuttgart.

The Standard Mix does it:

Diagnostics with AUTOSAR and ODX – Part 2: ODX in the AUTOSAR Development Process

The Open Diagnostic data eXchange (ODX) format is an XML-based data format for describing the data relevant to vehicle diagnostics. ODX was conceptualized as an open format for exchanging diagnostic data between automotive OEMs and their suppliers. AUTOSAR is the future-oriented reference architecture for ECU software. Clearly specified interfaces, standardized behavior and XML-based data formats are key features of the AUTOSAR standard. This is the second article of the "Diagnostics with AUTOSAR and ODX" series, and it addresses the topic of ODX and how available ODX data can be profitably integrated in AUTOSAR development.

ODX was standardized in the framework of an ASAM/ISO working group, initially in ASAM since 2003 and later in ISO. The necessity of ODX development resulted from the lack of acceptance of the previous standard for describing diagnostic data. The exchange of diagnostic data beyond process boundaries was only possible with tremendous effort. A key goal of ODX standardization is data reuse. It should be possible to use and further process the data with different tools – including in different business areas.

The ODX data model in Version 2.2.0 consists of seven sub-models **(Figure 1)**. The focus of standardization activities was on parameterizing diagnostic testers. Therefore, the lower three submodels with definition of diagnostic services, communication parameters and a description of vehicle accesses represent the real core of the standard. At the same time, they form the typical content that is required for tester communication with one or more ECUs, including data interpretation.

The flash container, ECU configuration, function-oriented diagnostics and so-called Multiple ECU Jobs are described in the upper four sub-models. Their processing and significance are lower compared to the first named sub-models.

In this article, only ODX-D and ODX-FD will be discussed in depth, because these two categories are of special interest with regard to AUTOSAR. ODX-D contains the service description, which defines diagnostic requests and associated responses together with interpretation of the transmitted data.

ODX-FD is an extension to ODX-D, in which diagnostic-relevant aspects of vehicle functions can be described. Functions can be hierarchically structured and grouped according to any desired criteria. Input/output parameters and diagnostic data (e.g. DTC, DID, etc.) may be allocated to each function. This data is assigned specific values and is allocated to diagnostic services via references in the ODX-D section. Essentially, ODX-FD documents vehicle diagnostics from the perspective of functions. If problems occur in a vehicle function, the ODX-FD data can be used to determine the relationship between the function and potential error sources – i.e. ECUs, sensors and actuators.

ODX was released as ISO standard 22901-1 in 2008. ASAM published the first version of the standard as ODX 2.0.0 in 2004. Before ISO release, two other ASAM releases were issued into which corrections, explanations, improvements and extensions flowed **(Figure 2)**.

ODX and ECU Software

ODX gives the author of diagnostic data wide-ranging freedom with regard to the structures used. One and the same behavior can be described differently. This lets users optimally prepare diagnostic data for use in specific test systems. Nonetheless, support for all conceivable variations of the standard in processing tools continues to be more of an aspiration than reality. It is possible to exchange data, provided that the structures used are supported in both worlds. A commonly used method for documenting the exchangeable contents are authoring guidelines. They specify the type and scope of the ODX subset to be used for the process partners. This approach is established today. The automotive OEMs who participated in ODX standardization also took up the process and created an authoring guideline for data exchange between automotive OEMs (ODX-RS, Recommended Style).

The main motivation for ODX standardization was the desire to standardize the parameterization of data-driven test systems. The data's usability in other application areas is limited, because the different application areas place different requirements on the structure and degree of detail. A generic tester is expected to support as many vehicle configurations or ECU configurations as possible. A multiple or ambiguous description of tester data gives the user flexibility here. For example, in ODX it is possible to describe multiple ECU responses to one diagnostic service. At runtime, the appropriate response is utilized to decode the diagnostic data. This is especially helpful if it is not entirely clear which specific software is running on the ECU. On the other hand, unambiguous and exact data description in specification quality is essential for code generation. It is obvious that the description with multiple responses cannot be used to generate the ECU software, because the ECU must react unambiguously (in a defined way) to a diagnostic request. The example shows that requirements for the (quality of) diagnostic data are different – even contradictory – for the two use cases.

Therefore, if the diagnostic software components will be generated based on ODX, the parts of the standard that do not conform to the requirements cited above (specification quality) must be excluded.

The following list identifies some data configurations that violate the specification character.

> Multiple responses to one diagnostic request (see above).
> Diagnostic services that are not defined for the underlying protocol, e.g. KWP services in a UDS-ECU.
> Multiple diagnostic services with the same service signature (SID/LID), making it impossible to derive clearly defined ECU behavior.

Figure 1: ODX categories

Figure 2: ODX history

Figure 3: Parameterization of test systems via ODX (left). Parameterization of Software Components via ODX using authoring guidelines (right)

> Use of special context conventions in error memory: the standard does not aim to provide a detailed description of error memory in ODX. In principle, it is possible to describe supplemental information for DTCs, but the standard only specifies the format here (SDG = interleaved list of name-value pairs). The semantics of the data, on the other hand, are not defined; therefore, generic processing in automated tools is not possible.

> The widely used ODX Version 2.0.1 lacks a mechanism for describing the dependency of a diagnostic service on session/security levels. The related executability tests and resulting rejecting responses cannot be generated, rather they must be implemented in the individual application. In the ODX 2.2.0 version, this problem no longer exists. Status information can be formally described here.

The list shows that conformance to the ODX standard is necessary but insufficient to parameterize software components. Checker rules defined in the standard primarily cover the use case of tester parameterization. To assure specification quality, numerous consistency checks are necessary, which must exclude data constellations such as those identified here.

In summary, the following picture emerges: ODX was designed to fulfill the requirements necessary for parameterizing test systems (see **Figure 3, left**). However, parameterization of software components assumes that the possible degrees of freedom are limited to the degree required by a specification (see **Figure 3, right**). This can be achieved by means of authoring guidelines.

Figure 4: Combination of ODX and AUTOSAR

AUTOSAR with ODX

ODX and AUTOSAR are established standards for developing ECU software or describing the diagnostic data of a vehicle or individual ECUs. It therefore makes sense to determine how available ODX data might be integrated in the development of the diagnostic content of the ECU software (DCM/DEM).

AUTOSAR development is very function-oriented (see Part 1 of this series of articles in the last issue, 10/2011). In early phases of development, it is therefore functional descriptions and definitions that are primarily created. ODX-FD bridges the gap between an ECU's functions and diagnostic content, but it is primarily relevant to testers. ODX-FD data can therefore be derived from AUTOSAR functions, even if the concrete diagnostic description does not exist yet in the form of ODX-D data (see **Figure 4, step 1**). The ODX-FD description that results reflects the structure and grouping of AUTOSAR functions in ODX. Linking in the ODX-D container (i.e. mapping between functions and the specific diagnostic data) is still not possible at this time point.

It was shown above that the information needed to configure software components can primarily be found in ODX-D. In AUTOSAR, the ECU configuration is described in the ECU Configuration Description, from which the ECU software is also generated. It therefore makes sense to transfer the ODX-D data (if it exists) to the ECU Configuration Description and use it in the AUTOSAR process. Whether and to what extent ODX-D data exists depends on the cooperation model between the automotive OEM and suppliers. An extreme case is the new development of an ECU "from scratch" (see **Figure 4, step 2a**). In this case, a large share of the diagnostic content is prescribed by the OEM. The other extreme case involves integrating an existing ECU in a new vehicle (see **Figure 4, step 2b**). Changes to the diagnostics are then only possible with tremendous effort. The diagnostics are therefore influenced much more by the ECU than by functions.

In general, neither extreme is exclusively applicable, rather the different approaches are combined. Typically, diagnostic requirements are specified between automotive OEM and the supplier from the functional perspective and ECU perspective (and the perspective of its periphery), to finally yield the ODX-D data for the ECU.

In the next step, ODX-FD data can be linked to ODX-D data (see **Figure 4, step 3**). From the ODX-D data, the ECU Configuration Description is generated, which then serves as the foundation for creating the software components (see **Figure 4, steps 4, 5**). Furthermore, the ODX-FD and ODX-D data form the foundation for creating the tester run-time format (see **Figure 4, step 7**). The use of ODX as a foundation for both aspects of the process (software components and tester parameterization) ensures that different development versions of the tester and ECU will match one another precisely.

The question arises whether the reverse process is also possible, i.e. generating ODX-D from the ECU Configuration Description. The answer depends in part on the AUTOSAR version being used: The AUTOSAR format of versions up to and including 3.x is not powerful enough to describe the key information needed for tester parameterization, e.g. it lacks conversion information for data objects. AUTOSAR 4 is more powerful and may contain this conversion information. Nonetheless, this information in particular is usually irrelevant to the use case of ECU parameterization, so it is questionable whether this information is actually described here in practice.

In addition, the function-driven approach prevents cross-vehicle harmonization of the diagnostic contents as described in this article. Therefore, it remains to be seen which direction future diagnostic data flows will take. Experience suggests that pure forms of the discussed approaches will not prevail, but instead they will be adapted to the specific development situation and combined.

Integration

Integration of the different subprocesses with their various interfaces (interfaces, data formats, etc.) is one of the greatest challenges in introducing new technologies such as AUTOSAR and ODX. Prior experience suggests that the most efficient approach is to rely on practice-proven solutions in introducing these technologies. Vector offers comprehensive AUTOSAR and ODX tool chains from a single source. You will find more information on this subject at: www.autosar-solutions.de and www.odx-solutions.de.

Note: Part 1 "Diagnostics with AUTOSAR" is also available for download at www.vector.com/downloads/.

Translation of a German publication in Hanser Automotive, 11/2011

Literature:

[1] AUTOSAR specifications: www.autosar.org
[2] Pascale Morizur, Matthias Wernicke, Justus Maier: Neue Wege zur Steuergeräte-Software Teil 1, Elektronik automotive 11.2009
[3] Pascale Morizur, Matthias Wernicke, Justus Maier: Neue Wege zur Steuergeräte-Software Teil 2, Elektronik automotive 12.2009
[4] ISO 14229: Road vehicles – Unified diagnostic services (UDS)
[5] ISO 26262: Road vehicles – Functional safety
[6] ISO 22901: Road vehicles – Open diagnostic data exchange (ODX)
[7] Klaus Beiter, Oliver Garnatz, Christoph Rätz: Gesetzliche On-Board-Diagnose und ODX, Diagnose in mechatronischen Fahrzeugsystemen III S. 44 ff., Expert-Verlag 2010

Dr. Klaus Beiter
leads a development team for the Automotive Diagnostics product line at the company Vector Informatik GmbH in Stuttgart. He is a member of the ASAM/ISO ODX working group.

Oliver Garnatz
is employed at Vector Informatik GmbH as a product manager in the Embedded Software Components area. He is a member of the Automotive Diagnostics area of ISO and the AUTOSAR area.

Christoph Rätz
is the Director of the Diagnostics product line at the company Vector Informatik GmbH in Stuttgart.

WWH-OBD – made simple!

Implementation of the new Requirements for OEMs and Suppliers

All new vehicle registrations for heavy-duty vehicles must conform to the requirements of the Euro-VI emissions standard starting in 2014. Vehicle manufacturers are obligated to implement a WWH-OBD capable diagnostic system. Even earlier, guidelines for newly developed vehicles take effect; here the deadline is already on 01-01-2013. Consequently, help in testing the implementation with WWH-OBD capable diagnostic tools is very welcome.

On-Board Diagnostics (OBD) in vehicles involves continual self-monitoring of the system. Indicator lights immediately show the results of this self-monitoring. These results are also saved and read-out by an external tester at discrete time points, e.g. during maintenance or in the context of a repair. After some carmakers already began to implement such concepts as proprietary form back in the 1980s – at that time nearly exclusively in the electronic engine management systems – the California environmental authorities (CARB) in the USA formulated registration requirements for new vehicle models. These requirements prescribed, in addition to conformance to specific emission limits, a defined set of self-monitoring functions for emissions-related systems in a vehicle.
Building upon the original OBD version of 1988, legally binding OBD content was defined in subsequent years in the rest of the states in the USA as well as in Europe and Japan. Although these regionally applicable OBD versions all build upon the same foundation – and in some cases are identical in content – they were developed in parallel to OEM-specific diagnostics and use very different methods, services and parameters.

Global Harmonization of Vehicle Diagnostics

In OEM-specific diagnostics, there has been a clear trend over the past decades towards increasing standardization with regard to transport and diagnostic protocols. Meanwhile, a uniform diagnostic interface to external testers (16-pin OBD socket) and the CAN bus as the transmission medium have largely prevailed.
This convergence towards UDSonCAN (Unified Diagnostic Services) makes it attractive to merge the diagnostic protocols of the legally required and the OEM-specific diagnostic contents, which World Wide Harmonized On-Board-Diagnostics (WWH-OBD) should be able to perform.
Standardization is occurring at the urging of the United Nations in the form of a Global Technical Regulation (GTR) and will be specified in the ISO 27145 standard.

The new specification also provides for new and extended functions. For example, the error codes are classified into Severity Classes A, B1, B2 and C, indicating the severity of a failure with regard to its effect on exhaust emissions quality. Errors of the second highest class B1 also switch to the highest class A if they are not corrected within a defined time frame that is monitored by the system, e.g. 200 operating hours. To ensure that emissions-related functional faults and their effects on exhaust emissions immediately recognizable to the driver and to authorities and test organizations, the "Malfunction Indicator Lamp (MIL)" is activated by errors of the different severity classes in different ways.
To also take future requirements and technical progress in vehicle networking into account, the Internet Protocol (IP) is included as a transport medium in standardization in addition to CAN. So, in the future UDSonIP will also be permitted in implementations of WWH-OBD.
In some leading industrial nations, the implementation of WWH-OBD will soon be required as a vehicle registration condition for newly developed models of heavy-duty vehicles. Effective at the beginning of 2014, for example, all newly registered heavy-duty vehicles in the EU must conform to the Euro-VI standards and must therefore be diagnosable via WWH-OBD. Newly developed vehicle models must already fulfill the standards by 01-01-2013.

There are plans to expand the scope of validity of WWH-OBD to passenger cars, vans, self-propelled work machines, etc. Also being discussed is the elimination of limits to exhaust-related systems. For example, it could also be used to monitor safety-related systems such as brake, steering and restraint systems and lighting and examine them in the framework of periodic tests.

Intelligent WWH-OBD-capable Tools

Vehicle OEMs must verify the correctness and completeness of OBD functionality in suitable form for registration authorities. Checking is performed in the framework of homologation and assumes that suitable tools are available for verification.
Both the vehicle OEM and system supplier need suitable diagnostic tools to test OBD functions in advance of homologation, i.e. during development or system integration.
The good news here is that the information exchanged between the vehicle and external tester in the framework of a WWH-OBD is largely identical to the information that is already exchanged in diagnostics per OBD-II or EOBD. Here, the new standard only specifies minor extensions of functionality. What will change is primarily only the way in which these contents are transported.
The graphic in **Figure 1** illustrates the difference in OBD communication based on the previous standard and the

Figure 1: Advantage for the user in diagnostic access: Intelligent tools automatically detect which standard is being used, and they automatically adapt to it and deliver the results.

new WWH-OBD standard. What is displayed is the readout of currently existing but not yet confirmed Diagnostic Trouble Codes (DTC), i.e. the "pending DTC". In both cases, the vehicle transmits the same error P0198 "Engine Oil Temperature Sensor High" to the scan tool. In the case of WWH-OBD, however, additional information, such as the severity of the error in terms of emissions behavior, as well as its complete status information is also transmitted and displayed.

When the byte contents of requests and responses are examined, it becomes clear that their structure appears different in WWH-OBD than it does in OBD-II: First, it is apparent that the length of error codes in the WWH-OBD is specified as three bytes compared to two bytes for OBD-II, where the first two bytes from SAE J2012 or ISO 15031-5 continue to be used.

The older OBD version reads out the DTCs separate from their status in different OBD modes, i.e. with:

> Service $03 — Request Emission-Related Diagnostic Trouble Codes for confirmed DTC,
> Service $07 — Request Emission-Related Diagnostic Trouble Codes Detected During Current or Last Completed Driving Cycle for pending DTC and
> Service $0A — Request Emission-Related Diagnostic Trouble Codes with Permanent Status for permanent DTC),

WWH-OBD, on the other hand, uses the protocol service ReadDTCInformation [0x19] with the sub-function reportWWHOBDDTCByStatusMask [0x42] and a bit field in Byte 4 of the request to specify the status of the error codes which are to be reported.

An intelligent test system, such as Indigo from Vector, is able to independently recognize the standard upon which the test is based, provide the relevant functions for the pertinent OBD standard and in the process make the protocol-specific differences transparent to the user. This lets users focus on the contents in their work. If necessary, the user can also delve into the depths of the protocol and analyze the raw data. This is useful, for example, in determining the causes of errors, if the ECU does not provide the expected response to an OBD Request.

Summary

Modern high-performance diagnostic tools must offer full support of the new WWH-OBD standard – and they must do so immediately. They can very well help to achieve compliance to regulations within the rapidly approaching deadlines of the beginning of 2013 and the beginning of 2014. They fit seamlessly into existing tool chains and make the migration to WWH-OBD especially simple for vehicle OEMs and suppliers. Experience has shown that it is best to rely on the most efficient and practice-proven solutions. Vector offers the easy-to-use diagnostic test tool Indigo, which already supports the WWH-OBD standard today. To ensure that the change to WWH-OBD is made very simply!

Helmut Frank
has worked on many different projects in the automotive industry, including a position as product manager and key account manager in the area of service diagnostic equipment. Since October 2005, he has been employed at Vector Informatik as Business Development Manager for the Diagnostics product line.

CANdelaStudio – Efficient Specification of Diagnostic Requirements at KTM

Case Study KTM

The Customer

The KTM-Sportmotorcycle AG with registered office in Mattighofen, Austria produces and develops race-ready offroad and street motorcycles. KTM is the largest motorcycle manufacturer in Europe and a leader in the offroad competition sector. In MY 2014, the 1190 Adventure has been turned into what is currently the world's safest motorcycle.

The Challenge

Consistent creation, management, and distribution of diagnostic data throughout the diagnostic development process.

In the development of the new 1190 Adventure, and its up to 9 ECUs, an optimization of the diagnostic development process should be performed. In previous development projects, diagnostics was specified in text form at KTM. Automated further processing of the specification was therefore not possible. Rather, the ECU specification required time-consuming interpretation and conversion into input data for the utilized diagnostic testers both internally at KTM as well as at the ECU suppliers. Thus diagnostic testers in development, production, and service became available only at a late stage.

The Solution

The CANdela approach – a standardized diagnostic data format and the corresponding tool chain.

CANdelaStudio is used as a central component of the CANdela approach in the development of diagnostics for the new 1190 Adventure motorcycle. The KTM developers use this to specify diagnostic data in machine-processable CDD format (XML). Thus these data are available at an early stage as input data for many tools throughout the diagnostic development process. Text-based specifications can be generated also out of the CANdela description and forwarded as needed to KTM suppliers still at the same time.

The Advantages

A standardized diagnostic data format and the Front-loading Appraoch provide the KTM developers time for additional testing and thus a higher product quality.

With CANdelaStudio, KTM significantly reduces its effort for the diagnostic requirements specification phase. ECUs with diagnostic capability are available at an early stage. Diagnostic specifications can be used, on the one hand, for automated generation of ECU software or for parameterization of diagnostic tester. On the other hand, they are available for reuse in new or further ECU development projects. The result: The entire diagnostic development for the ECUs of the KTM 1190 Adventure motorcycle was completed much faster than for earlier motorcycle models.

More benefits:

> Existing diagnostic tester at KTM, such as CANoe, can be parameterized at an early stage in the process
> ODX data for parameterization of KTM production and service testers are generated directly from CANdelaStudio at the press of a button. KTM developers do not have to be specially trained in ODX.
> Easy creation of diagnostics requirements as PDF data for "smaller" KTM suppliers is still possible.
> Coordination of diagnostic specification between the process participants (KTM and suppliers) is much easier.

From Diagnostic Requirements to Communication

Standardization is the Trend in the Development of Automotive Electronics

A key aim of open architectures, configurable components and harmonized exchange formats is to let developers focus more on the development and reuse of innovative and product differentiating functions. In recent years, a number of independent standards have been created, all of which have affected processes and tools for diagnostic development – in particular ODX and AUTOSAR. At the same time, the systematic capture, management and tracking of requirements took hold, which also had a significant impact on processes, methods and tools.Is it possible to do without one or more standards? Is there a super-standard? Or can the standards and methods be combined with one another effectively and efficiently?

Requirements Engineering

The development of a system starts with the requirements from the user's perspective. The capture of requirements marks the beginning of an iterative process **(Figure 1)**, in which requirements of a system are progressively made more specific and precise. If the solution space for fulfilling requirements is still large, the later specification describes individual subsystems precisely and without ambiguities.

In practice, requirements differ in terms of how specific and precise they are. Text-based requirements describe a system property to be fulfilled in text form, usually incompletely and purposely fuzzy, phrased or just in note form. Specification requirements, on the other hand, are precise and not only describe the requirement itself, rather they also include the solution and leave very little freedom for the specification. Formal languages are often used for the description, which are appended to text-based requirements in files. Reference requirements contain a reference to a specification, e.g. "as in the previous system". Technically, these reference requirements actually reference specifications in other databases or data management systems in many cases.

Ideally, requirements are defined as precisely as possible from the start, but only as specifically as necessary. Unclear or ambiguous requirements lead to considerably increased effort over the course of the development process, because clarification means there is a need for additional coordination, and it often results in a specification change. In the least favorable case, the system implementation may even need to be modified. On the other hand, requirements that are unnecessarily specific can often actually obstruct the path to the quickest and most cost effective solution. If aspects of a solution path are intermixed with requirements early on, this unnecessarily reduces the solution space. Often, this also eliminates the opportunity for re-use. Especially when requirements change over the course of development, it is important to separate the substantial requirements from relicts of earlier solution approaches.

During development, the totality of implementation progress for all requirements offers a good overview of the implementation progress of the total system or of a subsystem (maturity level tracking).

If you want to systematically exploit the advantages of a requirements-driven process, then the process described above must be applied to all subsystems, including those of different development disciplines that are actually independent. Naturally, this also applies to diagnostics.

Today, spreadsheet-oriented tools and databases are usually used to manage requirements. Here, requirements are either not described formally, or they are only described formally in part. These tools must be flexible enough to capture and track all requirements – even those that are very fuzzy.

Regarding the specification, various other tools have become established in the various disciplines, e.g. modeling and authoring tools, which usually generate a formal specification. In contrast to user requirements, precise definition of the content is the primary goal and not flexibility, and this fundamental difference results in different, specialized tools. Consequently, classic requirement management tools can only be used meaningfully up to a certain level of detail. This also applies to diagnostics.

Standardized exchange formats are specially designed for a specific discipline. ODX, for example, specifies data that is relevant to the diagnostic tester. Exchange formats usually use a formal data model that assures a consistent specification that is complete in its details. On the other hand, these formats are too restrictive for formulating fuzzy requirements. Classic requirement management tools are well-suited to describing text-based diagnostic requirements. The standardized data exchange format ODX, meanwhile, would be unsuitable for describing or exchanging these text-based requirements, because it is too formal and precise.

ECU Software

Today, AUTOSAR (AUTomotive Open System ARchitecture) is the reference architecture for ECU software in the automotive industry. AUTOSAR standardizes the description of individual component or vehicle functions and the description of the overall system.

The diagnostic software in AUTOSAR consists of the three basic software modules DCM, DEM and FIM.

The DCM (Diagnostic Communication Manager) implements diagnostic communications according to UDS and OBDII. The DEM (Diagnostic Event Manager) implements a fault memory and manages fault status and supplemental information on fault symptoms. In the case of active faults, the FIM (Function Inhibition Manager) prevents execution of certain functions and suppresses secondary errors.

DCM, DEM and FIM are configured by the ECU Configuration Description (ECUC). Their contents are best understood by illustrating how requirements relate to the configuration of software components.

Figure 1: Iterative development process

Fuzziness and flexibility, which are advantageous in capturing requirements, must be avoided in configuring the ECU software. The software must be described precisely and unambiguously for all operating conditions that occur.
Significant contents of the diagnostic data that are relevant to the software configuration include the diagnostic services that can be called by an external diagnostic tester with request/response and their parameters (service identifier, sub functions and data parameters). The length and data type are relevant for all data parameters; constant parameters also require a constant value. In UDS, access to certain data packets may be restricted to certain sessions or security levels. This information is also contained in the configuration data, so that the software can assure conformance to the prescribed rules.
The second important aspect of the software configuration data is that it links the diagnostic software to the application. The parameters passed by diagnostic services can be linked to variables or functions of the application software. Software generators can then generate the relevant calls.
Since AUTOSAR diagnostics is limited to the UDS and OBDII protocols, the layout of diagnostic services of these protocols is implicitly assumed and is not explicitly described in the ECUC data.
The AUTOSAR ECUC data is stored in a standardized XML format, which enables its processing in code generators.

Supplying Data to Diagnostic Testers

Diagnostic data used to parameterize generic testers must contain all information relevant to the vehicle or its ECUs from the perspective of diagnostic communication. A significant difference compared to the configuration data described above is the vehicle scope. Especially in the service area, a single diagnostic tester needs to cover a large number of different vehicles, models and variants over many model years. The resulting volume of data requires efficient mechanisms to avoid redundancy and to achieve compact storage of the necessary data.
The specification character required for configuration is not really necessary for parameterizing testers; on the contrary, it may even be advantageous for a parameterization to contain multiple equivalent alternatives, because the appropriate data can then be automatically selected at runtime. When a diagnostic tester is connected to a vehicle, it is often unclear which ECU variants and software levels are installed in the vehicle under test.
In terms of content, the diagnostic tester data differs from the configuration data in that conversion information is an essential component. The compactly coded bus messages and their parts are displayed as physical values with units at the tester.
Examples of established data formats for parameterizing diagnostic testers are the cdd format from Vector and the ISO-standardized ODX format.

Example of a Tool Chain

During diagnostic development, the following tasks are performed, which are supported by the tool chain shown in **Figure 2**.

Defining, Gathering and Coordinating Requirements

IBM DOORS is widely used among automotive OEMs as a tool for capturing and managing requirements.

Creating and Coordinating the Specification

Here, CANdelaStudio can be seamlessly integrated into the requirements-driven process chain as an authoring tool for specifying ECU diagnostics, because CANdelaStudio supports the capture and import/export of requirements. Diagnostic objects (diagnostic services, data objects, DTCs) are generated at the press of a button from the requirements, which are already formally described. These objects are each linked to an original requirement. In this way, the user can have the imported requirements automatically adapted and synchronized to match updated requirements, and if necessary the specification can be modified. Closely interlinking requirements and specification is very advantageous in the typically iterative process, because it avoids duplicated efforts in creating and re-comparing the specification data.
The finished diagnostic specification serves as the input to subsequent steps in the process chain. CANdelaStudio saves the native diagnostic specification in cdd format, and an ODX file can also be exported at the press of a button.

Figure 2: Tool chain of diagnostic development

Generating and Integrating ECU Software

DaVinci Configurator Pro is a tool for configuring and generating the AUTOSAR basic software and an ECU's RTE. The user imports a diagnostic specification (ODX or cdd) and generates an initial ECUC configuration from it. Afterwards, the user progressively supplements the configuration for the ECU and makes it more specific and detailed. If there is a new version of the diagnostic specification, it is easy to re-import it, and the contents are automatically merged with those of the previously created configuration. The diagnostic software for the ECU is generated based on the resulting configuration.

Testing ECU Diagnostic Software

CANoe.DiVa is used to test the diagnostic implementation in the ECU at both the supplier and the OEM. CANoe.DiVa generates an extensive set of ECU-specific test cases based on the ECU descriptions in ODX or cdd format, which are then automatically executed in CANoe. Test results are shown in detail, and the user can comment on any test cases, or group, sort and filter them according to various criteria.

Using ECU Diagnostics

CANoe, CANape or Indigo is used as the diagnostic tester, depending on the application area. Having the CANdelaStudio specification as a common source for tester parameterization and ECU configuration ensures that the tester and ECU software match one another.

Figure 3: Contentual similarities of the several description models

Summary

The AUTOSAR and ODX standards that have appeared in the diagnostics area in recent years complement one another well and continue to be effective in meeting objectives. Although they cover related contents, they have very different areas of focus and overlap just slightly **(Figure 3)**. The operation area of the one standard cannot be covered by the other. The AUTOSAR method is also compatible with ODX.
In practice, however, there is still the challenge of assuring consistency of the data described in the different standards over a distributed and usually iterative development process. This challenge can be overcome by a well-defined process, targeted data transfer and support by tools available on the market today.

Dr. Klaus Beiter
leads a development team for the Automotive Diagnostics product line at the company Vector Informatik GmbH in Stuttgart. He is a member of the ASAM/ISO ODX working group.

Christoph Rätz
is the Director of the Diagnostics product line at the company Vector Informatik GmbH in Stuttgart.

Diagnostics from a Distance
Interactive Diagnostics for Vehicles Worldwide

Vehicle diagnostics is an important tool for quickly and efficiently localizing and correcting faulty behavior of individual vehicle components. In certain rare cases, however, it may not be possible to find the cause of the error locally without the support of an expert. This expert can now access the vehicle directly and interactively – even without having to be on site locally – using remote diagnostics, and can then examine the vehicle and systematically determine the cause of a problem.

Sweden, -20 °C, snowfall. A test driver conducts a test drive on a frozen lake covered with snow. During a braking maneuver in a curve, he notices some unusual vehicle behavior. He suspects that the cause lies in the brake system. After additional trials, the experienced test driver quickly recognizes that the behavior only occurs under very special conditions.
Although the test driver is very familiar with the vehicle, a precise analysis by a system developer is necessary. Only the system developer has the necessary background knowledge needed to quickly and comprehensively find the cause of this behavior. However, it is very rare for this engineer to be available locally to read out key vehicle data and drive actuators for test purposes using vehicle diagnostics.
With remote diagnostics, the expert can now access the vehicle despite the long distance from it, without having to quickly travel to Sweden.

Use Cases
Easy remote access to a vehicle or its components by experts is not only helpful during test drives. OEMs and suppliers can also benefit from remote diagnostics to diagnose their systems at later production startup.
And even in the service shop situations sometimes occur, in which it is essential to get the advice of an expert. In some cases, this is the only way to accomplish a quick and cost-efficient repair when an unpredictable and complex problem exists.

Diagnostics with Different Focal Points
Effective vehicle diagnostics is a key factor for achieving a high level of customer satisfaction with regard to the duration, cost and success of repairs.
It is an indispensable tool, which accompanies the vehicle over its entire life cycle – from development to production and finally customer service. Very different requirements are set in the various life phases, which must be considered in the development of diagnostics. During vehicle development, a deeper look into the ECU and more extensive interventions are needed. In production, diagnostics is used for the "OK / Not OK" test. In customer service, guided troubleshooting helps to localize errors without requiring very special knowledge, and it can be used later to easily verify the success of a service repair.
As consequence the diagnostic testers – differ considerably based on these very different requirements – with regard to their user control concepts, level of detail and access capabilities. Accordingly, a service shop tester only makes a part of the diagnostics implemented in the ECU available, while other parts are reserved for development or production.
However, if an unexpected problem now occurs in the field, the experts may sometimes require access to precisely these development-specific information or functions.

Data Protection
However, the general distribution of all diagnostic data with the customer service tester is not a solution either, because this would also make undesirable and very wide-ranging system interventions possible, interventions that should actually be reserved for just a small group of experts. Therefore, the data and functions are handled confidentially and are only accessible to a small group of users. This also makes it more difficult for unauthorized third parties to gain access to information on how functions of individual systems are implemented or to manipulate them. Therefore, precisely those parts of diagnostics are selected for the customer service tester that are needed for use cases in the service shop – user operation is made as simple as possible, and unintentional operating errors are prevented.

Interactive Remote Diagnostics
Using interactive remote diagnostics, the difficulty of physical separation of the expert from the vehicle is circumvented. Experts can access the vehicle as though they were present locally, and they can contribute their expert knowledge in this process. The service shop employee can execute supportive actions – such as activating the brake pedal – while the expert reads out measurement values and precisely observes the behavior of relevant vehicle components. If vehicle conditions permit, it is even possible for the expert to have access to actuators from a distance. The expert can use further actions to confirm initial suspicions of a reason that could explain the observed behavior, or else exclude it as a cause, and can thereby effectively determine the cause of a problem.
For our test driver in Sweden mentioned above, this means that the expert neither has to quickly travel to Sweden nor is it necessary to send the development diagnostic data to the test driver for attempting to resolve the problem on his own – under the guidance of the expert. It is also unnecessary to reproduce the problem after returning from Sweden, if it is possible to study the matter right away locally. This is especially important if the specific environmental conditions have an effect on the observed behavior.
A primary advantage of remote diagnostics, however, is that the expert can react immediately to measurement results, conduct additional measurements, modify parameters and address actuators. This interactive access capability also illustrates the significant difference between remote diagnostics and the approach of using a logger or onboard tester.

Remote Diagnostics with High Levels of Performance and Data Protection
The benefits of interactive remote diagnostics succeed or fail with the ability to process diagnostic inquiries at a high rate of speed and with low latency. The new Version 4.0 of the Indigo diagnostic tester from Vector **(Figure 1)** supports the interactive remote diagnostics described above.

Figure 1: Diagnostic tester Indigo: Fault Memory and Measurement

Figure 2: Classic Diagnostic Tester

By comparison, a classic diagnostic tester is connected directly to the vehicle via a network interface **(Figure 2)**. In this case, all necessary diagnostic data and the required diagnostic and module knowledge must be available locally. When using remote diagnostics in Indigo, the classic diagnostic tester is replaced by an access point. Together with the communication server on the Internet, it serves as a routing hub and routes diagnostic requests and responses between the vehicle and the actual diagnostic tester **(Figure 3)**. The actual diagnostic tester is located remotely at the location of the expert. Neither the diagnostic data nor the expert needs to be sent on a trip – and yet it is possible to access the vehicle directly.

To use remote diagnostics, it is sufficient to download the access point on the vehicle side and invite the experts to a diagnostic session with an ID and password. It is especially noteworthy that no changes need to be made to the vehicle for the test system to be immediately ready for use.
With the implemented solution, the diagnostic data, test sequences and security algorithms remain within a protected environment – all control, interpretation and evaluation actions are performed on the expert's computer. A high level of data security is achieved in conjunction with end-to-end encoding.
In order to efficiently use full diagnostic capabilities efficiently, a number of technical measures are implemented to assure high bandwidth and low latency. This makes it possible to access vehicles with very short response times worldwide – even when transmitting large amounts of data.

Summary

By using interactive remote diagnostics, a system or diagnostics expert can connect to a vehicle anywhere in the world and examine errors locally and at the same time as they occur.
In this process, the expert does not need to rely on a locally available test system optimized for customer service, but can use an expert tool instead. Yet, the data required for diagnostics does not need to be distributed or transmitted – the data remains in the protected expert environment.
The interactive remote diagnostics offered by the Indigo diagnostic tester from Vector goes far beyond the static diagnostics of an onboard test system and also differs significantly from a remote desktop approach with regard to data protection and performance.
The diagnostic tool makes it possible to study unexpected behavior on test drives over great distances, and it also significantly shortens repair times when unexpected problems occur in the service shop. In service shops, in particular, efficient third level support with remote diagnostics can reduce repair time and costs and deliver a high level of customer satisfaction.

Rolf Weber
is team manager and product manager at the Diagnostics product line at the company Vector Informatik GmbH in Stuttgart. He is responsible for the test systems area within diagnostics, representing the diagnostic tester Indigo and the flash tool vFlash.

Christoph Rätz
is the Director of the product line Diagnostics at the company Vector Informatik GmbH in Stuttgart.

Figure 3: Concept of interactive remote diagnostics with Indigo

Data Recording for ADAS Development

Scalable recording of sensor and ECU data

The automotive industry is working extremely hard on technologies for autonomous driving. To replace human perception of the environment, the use of driver assistance systems with high-resolution radar and video sensors is essential. Massive quantities of data that need to both be transported via the communication networks as well as processed in real-time are thereby generated. This creates challenges in an unprecedented dimension on recorder solutions as well. When selecting a suitable system, a number of important aspects must therefore be taken into account.

Current radar and video sensors deliver a measurement data volume of approximately 100 MByte/s each. On top of this comes the sensor fusion data from within the ECU for an additional volume of approximately 50 MByte/seach.
If one assumes, for example, a vehicle that has been equipped with five radar sensors and two video systems, this, together with a number of other measured quantities, results in a data volume of approximately 1 GByte/s that needs to be managed during collection and storing. Thus, a test vehicle generates about 3.6 Terabytes of data per hour and 28.8 Terabytes of data per day (**Figure 1**).

Multiplied by the number of test vehicles and days of use, this results in massive quantities of data. Furthermore, experience shows that the requirements will grow further as the technology continues to develop.

Challenges during data recording in detail

In the first step, the ADAS sensors are to be connected to the recorder. For each sensor to transfer 100 MByte/s, each requires its own Ethernet connection with gigabit bandwidth to the recorder. The same applies for the connection of the ECUs to the recorders.

Planned as memory medium is the use of hard disks, whereby the following requirements are to be satisfied:
> Largest possible disk capacity
> Automotive-suitable temperature range
> Mechanical stability
> High data rate of at least one gigabyte per second
> Low price

Figure 1: Data from a wide range of sensors must be recorded and transmitted synchronously. By means of the sensor data fusion, one obtains a complete picture of the vehicle environment.

Because these partially contradictory requirements cannot be satisfied simultaneously, compromises are unavoidable. Depending on the project, solid state drives (SSDs) or classic hard disks are used. It is not necessary in all cases and for every trip to satisfy requirements such as the automotive-suitable temperature range.

To completely record the data of a test vehicle for autonomous driving for a work day with 1 GByte/s, the storage system requires a capacity of nearly 30 TByte. Such memory sizes and the required transfer rates can only be realized at the present time with RAID systems, i.e., by coupling multiple, large, independent hard disks in a disk network. RAID enables parallel disk access. As a result, the performance of the individual disks can virtually be added and – depending on the RAID level – also optionally offers protection against the failure of individual disks through redundancy.

To fully utilize the resources of the storage hardware, the
> recorder software,
> configuration of software and drivers,
> use of high-performance Ethernet PHYs,
> setting of the Ethernet message length
> as well as the choice of the appropriate RAID level
play an important role. The CPU of the recorder must, be able to reliably handle these large data streams. There must be enough RAM present to buffer an increased data volume in burst phases.

In practice, test drives may vary considerably from one to the next, for example, with respect to test scope and duration. While in one case it is possible to transfer the measurement data to another system after just one day, in another a long-term test might last an entire week. To be clarified here is the question of whether the data can be transferred by swapping disks in the logger or if there is a strategy for quickly reading out the data that does not block the test drive for an excessively long period.

If no provision is made for backing up the data during the day, the user has no alternative other than appropriately dimensioning the disk size.

The question of for whom the measurement data are intended also remains. If one wishes to send a sensor supplier the raw data from his specific sensors, the respective data must be extracted in a separate work step, provided all data are stored in a single measurement file.

The energy management system of the recorder is responsible for keeping the system switched on until powered down, provided no malfunctions occur due to undersupply or voltage drops. During pauses while reading out and transferring measurement data via WLAN, on the other hand, the recorder must not switch to a sleep mode. At the same time, there must always be enough energy in the vehicle battery to be able to reliably restart the car.

Even the best recorder hardware is dependent on optimally functioning recorder software. Its primary task is the receiving and writing of the measurement data on the disk system. In addition, the software provides the developer with services such as the starting and stopping of recordings and allows, trigger conditions to be defined.

The data arrive at the recorder in parallel via multiple Ethernet ports. Typically, these include XCP-on-Ethernet information from ECUs as well as raw data from radar sensors and video cameras that the engineers use for autonomous driving or the ADAS application that is to be tested. An important task of the recorder software prior to the efficient streaming of the data to the hard disk is the time synchronization of the various data sources. Using the CPU resources as optimally as possible, the data are to be prepared in such a way that no further processing is necessary after they are stored. To this end, a measurement data format is required that permits the constant recording of data quantities of any size.

In summary, the following can be postulated: The individual components of the hardware and software are to be selected with great care. A complete solution of hardware and software must be perfectly matched to one another.

Scalable basic concept provides maximum flexibility

With comprehensive scalability over all function units, such as sensor connections, hard disks, software and hardware, all conceivable requirements that may arise during ADAS data recording can best be satisfied.

Ethernet: Classic or Automotive

For the connection of the signal sources, classic Ethernet as well as Automotive Ethernet are equally suitable. The number of required connections may vary greatly from project to project. If the recorder does not have a sufficient number of ports for direct connections, a switch must be connected upstream. The switch then sends the data from the upstream sources together to the recorder via a 10 gigabit Ethernet cable. If necessary, the switch can also assume gateway functionality to physically convert Automotive Ethernet to classic Ethernet. The Ethernet base structure is thereby made scalable, for example, if the number of required ports increases during the course of a project.

Hard disks: Interchangeable RAID system

In addition to being provided with a high-performance RAID controller, there must be a sufficient number of hard disk slots available on the hardware side; these are to be flexibly equipped depending on the project specifications. Important here is that the entire RAID system with all disks either be easy to replace or that a high-performance interface for downloading the data be available.

Recorder software: One instance per data source

The most efficient operation can be achieved by allocating for each data source an independent measurement instance that receives the data and stores it on the disk. In this case, each instance generates a separate measurement file. As a result, the entire measurement is divided into multiple measurement files. The job of the software is to ensure the integrity and time synchronization of the measurement data. Project extensions that arise from an additional source thus only result in another measurement instance.

Recorder hardware: Clones desired

If all hardware resources are exhausted and hard disk slots as well as Ethernet ports are occupied, this problem can be solved through the use of an additional recorder. The time synchronization must not be lost, however. The question of the operating concept also arises: how is the user to start and stop measurements at exactly the same time on multiple recorders? Ideally, an additional recorder is added in such a way that, from the perspective of the user, it is fully transparent whether he is solving the measurement task with one or more hardware platforms.

Figure 2: Scalable decentralized recorder solution for very high data rates in the ADAS environment for XCP on Ethernet, video and radar raw data

The completely scalable solution

The recorder platform from Vector is based on an industrial PC design that is already equipped with a large number of 10-gigabit as well as 1-gigabit Ethernet ports. An expansion unit provides space for up to eight RAID hard disks, which can be completely and very quickly exchanged. There is also space for additional expansions, such as an uninterruptible power supply (UPS). The VX1000 measurement and calibration hardware is used to collect both the raw sensor data and the ECU fusion data and bring them into the PC via Ethernet.

CANape is the recorder software. The new "Distributed High Performance Recording" function included in CANape version 15.0 permits the parallel use of many measurement instances, thereby allowing the multi-core architecture of the computer to be fully utilized (**Figure 2**). The instances are, thus, not limited to one PC. Rather, the user can easily distribute them over other PCs in the network. The PCs used as part of the recorder solution need only be connected via Ethernet. CANape distributes the measurement tasks over the number of PCs and ensures the time synchronization. With triggered measurements, CANape also provides for precise time control of the processes over all measurement instances (**Figure 3**). The user needs just one CANape license for the entire setup.

Users benefit from using CANape as logger software in a number of ways. The system can easily be used not only by experts, but by laymen as well. Thus, engineers who work on the further development of their applications have at their disposal the full functionality of the software. To reach the required number of test kilometers, persons without development or tool knowledge can then perform the drives at all hours of the day or night. CANape is able to automatically record the data; no modifications to the vehicle are necessary. Because the system uses the standardized ASAM-MDF-4.1 format to store the data, the measurement data can directly be used by other tools in the evaluation chain.

Conclusion

By using recorder solutions with consistently scalable architecture, developers of driver assistance systems obtain very flexible systems that can be expanded to a nearly unlimited extent. The data rates of 500 to 700 MByte/s required in current autonomous driving projects for the storing of radar, video and ECU data can still be managed at the present time with just a single PC. As requirements increase, investment security is ensured by the VX1000 hardware – which can be used for both ECUs as well as for raw sensor data – and by the scalability of the complete solution.

Figure 3: Recording of sensor and ECU data, optimization of the algorithm and stimulation of real or virtual ECUs with CANape

The complete sensing of the surroundings with all relevant objects that is required for autonomous driving can be realized more quickly and more reliably with a comprehensive solution consisting of software and hardware tools as well as embedded components. During software development with the different types of sensor sources and their fusion, the use of a working prototype is helpful. A combination of the vADASdeveloper prototyping environment from Vector and the Create algorithm library from Baselabs has proven itself in practical use (**Figure 4**).

In addition to the validation and testing, Vector also offers project-specific support for the forthcoming requirements with computing-intensive algorithms on multi-core, high-performance platforms, e.g., during the definition phase of ECU architectures with Hypervisor-/POSIX-based concepts. Furthermore, the computers in the vehicle that calculate tactical algorithms for driving maneuvers demand more flexibility from the underlying operating system. Here, the AUTOSAR adaptive platform comes into play. It makes the ECUs fit for future requirements.

Links

Website Vector: www.vector.com
Product information CANape: www.vector.com/canape
Product information VX1000: www.vector.com/vx1000
General information Vector's ADAS solution: www.vector.com/adas
Product information BASELABS Create: www.baselabs.de/sensor-fusion-algorithm-development

Autor
Andreas Patzer (graduate engineer) studied electrical engineering at the Karlsruhe Institute of Technology. His focus was on measurement and control technology as well as on information and automation technology. In 2003 he joined Vector Informatik GmbH in Stuttgart, where he is responsible for "Customer Relations and Services" as team leader in the Measurement & Calibration product line.

Translation of a German publication in Elektronik automotive issue 2-3/2017

Image rights: Vector Informatik GmbH

Figure 4: Visualization in vADASdeveloper that can be adapted to every application case, both for scenarios with dynamic as well as static objects, such as highway assist, lane-change assistant with blind-spot monitoring or automatic parking.

VECTOR

All Puzzle Parts for OBD Documentation

Integrated process for generating the legally required OBD documentation faster and without errors

OBD-II and HD OBD require extensive documentation from manufacturers for approval of new vehicle models by certification authorities. Creating it requires precise knowledge of the legal requirements, details of the implementation of the monitoring functions in the ECU, knowledge of extensive responsibilities, insight into the calibration parameters of the ECU, and much more. Besides the complexity of these tasks, correctness (of form and content) is extremely important, since there is a threat of fines for each approved vehicle. A structured process for creating On-Board Diagnostics documentation supports manufacturers and suppliers in meeting these challenges and makes it possible to build up a knowledge base.

Environmental protection legislation sets emissions standards for vehicles with combustion engines. With growing environmental consciousness in recent years, these regulations have been extended to more and more vehicle classes, and the limits have been lowered again and again. These rules differ depending on the approval country, approval year, and vehicle type. The regulations not only govern the upper limits for pollution emissions, but also require monitoring of all emission-related components during operation. If an assembly fails and produces more emissions than stipulated, the ECU must detect this, store the information, and inform the driver. Furthermore, the legislation requires that standardized scan tools can query these monitoring functions and fault memories in order to allow checks by authorities, repair shops, and users.

The effort to implement and comply with the legal requirements represents a significant part of ECU software development work. In order to verify implementation of the legal requirements, the vehicle manufacturers must submit extensive documentation for approval of each vehicle variant. The so-called OBD documentation describes a list of emissions-related faults and their effect, monitoring, and detection **(figure 1)**.

All information needed for this documentation is produced automatically in the course of the development process, but often in an unsuitable form. For example, the fault detection strategy is implemented in the software and must be described in plain text. Limit values for sensors that trigger an error when exceeded are calibrated as parameter values in the HEX file of the ECU and must be identified as

Emission Testgroup: CARB 2.B

Label for fault detection criteria	Threshold value	Label for threshold	Secondary parameters	Label for secondary parameters	Enable condition	Label for enable conditions	MIL illumination [no. of trips]
TWCD_rOSCAlr gPrfr01 numMeas	(PV ASDrf_CoeffBGrpPosAbv_0MAP.Value.ExpandLink_MAP1) (PV ASDrf_CoeffBGrpPosAbv_0MAP.Value.min) to (PV ASDrf_CoeffBGrpPosAbv_0MAP.Value.max)	TWCD_rThresBl aComp_CARB3 TWCD_numMeas_C03	(DDOC.DFC_AC0aDetMax_CA.Secondary parameters)				2
				ienvcmc_w	>= 0.96	E0SDOes_rstLamEnaMinPart_C	
				ienvcmc_w	<= 1.04	E0SDOes_rstLamEnaMaxPart_C	
				B_tai	= false		
					>= 10 sec	E0SDOes_tPrAOMDurPart_C	
				Epm_nEngLR	>= 900 rpm	E0SDOes_nEngMinPart_C	

Figure 1: OBD documentation example: CARB Summary Sheets

a monitoring range. Or diagnostic trouble codes are listed in the diagnostic description and must be assigned to a standardized SAE code.

Above all, creating the OBD documentation is an organizational challenge. It is necessary to ensure that all information is collected in the appropriate form and correct version and can be processed by technical experts according to the legal requirements.

Typically, this task is performed largely by hand without special tool support. The person responsible for the OBD documentation collects this information in text or tabular documents and processes it. This involves significant effort, since, besides the initial data gathering, it is necessary to ensure that everything is up to date for the approval:

> Has the detection algorithm been modified during development?
> Does the calibration data correspond to the current vehicle variant?
> What information can be reused for derivatives?
> Who is the contact person for inquiries about a diagnostic function?

It is very important that all information is correct at the time of submission. The authorities set fines based on the number of approved vehicles. The regulations of the CARB (California Air Resource Board) set penalties between 25 and 50 US dollars per fault and approved vehicle. Depending on the number of units sold, this can quickly add up to significant amounts.

Integrated Complete Solution

A dedicated data management solution can offer significant added value and support error-free creation and

Figure 2: Clearly-organized, domain-specific view in the Function Inhibition Editor with clear assignment of diagnostic fault code and ECU function

maintenance of OBD documentation. Such an integrated approach is described below with the example of vCDM, the calibration data management solution from Vector. This database-supported platform allows ECU parameters to be managed reliably. All functions are available to calibration engineers as a multi-user solution in order to manage the data supply of various vehicle variants during development and series maintenance. Project approvals control access to the data. Furthermore, all calibration data is subject to version control, which allows all changes to be tracked consistently.
Defined functions and data are available to each developer according to his or her role and the project approvals. This allows an OBD calibration engineer to run the diagnostic application for an ECU with the "Function Inhibition Editor". The diagnostic fault codes and ECU functions are represented in a clearly organized matrix, and the blocking conditions can be set directly **(figure 2)**. Additional information such as the SAE code is displayed directly.

A Comprehensive Knowledge Repository

For the purpose of data management, it makes sense for developers to store additional text for every entry (software, parameter, diagnostic fault code). Arbitrary text categories can be defined and assigned, for example the "Application Tip" knowledge block for calibration parameters, or the "Enable Condition" and "Threshold Value" knowledge blocks for a diagnostic fault code. The current forms of these knowledge blocks are displayed there or modified. The knowledge management system exports and imports these from/to standard Office formats via standard functions for processing or reuse of the information. This makes it easy to load information that already exists into the knowledge repository and reuse it. Each knowledge block has a scope which controls which software versions or variants these texts can be reused for.
A report generator can be helpful for defining and printing templates for individual reports. In the generator layout, users can access all elements (parameters, diagnostic fault codes, knowledge blocks) and position them in the layout accordingly. It should also be possible to use formulas and variables within a knowledge block which can be "calculated" accordingly when it is output:
Threshold Value knowledge block for fault code P0401:
Content: "=$ECU_Parameter1 to $ECU_Parameter2"
Output: "=12.0 to 20.0"

This ensures that the values of the vehicle variant are always used when creating a report. In this way vCDM allows the company to build up comprehensive knowledge management in addition to the actual calibration data **(figure 3)**.

Efficient OBD Documentation

The OBD documentation in vCDM takes advantage of the versatile data management functions and provides a configuration that makes it possible to start documenting content immediately. The necessary vehicle variants can be created, and calibration data can be imported or applied. For the OBD domains, the relevant knowledge types are pre-configured, and lists of diagnostic fault codes can be imported. For certain engine ECUs, the fault codes can be read out automatically from the ECU description file (A2L). The report generator provides some templates with the required format for the European and US OBD documentation requirements. The base system offers extensive options for independent and customized extension. For example, users can define and populate their own knowledge blocks. Adaptable templates allow them to create their own reports or insert additional columns representing this knowledge so they can be used to represent project management requirements, for example. This provides a software tool to which all the technical experts involved contribute in order to provide and reuse information efficiently **(figure 4)**.

Conclusion

It is to be expected that the regulated requirements that On-Board Diagnostics places on ECU development will continue to increase constantly in coming years. These tasks are currently performed "by hand" in the development process with significant effort. An integrated tool can support all the engineers involved according to their tasks and makes it possible to reuse existing knowledge in a sensible way. This leads to significant time savings and helps to reduce errors and the associated costs.

Links
Website Vector: www.vector.com
Product information vCDM: www.vector.com/vcdm
General information about Calibration Data Management: www.vector.com/cdm

Autor
Michael Vogel (Graduate Engineer) is Business Development Manager for vCDM at Vector Informatik GmbH.

Translation of a German publication in magazine Automobil-Elektronik issue 4/2017

Image rights: Vector Informatik GmbH

Figure 3: Linking knowledge content to calibration data allows reuse and ensures that documents always show current content

Figure 4: All developers involved in the OBD documentation take advantage of knowledge that already exists in a central tool

Trends and Effects on Development Methods and Tools

Calibrating ECUs

The developments and challenges of the next five to ten years in the area of measurement and calibration of ECUs will be determined by global trends and will involve several changes. Established work methods are frequently reaching their limits, and adoption of new approaches by companies is inevitable. Data-based calibration methods, "intelligent" data management with virtually transparent data exchange, and flexible integration of expert knowledge via apps, among other things, will supplement the work of the application engineer.

In the view of numerous managing directors in the automotive industry, the key trends influencing this sector in upcoming years will include handling of new markets, standardization of vehicle platforms and modules, driver assistance systems, and further optimization of internal combustion engines [1]. This will create new demands on companies and their development departments, requiring fundamental changes to many work methods.

For example, while new markets require much closer collaboration with customers, business partners, and service providers at the international level, more standardization might lead to a greater number of variants. Further development of combustion engines, in turn, will further increase the already high level of complexity of the power train. Today's modern diesel drives with exhaust gas cleaning already require calibration of over 60,000 parameters, with 4, 8, 10, or 20 variants.

The complexity and configurability of ECUs is also continuing to increase in other electronics areas such as the chassis or body and for assistance systems. Even if only 5000 parameters have to be managed for chassis systems, engineers must ensure this for 80 to 100 product variants. In the case of ultrasonic sensors, a typical component of assistance systems, there are only a few calibration values but the sensors are installed in 1000 vehicle variants.

Efficient development, testing, and validation of driver assistance systems is possible only through direct integration with the existing measurement and calibration tool. In addition to high data rates, the sensor data fusion calls for expansion of existing signal-oriented concepts in the direction of object-oriented representation and processing of information.

From the Virtual Calibration to the Automatically Generated Parameter Set

Other trends include calibration in virtual environments, automatic generation of knowledge from existing datasets, and migration of web technologies into industrial work sequences. While calibration has shifted in recent years away from the vehicle to test benches and HIL systems, in the future more calibration tasks and optimizations will be carried out with simulations and models in the lab **(Figure 1)**. This spares companies the need for expensive prototypes and test vehicles. The approach will no longer be simply to measure signals and collect and manage data. Rather, automatic generation of results will be the focus. The existing database will serve as the basis for creating new parameter sets and variants using corresponding calculation algorithms.

Reliable, Value-added Integration of Cloud Technologies

Social Media, Internet of Things, and Web 3.0 are showing us how data exchange, connectivity, and expandability can be realized today in a simple, all-encompassing manner using apps. It is a matter of providing the potential of these approaches as integrated, reliable company solutions in order to tap into it for measurement and calibration applications. For example, applications for measurement data analysis offer excellent potential for added value through use of these technologies. While previous solutions required use of expensive special software and its time-consuming configuration, a cloud solution is a straightforward means of providing simple sharing of measurement data as well as central algorithms for data mining.

Future-proof Tool Chain for ECU Calibration

All of these trends will have a lasting impact on the tool chains for measurement, calibration, and diagnostics in the coming years. As a tool manufacturer supplying high-performance products for ECU calibration to customers both now and in the future, Vector is focusing intensively on these trends. The company has identified four strategic topics in the area of ECU calibration that it is concentrating its efforts on: assistance systems, added value in power train, flexible interfaces that allow tool departments of companies or external service providers to easily integrate Vector products in their solutions, and integrated, reliable provision of Internet technology in measurement and calibration solutions.

Current Solutions Ideally Suited for Further Development

Calibration tasks in power train are multi-faceted and range from measurement data acquisition to offline calibration. The latter focuses on working with measurement files and parameter values. The application engineer creates and manages variants or uses model-based or virtual technologies for optimization and dataset generation.

Figure 1: ECU calibration is increasingly shifting from the vehicle to test benches and HIL systems. To reduce costs for expensive prototypes and test vehicles, more calibration tasks and optimizations are being carried out with simulations and models in the lab.

Figure 2: As part of model-based software development, the functions of applications are tested in an iterative process. A convenient measurement, parameter assignment, and visualization interface for models such as CANape significantly accelerates the development process.

High-performance solutions for various offline task areas are already available today and form an ideal starting basis for further development. The palette includes a range of tools. vSignalyzer is a measurement data analysis tool. CDM Studio is a stand-alone calibration data handling tool. vCDM is an enterprise solution for calibration data management. CANape, which in combination with Simulink, is a powerful platform for model-based calibration **(Figure 2)**.
The bandwidth requirements and response times during measured data acquisition and online calibration are rapidly increasing. Chassis and assistance systems now require transmission rates of up to 100 Mbps. With the VX1000 measurement and calibration hardware, a fast, direct, and cost-effective interface to the ECU is available that communicates with the ECU using the XCP on Ethernet standard ASAM protocol and can be easily integrated in the housing in a space-saving manner using a POD (Plug-on-Device).
For data exchange inside the team, solutions are in preparation that will allow sharing of work results, measured values, parameter sets, reports, etc., via a trusted cloud virtually at the press of a button. The company network as well as a cloud hosted by the tool manufacturer may be used for this. Whether in the office or on the test track, data can be read, changed, written back, and synchronized easily.

Figure 3: Equipped with a uniform user interface, the "Calibrators Workbench" will support engineers with a modular system of tools. It will provide similar methods and allow editing, analysis, and management of ECU-internal data. Users can easily develop and integrate their own expansions through reuse of standard functions.

Vision 2020: Calibrators Workbench

Scalability paired with investment protection will also be at the top of the priority list for customers for a number of years. Needed are tools and implementations that grow with requirements – from a single workstation to teams and finally to a enterprise-level solution. The specialists of Vector have named their vision of the next generation work environment the "Calibrators Workbench" **(Figure 3)**.

What is meant here is an integrated offering of tools that are all equipped with a uniform user interface and the same look and feel. The tools provide similar methods and allow editing, management, and analysis of calibration data using a variety of tools. Essential features of the Calibrators Workbench will include expandability using apps or vApps (virtual apps). This will enable engineers to reuse existing functionality (reading/writing of measurement files, parameter set files, map editors, etc.) for their own applications, which reduces development expenses. It will thus be possible to buy additional expert knowledge when needed and easily integrate it into the Calibrators Workbench. The apps equipped with server or cloud infrastructures can originate from Vector or external service providers. As a central node for knowledge management, Calibrators Workbench will ultimately provide a simple cloud-based sharing of work results.

Translation of a German publication in Hanser Automotive 11-12/2015

List of references:

[1] KPMG's Global Automotive Executive Survey 2014 – Strategies for a fast-evolving market

Image rights:
Lead figure: © Fotolia.com/vege, edited by Vector
Figure 1 – 3: Vector Informatik GmbH

Links:
Website Vector: www.vector.com
Product Information CANape: www.vector.com/canape
Product Information VX1000: www.vector.com/vx1000
Product Information CDM Studio:
www.vector.com/cdm_studio
Product Information vCDM: www.vector.com/vcdm
General Information Calibration Data Management:
www.vector.com/cdm

Michael Vogel
is employed at Vector Informatik GmbH as Business Development Manager responsible for vCDM.

Analyze Large Quantities of Measurement Data Rationally and Flexibly

On test benches and in durability tests, automotive OEMs collect important information on the behavior of vehicle components under realistic conditions. However, in view of the enormous quantities of data that are generated and their complex interrelationships, it is often a time-consuming process to subsequently identify and analyze the relevant data sets. To accelerate the analysis of measurement data in testing its automatic transmissions, Daimler AG relies on automated data evaluation by the CANape measurement and calibration tool from Vector.

The development of automatic transmissions at Daimler began in the year 1960 with a 4-speed transmission, which would be considered a rather simple engineering design by today's standards. The rapid advancement of technological development is attributable to a wide variety of new requirements such as increased comfort needs, larger gear spreads, lower fuel consumption, more powerful engines, additional gears, etc. For example, the drive-off element was changed, planetary gears and torque converters were added, and in 1995 the first version with electro-hydraulic transmission control was launched.

The 7G-Tronic Plus automatic transmission represents a pinnacle of this development history. Designed in 2010, the 7G-Tronic Plus can handle torques of up to 1000 Nm and can be implemented in a broad range of vehicles: from the smallest rear-wheel drive vehicle of the C-class with a 4-cylinder engine to the high-performance models of AMG. The transmission is also used in the small variant of the Sprinter van. It attained a successful combination of the seemingly contradictory requirements of optimized fuel economy, driving fun and ride comfort, and in 2011 the transmission won the internal Daimler Environmental Leadership Award.

Automatic Transmission Requires Many Parameter Optimizations

The extremely broad implementation range in the different models requires optimal calibration of ECU parameters to achieve the desired driving behavior. The path to product maturity was accompanied by numerous test bench and in-vehicle durability runs. Measurement data accumulating from daily testing is saved on servers, where it is available to development and calibration engineers. The challenge in evaluating and analyzing these large quantities of measurement data is to identify those data sets in which errors occurred, such as limit value violations or excessive thermal stresses. The errors of a poor gear shift operation, for example, expresses itself in typical vibration and jerky behavior, which is noticeable to the driver and passengers. Excessive thermal stresses associated with increased wear occurs when allowable friction values are exceeded during clutch operations.

Proven Analysis Process is Pushed to its Limits

The measurement data, which exists in various formats, comes from the CANape measurement and calibration tool and from other data loggers. Previously, it was evaluated in a method where the first step was to process it in an internal Daimler tool, and it was written to an Excel file **(Figure 1)**. Then, an Excel macro would generate a graphic overview on which the user could discern "trigger points" where errors occurred. Using these results, the relevant measurement files were loaded in CANape, and finally the analysis points were displayed there. This approach was not only tedious; it was also burdened by other disadvantages and limitations. The Excel tools work very slowly, which is all the more challenging with large volumes of data. Moreover, they also limit the maximum amount of data that can be processed, because the total number of lines in Excel tables is limited, and the program only offers limited graphic display options. Maintenance efforts for further development of the solution also had to be provided by Daimler.

Automated Evaluation by Data Mining

Since CANape is already widely used at Daimler – whether in calibrating ECUs, logging measurement data in test bench tests and durability tests, or for evaluation purposes – the managers there decided on an implementation strategy based on this Vector tool. The graphic display functions are optimally tailored for measurement data applications and – an important prerequisite – CANape offers the option of formulating company or project-specific evaluation algorithms using its internal programming language.

Calibration Tool as a Platform for Analysis and Graphics

To obtain a usable analysis tool as rapidly as possible, Daimler engaged Vector to implement the concept. Vector's task was to represent the evaluation algorithms Daimler wanted as CANape scripts and to graphically prepare the results. The data mining functionality in CANape is now used to analyze the measurement data of interest. In an analysis of results, the tool lets the user visualize the measurement file precisely at the location at which an error occurred **(Figure 2)**. CANape can be used as both the platform for executing the analyses and the platform for displaying the results. The size limitation has been overcome, and data volumes of up to 100 GByte can be processed effortlessly.

On the configuration side, the user selects the measurement data, chooses exactly what to study from a list of possible evalu-ations, e.g. upshifting or downshifting, and starts the analysis process **(Figure 3)**. After completing the evaluation, statistics are generated which provide a summary of the analysis. They show, among other things, that a 1-2 shift occurred a total of around 1,200 times **(Figure 4)**. The heat entries can be displayed above other physical parameters in XY diagrams, for example. This results in dot

Figure 1: The measurement data generated on test benches and in durability test vehicles are saved on the server and are available to calibration engineers for evaluation.

Figure 2: Data mining functionality is used to identify poor and good gear shift operations in a 2-axis diagram. This lets you quickly detect any limit value violations.

clouds, where each point represents a shift operation. The user can then recognize points lying outside of value limits based on their position on the diagram. When a point is selected, the related measurement file is loaded, and the user can generally visualize the value over a time axis in a display diagram. The time segment in which this shift operation occurred is shown directly.

Since the window contents are time-synchronized in CANape, all other windows show the specific contents, e.g. torques or engine speeds, which exactly match the time point at which the selected data point was measured.

In the windows, it is not only possible to show signal responses, but also limit value lines, e.g. the maximum tolerance values for frictional work and frictional power. Points lying outside of the limits indicate limit violations and errors in the shift operation. This makes it easy to identify those gear shift operations that require closer examination.

Flexible Adaptation of the Measurement Data Evaluation

The data mining functions of CANape allow Daimler test and calibration engineers to essentially perform an entire analysis of all of the measurement data and to see whether limit values were violated, or if undesirable events occurred. This is an important step for developers to attain more efficient use of the existing data material, and in the end it lets them arrive at conclusions quicker and more precisely in determining whether a specific ECU software level fulfills the required maturity level.

The requirements for the analysis are subject to a continually changing dynamic. Scripts may be modified either by the end customer or by Vector as a service. If the language tools of CANape are inadequate for any reason, other function libraries may be gene-rated from C code or Simulink models, and then they can be used as DLLs in CANape. This makes it possible to implement any desired evaluations.

Figure 3: Data Mining user interface of CANape that was customized to the individual requirements of Daimler engineers.

Figure 4: Evaluation of traction upshift: initial assessments of the durability run can already be made based on information in the predefined display windows.

Translation of a German publication in Elektronik automotive, 10/2013

Image rights:
Daimler AG

Links:
Daimler AG: www.daimler.com
Vector Informatik GmbH: www.vector.com
Product information CANape:
www.vector.com/vi_canape_en.html

Erhan Tepe
graduated with a major in Information and Communications Technology at the Polytechnic College of Reutlingen. After a two-year position as a programmer at a supplier, he obtained a Master's degree at the European School of Business. In 2007, Mr. Tepe went to work for Daimler AG, where he is employed as a test engineer in automatic transmission testing. His work area involves test stand and vehicle testing in the validation of automatic transmissions.

Andreas Patzer
graduated with a major in Electrical Engineering at the Technical University of Karlsruhe. Focal points of his studies were measurement and control engineering as well as information and industrial engineering. In 2003, he moved to Vector Informatik GmbH in Stuttgart, where he is team leader for Customer Relations and Services in the Measurement & Calibration product line.

Verification of Driver Assistance Systems in the Vehicle and in the Laboratory

Driver assistance systems acquire the vehicle's environment via a wide variety of sensors. Warnings to the driver or (semi-)autonomous interventions in the driving situation are always made based on correct results of the object recognition algorithms. This article addresses the typical challenges that arise in verifying object data and testing the image processing algorithm. The XCP standard enables the necessary high data throughput in measurement and calibration.

Behind the wheel, humans acquire information about their environment via their sensory organs – specifically their eyes and ears. Signal processing in the brain interprets the collected information, decisions are made, and actions are initiated. Decisions might include whether a space on the side of the road is large enough for parking or whether the distance to the car ahead needs to be adjusted. Driver assistance systems (Advanced Driver Assistance Systems or "ADAS") support the driver in making these decisions, thereby enhancing safety and improving comfort and convenience as well as economy.

Access to Sensor and Algorithm Data

Driver assistance systems must be able to reliably detect the environment as a type of "attentive passenger". Radar, ultrasonic and video sensors are very often used to provide information to ECUs on the driving situation or the vehicle's environment. Complex algorithms process the sensor data to detect objects such as road signs, parking vehicles, other participants in traffic, etc., and they initiate actions. To verify the sensor system, it may be sufficient to simply measure the results of the algorithm and compare them to reality. An example here is the distance measuring radar of an Adaptive Cruise Control system: The sensor detects objects by return reflections of the radar beam. The ECU supplies range distance information for each object as coordinates. In this case, it is not necessary to acquire all of the radar reflections in the sensor. However, all input variables of the algorithm must be measured if the data is being logged for later stimulation in the laboratory, for example. In this case, over 100,000 signals with a data rate of several megabytes per second would not be atypical.

Image processing ECUs with video sensors are used for road sign detection systems or lane-keeping assistants. An algorithm analyzes the video images and detects road signs or lane markings. One typical requirement for data processing in the ECU is a high level of microcontroller performance. On the other hand, whether the sensor data originates from a video or radar system has little impact on measurement instrumentation requirements – a high-performance solution is essential for transporting the measurement data. In evaluating and optimizing the algorithms, the measurement instrumentation must be able to acquire all of the algorithm's input and output variables and all necessary intermediate variables within the algorithm without incurring additional controller load **(Figure 1)**.

Serial bus systems such as CAN and FlexRay run into their performance limits in terms of the necessary data throughput rates. Therefore, controller-specific interfaces such as Nexus, DAP or Aurora are used to transport the large quantities of measurement data. It makes sense to rely on established and proven standards to avoid having to develop a separate solution for each technical measurement task. The VX1000 measurement and calibration hardware from Vector is ideal for this; it transfers the data from the controller interface to a base module via a small PC-board (plug-on device or POD), where it converts it to the standardized XCP on Ethernet; it then transfers the data stream to the PC at a high throughput rate [1].

Validating Sensor Data with Reality

The ECU's object detection results must now be verified against reality. Is the distance to the vehicle ahead on the road actually 45.5 meters? To compare the sensor data with reality, it is first necessary to acquire that reality. A camera, which is independent of the sensor system, records the driving situation. Developers can now quickly and reliably verify the object detection algorithms of their ECUs by comparing the objects detected by the ECU with the video image. The CANape Option Driver Assistance measurement and calibration software from Vector is used to overlay the object data on the video image. This lets developers determine exactly where something was detected and whether what was detected makes sense. In **Figure 2**, an "X" can be seen in the image at each point representing data obtained by the sensor. Coordinates detected by the sensor, such as the distance ahead and angle to the side, are converted to pixel coordinates of the video image in the PC.

Approving and Optimizing Algorithms

If deviations occur while comparing detected objects with reality, then the algorithm needs to be optimized. This is done by modifying the calibration parameters of the system, and it requires that the calibration parameters be defined in the code such that they are located in RAM at runtime and can be changed by a write access. The mechanisms of the XCP measurement and calibration protocol [2] are available for calibrating these parameters. At runtime, the developer modifies the parameter values and gets immediate feedback on the effects. XCP is not limited to use in the ECU. For example, the algorithm could also be run as a virtual ECU in the form of a DLL on the PC. Calibrations and measurements are also made over XCP – which makes the PC a rapid prototyping platform.

What is the most convenient way to incorporate an XCP driver in a DLL? How are the input data linked to the DLL? In the case of a Simulink based development, the "Simulink Coder" from MathWorks is used to generate the code for different target platforms from the model. The CANape tool from Vector might be specified as such a target platform. In the process of generating the code for CANape, an XCP driver is automatically integrated. At the end of this process, there is a DLL with an XCP driver and an ECU description file in A2L format. Both are integrated in CANape, and the input and output ports of the DLL are linked to real data. At the measurement start, CANape transmits the measured sensor data as an input vector to the algorithm, and the virtual ECU computes the results. The calibration

Figure 1: Acquisition of inputs and outputs, the environment and all data relevant to evaluating the algorithm. Display of all data, coordination of parameters.

Figure 2: Video image of the environment with objects detected by the distance measuring radar system and display of objects from a bird's eye perspective.

parameters are optimized in the same way as in a real ECU. A C++ project supplied with CANape leads to the same result as manually written code.

Stimulation with Sensor Data

Developers of sensor systems are confronted with two problems:

> Meaningful, realistic data from a sensor is often only available in the vehicle; the necessary environment is lacking in the laboratory.
> Achieving reproducibility of sensor data requires tremendous effort.

For these reasons, stimulation of ECUs with previously logged sensor data is a key component in development – whether it involves a real or virtual ECU. The data may be written directly to the ECU's memory, circumventing the inputs, and the VX1000 System provides the necessary bandwidth. Or the data may be transported into the ECU via its sensor inputs **(Figure 3)**.

In a virtual ECU, stimulation involves streaming a logged video or signals from measurement files to an input port in CANape. In real ECUs, the physical inte rfaces must be considered. In video systems, for example, the video sensor signals can be routed to a monitor on which a logged traffic situation is running. The ECU is always stimulated in the same way by using the same videos or signals – and this assures reproducibility of the data. Any changes in the behavior of the algorithm are then exclusively a result of calibration and not of changed input vectors. In both virtual and real ECUs, stimulation is not limited to feeding data to inputs; necessary states and preconditions can also be set over XCP.

Summary

Optimal calibration of ECUs requires a great deal of effort. Measurement and calibration tools communicate with the ECUs, and this makes code instrumentation unnecessary. Processes are defined for generating A2L description files and much more. However, all of these activities are kept independent of the ECU's tasks. XCP is a standardized solution here; it is well-suited for all types of ECUs. Although driver assistance systems may involve special requirements in terms of data volume and performance, the use of existing tools based on XCP – such as CANape and devices of the VX1000 product line-up – is a convenient solution for ADAS ECUs too. They represent a natural step in the advanced development of existing solutions that can be seamlessly integrated into existing development processes – from the support of video data to the use of a rapid prototyping platform to develop image processing algorithms.

Figure 3: Different input sources for stimulating a virtual video-based ECU in CANape. Real online data from the ECU via the VX1000 solution – from a camera or a logged video sequence.

Translation of a German publication in ATZ elektronik, 10/2014

Image rights:
Vector Informatik GmbH

Links:
Vector Informatik GmbH: www.vector.com
Product information CANape:
www.vector.com/vi_canape_en.html
Product information CANape Option Driver Assistance:
www.vector.com/vi_canape_en.html
Product information VX1000:
www.vector.com/vi_vx1000_en.html

Andreas Patzer
graduated with a major in Electrical Engineering at the Technical University of Karlsruhe. Focal points of his studies were measurement and control engineering as well as information and industrial engineering. In 2003, he moved to Vector Informatik GmbH in Stuttgart, where he is team leader for Customer Relations and Services in the Measurement & Calibration product line.

Calibration Data Management – A Puzzle Game no more

Manage Complex Calibration Data at the Individual Workstation, in Teams and Throughout the Company

In ECU development, short innovation cycles and high cost pressure lead to a distribution of work, in which the software development process is separate from the process of adapting it to its desired behavior in the vehicle. In today's vehicles, tens of thousands of such calibration data values must be determined and managed – and indeed for each individual vehicle variant. To avoid errors, quality standards are needed that have the same level as in the development of the actual software. This article discusses the requirements of tools for managing parameter values and presents a universal solution.

In the functional development of ECU software, high quality standards (SPICE, CMMI) must be maintained. The correct parameterization of the ECUs is as crucial as the correct functionality of the software in the ECU. Consequently, modifying parameters must meet the same quality levels. This modification is referred to as "calibration" which is essential for process adherence and quality assurance to trace calibrations precisely and to ensure consistent parameterization of every conceivable variant.
The trend toward shorter innovation cycles and more stringent requirements for quality and efficiency make it essential to achieve a higher degree of reusability. It must be possible to reuse the software in many models and vehicle variants in developing electronic systems. Each variant in an ECU leads to a separate parameter set, and this increases the number of parameter sets substantially.

Today, around 60,000 parameters must be calibrated in the ECUs of a diesel engine that conforms to the Euro-6 emissions standard. Although ECUs in the chassis and car body areas involve fewer parameters, they typically exhibit higher numbers of variants, and this also requires a dedicated data management solution. In summary, the manufacturer of an automobile today must generate and manage thousands of parameter set files.

Improving Quality with a Clearly Structured Process

The use of a special data management solution offers significant help in the complex process of managing calibration data. It guarantees that the thousands of parameters are used in the correct variants in precisely defined combinations for hundreds of vehicle calibrations. At the same time, each change must be precisely traceable for quality assurance purposes **(Figure 1)**.

Such a clearly defined approach creates process reliability and increases data quality. The original files of the engineers who maintain their work results in a Calibration Data Management (CDM) system are subject to version management. This means that the user can tell at a glance which parameter sets have flowed into a specific variant and where they are being reused. This assures automatic updating of a new parameterization version in all relevant variants.
Within the project, a quality and maturity level model helps in documenting changes for data integration and monitoring the progress of the project. The CDM system takes a multi-stage approach to data integration. After all results from the calibration engineers have been delivered into the system, it is able to resolve conflicts due to faulty or duplicate parameterization by means of assigned parameter responsibilities and rules. The results can then be validated by the four-eye principle and checked for completeness. Only then is the data ready for consolidation for the next data revision level and release for the next phase. This also facilitates cooperation with development partners, and it assures access to correct data in different competency areas.

Efficiency by Intelligent Variants Management

Along with supporting calibration tasks by providing a clearly structured process, another core task of data management is to productively manage large numbers of calibration variants. A "calibration variant" represents a set of vehicle attributes such as engine displacement, transmission type and emissions standard. The different product configurations require changes to the parameterization of the ECU software, and they are referred to as "derivative calibrations".
Variant management provides mechanisms which ensure, for example, that all variants automatically get consistent parameterization – thereby avoiding unnecessary redundant work. For parameters that depend on system attributes, rules are used to describe dependencies. This guarantees consistent parameterization of all variants with identical system attributes without additional work.

Think Big, Start Small

A company's adoption of a CDM solution rarely done in one big step, but frequently, solutions emerge for specific challenges that come up in performing specific tasks. For instance, a calibration engineer might need assistance in distributing work results to all calibration derivatives. Or a project leader needs to resolve or avoid conflicts which occur when calibration data is entered into the system by team members. The proliferation of such insular solutions makes a unified data management solution for the entire company increasingly all the more important. A central CDM solution can represent the entire process and supports adjacent areas such as software development and validation **(Figure 2)**.

Figure 1: The CDM System offers a clearly defined process that leads to substantial quality improvements in the calibration of ECUs.

That is why it is helpful for the data management solution to satisfy the specific requirements of a calibration engineer as well as those of a team and the entire company. Such a scalable solution can be introduced and built up stepwise, while protecting investments.

Data Management for the Calibration Engineer

In the ECU calibration process, the engineer first evaluates the initial behavior of a system component using software tools (MCD tools) on the test bench or in the vehicle. The engineer then adjusts the parameters towards attaining the target behavior. However, this "online calibration" approach only represents part of the required work.

Frequently, it is also necessary to compare and edit the parameters for different vehicle variants without a direct connection to the ECU (offline calibration). This is the case when parameters are "pre-calibrated" with values based on the requirements of software development, when the results of similar prior projects are adopted, when values are copied from online calibration to different variants or when numerical methods are used for model-based optimization.

For these tasks, the calibration engineer needs a tool with a user interface similar to Microsoft Excel, in which parameters (rows) can be shown together with calibration variants (columns), and any of the parameter values can be edited. In addition, the tool must support the special requirements of calibration (ASAM standard formats such as CDF2.0, characteristic maps, diagnostic information).

Cooperation on a Team

ECU calibration is largely teamwork. These teams frequently consist of members from different companies (automotive OEMs, ECU suppliers and engineering service providers). With increasing globalization and specialization, efficient co-working of calibration teams is continually gaining in importance. It must be possible to distribute work results quickly. Networking via e-mail and Internet is helpful - however - a lot of manual work steps are often required to exchange data between different systems. This is a process that is time-consuming and prone to errors. The data management system now gives teams the ability to automatically distribute the information. Integration of the e-mail system informs all participants about data changes, new releases and software changes. This guarantees that the teams are working with identical data in a timely manner.

Just as important as automated distribution is the availability of data at work sites with difficult network connections. Ideally, an engineer should be able to "check out" a calibration project from the central database on a laptop, so that it is available in the local network for shared access.

Data Management in the Enterprise

A solution for data management is closely linked to the tool chain for developing electronic systems. Changes to the calibration data have a direct relationship to requirements management systems, change management systems and many other ALM applications (ALM: Application Lifecycle Management). Efficient CDM systems exhibit high flexibility and can be integrated in existing systems via automation interfaces, APIs (Application Programming Interfaces) or Web services. An integration with OSLC (Open Services for Lifecycle Collaboration), for instance, makes it possible to implement universal traceability of changes.

Figure 2: A scalable solution covers all requirements for process assurance, data quality and variants management.

High Acceptance by Adapting to Specific Aspects

A CDM system has specific requirements when it is used across different technical areas. That is because the work procedures for calibrating a chassis controller differ noticeably from those for an engine controller. The different requirements of the domain and the business organization result in different methodologies. When introducing data management it is important to represent the different work modalities of the technical departments as ideally as possible, instead of prescribing a "standard process".

A Scalable Solution for all Uses

High levels of calibration data quality, process adherence and efficient variant management are the primary prerequisites for successful calibration projects. Vector offers a scalable CDM product with solutions ranging from the individual calibrator to calibration departments and the complete company-wide solution.

For the calibration engineer, "CDM Studio" serves as an efficient tool for editing parameter set files **(Figure 3)**. The parameters generated in the ECU calibration process can be conveniently displayed, compared and edited. Due to the many different import options it can be used throughout the complete process – from software development in MATLAB to online-calibration. CDM Studio supports all commonly used measurement and calibration tools such as CANape, INCA, MARC and others.

When multiple calibrators work on a team, solutions are needed for reliable data storage and data management. "vCDM" is a database-supported platform for calibration teams and the entire company. The system avoids data conflicts with work packages and assigns parameter responsibilities. It detects and resolves parameter conflicts and enables seamless tracking of data changes.

A large number of variants can be handled reliably by managing dependencies, automatically computing parameter values and reusing work packages. This leads to consistent calibration variants with higher data quality, which can be used in data mining and report functions for further quality and efficiency improvements. Finally, the integrated CDM Studio is used to graphically display and process the calibration data. This results in a scalable solution with the same look and feel, so that users can easily switch between different applications.

Figure 3: CDM Studio shows the status of ECU calibration in a well-organized layout.

Conclusion

In ECU calibration, tens of thousands of data values must be determined and managed for each individual vehicle variant. To attain the required quality, it is necessary to have an assured process and universal tool support. It is also essential to consider the individual working methods of different departments and businesses.

Wolfgang Löwl (Group Leader in Tool Development at Robert Bosch GmbH, Powertrain Area): "Together with Vector, BOSCH developed a high-performance CDM system which fully met our special requirements. We have been using it successfully for 10 years now."

Outlook

In the future, it will become increasingly more important to utilize the enormous quantities of data – not only to improve quality, but to also add value. For example, data mining algorithms and model-based calibration can be used to automatically generate new calibration data sets from the history of existing calibration data. These data sets can then be validated with special models. In the future, this will make it possible to shift increasing numbers of tasks from the vehicle to the office.

Translation of a German publication in Automobil-Elektronik, 6/2015

Image rights:
Vector Informatik GmbH

Links:
Website Vector: www.vector.com
General Information Calibration Data Management: www.vector.com/cdm
Product Information CDM Studio: www.vector.com/cdm_studio
Product Information vCDM: www.vector.com/vcdm
Product Information CANape: www.vector.com/canape

Stephan Herzog
is employed at Vector Informatik GmbH as a Business Development Manager responsible for the Measurement and Calibration areas.

Andreas Patzer
is employed at Vector Informatik GmbH as team leader in the Measurement & Calibration product line; he is responsible for Customer Relations and Services.

Michael Vogel
is employed at Vector Informatik GmbH as Business Development Manager responsible for vCDM.

Rapid Control Prototyping for Functional Development of Dynamic Driving Systems

Case Study

The Challenge

To quickly validate control loops and functions on a real-time platform in a real system

The complexity of the system network and of coordination effort is increasing due to the growing number of chassis control systems and driver assistance systems, as well as their mutual functional interactions. This means that validation of the functional model at BMW Group must be performed on the real system early in the development process – while avoiding extensive changes to the ECU.

The Solution

Rapid Control Prototyping with "Bypass/Fullpass" solutions and real-time capable hardware

To conduct tests of the control strategy in early development phases without existing ECU hardware, engineers need a Rapid Control Prototyping Platform. Here, BMW is using the VN8910 network interface with integrated real-time computer instead of a prototyping ECU. The input variables of the bypass model are read from the ECU via the VX1131 measurement and calibration hardware; the output variables are then computed in the VN8910 and sent back to the ECU. The "Bypass/Fullpass" model is created from MATLAB/Simulink. Internal model variables can also be measured and calibrated with CANape.

The Advantages

Fast, simple and cost-effective testing of different open and closed loop control methods

Improved and dynamic test options benefit BMW developers with a higher level of functional quality:

> BMW has a universal tool chain for the entire development cycle with CANape, the VX1131 system and the VN8910.
> Low integration/migration effort, because the same control models are used over the entire development process.
> The CANape MATLAB Integration Package makes it very easy to connect the model's inputs and outputs to real variables.
> Users can quickly generate, compile and link the code of the function algorithms. This produces A2L, DLL and MAP files that can be read directly into CANape.
> CANape can be used to feed parameter values back into the MATLAB/Simulink development environment.
> It is easy to acquire, condition and visualize data of the relevant control parameters with CANape. Users can conveniently navigate through the model and directly access parameters and measurement values.
> Quick reaction time in making modifications – and without programming effort; design errors are found quickly.

Riding on the Razor's Edge

Optimal Parameterization of an EngineController for Drag Racing

In calibrating engine controllers for production vehicles, electronic developers typically work with engine test stands and numerous test drives over different route scenarios. However, no such tools are available for special engine controllers used for drag racing. Using Vector's CANape measurement and calibration tool enables an engine controller to be calibrated for top performance without using a test stand while staying within a tight budget even under the continual risk of destroying the engine after just a few test runs.

Anyone who has been to a weekend race event and seen mid-class vehicles – with production engines with hundreds of horsepower – covering a distance of a quarter mile (402.34 m) with deafening noise and incredible accelerations, is very likely watching a drag race **(Figure 1)**. Because top engine performance is required very quickly in such acceleration races, a large share of development effort goes into calibrating the engine controller. The art of the race team's efforts is to achieve optimal results with a minimal budget. It is necessary to approach the stress limits of the engine so closely that it delivers maximum power without being destroyed. Not only the driving but also the process of calibrating the engine can best be described as "a ride on the razor's edge."

Optimally Calibrated Engine Controller Enables Maximum Power

In a personal endeavor, it takes a great deal of passion and enthusiasm to spend the time and money it takes to build and maintain a vehicle for drag racing. The key item here is the engine. A production engine is purchased, which is then modified by mechanical rebuilding to prepare it for the demands of racing. While the rebuild represents one side of the coin, the other involves calibrating the engine controller. All sorts of challenges must be mastered here, since the parameters of the production engine controller hardly harmonize with the modified engine any longer.

Measuring and calibrating ECUs is a challenging but daily routine in the work of carmakers and suppliers involved in production development. While calibrators run various course scenarios with the engine on the test stand and in the vehicle, they access internal parameters and ECU measurement variables via an A2L description file and find the optimal parameters. The complexity of this task significantly increases due to a whole series of constraints. On the one hand, numerous engine and vehicle variants need to be considered, and different emissions standards need to be met. At the same time, the ECU must be imprinted with an OEM-specific driving behavior. All optimizations are also subject to the premise that certain fuel economy limits must be observed. Engine calibrating is simplified by the fact that calibrators and software developers control the entire software process together. This ranges from creating the code and developing the software to the compiler/linker run, A2L generation and the flash process.

Optimal Results in just a Few Test Drives

The engine calibrating process for drag racing is fundamentally different. Neither maintaining fuel economy nor supporting different engine or vehicle variants play a role here; rather, all efforts are directed toward one goal: covering the approx. 400 meter distance as quickly as possible. Furthermore, the racing teams are not corporations with strong financial backing; rather, they are typically private individuals pursuing an expensive hobby. If faulty calibration leads to engine destruction, a lot of money needs to be spent to buy a new one.

No test stands are available for engine test runs either. For one, there are no suitable test stands due to the lack of demand for this niche application. Secondly, it is not possible to optimize the engines at their maximum speeds of up to 10,000 rpm and up to 3.5 bar charge pressure in quasi-static operation. The loads are so great that the engines could only withstand these speeds for a brief period of time (2 to 3 seconds per gear) without succumbing to heat overload.

The only feasible approach for the race teams is to acquire as many measurement variables as possible during the drive and then optimize calibration parameters based on this information. However, this approach too necessitates living with all sorts of limitations. On the one hand, the engines can only be used for a few drives before they need to be replaced. On the other, the drives usually last less than ten seconds. Therefore, the key to success or failure of the optimization process lies in finding an extraordinarily rational method.

Special Engine Controller Replaces Production Device

A highly efficient engine controller for drag racing comes from the maf-map-engineering company of Berlin, Germany. This company, founded on a passion for car racing, offers a complete solution that gets maximum power out of the engines. Its capabilities are illustrated by the fastest VW Polo in the world, which trumps with 1,047 horsepower. Its performance is based on the ECU481 engine controller, whose entire hardware and software was developed by the company independently for ideal control of all components. The software is created based on physical models, whereby the Scilab modeling environment is used with an associated code generator for the functional layer. The basic software is still manually coded in C.

Since no test stand operation and only a few short test drives are possible, a primary focus is on reliable acquisition of all relevant parameters from the ECU via a cost-effective interface. That is why the standardized measurement and calibration protocol XCP on Ethernet was chosen. Early on, a decision was made to search for a satisfactory professional tool which lead the team to the CANape measurement and calibration tool from Vector.

Figure 1: To optimally calibrate the engine controller, calibration parameters are adjusted in real-time during the test run based on acquired measurement results.

Figure 2: Besides being used to develop a control algorithm that is fed concrete data from ECUs, bus data, analog data, etc., CANape also covers other applications such as online computations during a measurement.

Automated Parameter Optimization in Real-time

Key ECU parameters are calibrated by the maf-map-engineering specialists or the individual racing team; they are not calibrated after the test run, but during the test run itself – in real-time – based on acquired measurement results. Because of the extremely short test driving times it is impossible for the driver to mentally note all of the data, derive meaningful decisions from it and still send the right values to the ECU. CANape functions that enable automation of this process are especially beneficial here. Code is generated from Simulink models using the Real-Time Workshop code; after compiling and linking, it is run as a DLL in CANape. At runtime, during the test drive, the algorithm in the DLL obtains measurement data from the engine controller, uses it to compute optimal parameters and autonomously calibrates the parameters in the controller via XCP mechanisms and CANape **(Figure 2)**.

Since the effort required to develop separate application models for all parameters would be excessive, many parameters are still adjusted manually. After the drive precise analysis of the logged measurement data with CANape reveals any critical issues and enables rapid implementation of corrections **(Figure 3)**. In addition, the tool helps to reduce the number of necessary test drives to a minimum. New parameter values are derived from the results, which developers either set online in the controller's RAM or implement as new model values before the next code generation. More evidence that the approach works well and is on the rise is provided by the 2010 King of Germany (KOG) event, in which a vehicle with an engine controller from maf-map-engineering took first place in the class of front-wheel drive vehicles. Meanwhile, this success in the general classification has been noted by other industries. Contacts have already been made with boat racers who are looking for a comparable solution for their race boats: an engine controller optimally calibrated for this specific field of application with efficient measurement and calibration capabilities.

Figure 3: CANape visualizes the various parameters and offers convenient calibration options

Translation of a German publication in Elektronik automotive, 4/2011

Image rights:
Lead Figure and 1: Dr. Bernd Seydel
Figure 2 and 3: Vector Informatik GmbH

Links:
Homepage maf map engineering [only German]: www.maf-map-engineering.de
Homepage Vector: www.vector.com
Product Information CANape: www.vector.com/canape
Product Information CANape CANape Option Simulink XCP Server: www.vector.com/simulinkxcpserver

Andreas Patzer
After training to become an electronics technician in the IT field, from 1986 to 1993, Andreas Patzer studied Electrical Engineering at the Technical University of Karlsruhe, where he specialized in measurement and control engineering as well as information technology and industrial automation. After graduating, Mr. Patzer worked in the communications industry. In 2003, he joined Vector Informatik GmbH in Stuttgart, where he manages the interface between the customer, development and sales as a Business Development Manager for the Measurement & Calibration product line.

Model-Based Development and ECU Calibration With XCP and CANape

Case Study HOERBIGER

Figure: HOERBIGER

The Customer

HDM (HOERBIGER Drivetrain Mechatronics), a division of HOERBIGER Corporation, is a highly regarded clutch expert in the global automotive industry. It specializes in the development of dual-clutch mechatronic systems for sports cars, high-end sedans and heavy truck applications.

The Challenge

To conveniently test the behavior of Simulink models

In developing software for the second generation of a dual-clutch transmission, engineers convert the existing, manually written C code to MATLAB/Simulink models (Re-Engineering). The code is then automatically generated from the models and integrated directly in an AUTOSAR RTE. Each software module can be simulated in Simulink. However, existing MATLAB Scopes visualization options are inadequate in conducting detailed data analysis. The process of optimizing parameters is also time-consuming and rather inconvenient, requiring modification of values in the MATLAB Workspace or generation of specific GUI elements.

The Solution

CANape as a user interface for parameterizing and visualizing Simulink models and internal ECU data

The simplest way to interface CANape with the model in Simulink is with the Simulink XCP Server. Users have the same options available here as in connecting to an ECU: drag & drop selection of measurement and calibration parameters from the description file and visualization in display and calibration windows. The necessary A2L description file is generated from the Simulink model at the press of a button; this enables read and write access to parameters in the model without requiring additional instrumentation of the model.

The Advantages

Simulink models can be visualized and parameterized conveniently and efficiently

The CANape Option Simulink XCP Server is ideally suited for analyzing model behavior:

> The standard XCP protocol enables use of the same CANape configuration over the entire development process. Regardless of whether the model, a rapid prototyping platform or the ECU is connected.
> To test the model as realistically as possible, logged measurement data can be fed into the model as input parameters at runtime.
> It is easy to visualize the measurement data and modify parameters in the various CANape windows. Object-specific model instrumentation is unnecessary.
> Calibration Data Management with CDM Studio makes it easy to edit and manage parameter set files in the model. Users can copy and merge different parameter sets, download to the Simulink model and save parameters in different formats, e.g. as an M-script in MATLAB format.
> Simulation results are available in MDF format. This enables direct comparison with measurement data from the vehicle and from the manual or automated evaluation in CANape.
> The solution is scalable: in simulations that are especially computationally intensive, processor loads can be distributed to 2 computers.

AUTOSAR ECU Development Process Using DaVinci and MICROSAR from Vector

English Translation of a Japanese Technical Article from Mitsubishi Motors Corporation

AUTOSAR is a group paving the way for the standardization of software platforms across Electronic Control Units (ECU). Its activities have gained momentum in recent years. In order to find out how AUTOSAR could best be applied to ECUs, we – Mitsubishi Motors – joined forces with our supplier, Mitsubishi Electric, to test the development process and evaluate the tool chains relevant to the implementation of AUTOSAR software components on ECUs. For evaluation we used the EV-ECU of the i-MiEV electric car. Here is an overview of the project.

AUTOSAR

As the number of on-board computers (ECUs) in automobiles increase along with their functionality, there has been an increase in software implementation. As a result, the Europe-based AUTOSAR has been gaining attention as the leading platform for the standardization of basic software specifications. This standardization covers basic software modules e.g. for I/O access, CAN communication and the operating system. Furthermore, the standardization includes XML formats for exchanging configuration data.

Preparing for AUTOSAR

As it is not yet clear how AUTOSAR will affect the existing ECU development processes and development environments of automobile OEMs and suppliers in Japan, implementing AUTOSAR without careful thought may cause confusion and disruption. This is why we joined forces with Mitsubishi Electric, our ECU supplier, and Vector Japan Co., Ltd. to test the AUTOSAR ECU development process and verify the development environment (tools) as a preparation for implementing AUTOSAR software components. This project also covers evaluation of software reusability with AUTOSAR.

AUTOSAR Evaluation Overview

For this project, we evaluated the EV-ECU, the main control ECU for the i-MiEV electric car. We decided to extract and use a part of Mitsubishi Motor's model-based software from the EV-ECU. To carry out the evaluation, the extracted software was divided and assigned optimally to two ECUs using microcontroller evaluation boards **(Figure 1)**. The AUTOSAR ECUs designed for the purpose of this evaluation and the EV-ECU were then connected using a CAN communication network. **(Figure 2, "Local CAN")**.
To evaluate software reusability, the software assigned to one of the two AUTOSAR ECUs was divided again and applied to another AUTOSAR ECU **(Figure 7)**. This implementation process was compared to the implementation process of commonly reused software, to evaluate software reusability with AUTOSAR.

Figure 1: Division of ECU

ECU Development Process Testing

Here is a summary of the tests we carried out with a focus on the ECU development process with AUTOSAR.

> Differences from legacy ECU development processes
> Division of roles between automobile OEMs and suppliers
> Tool chains
> Data definition file format

Figure 2: Overview of vehicle network

This project evaluated the processes listed below. We have summarized the details of the workflow and testing criteria, as well as the results for each process. The development and implementation environment can be seen in **table 1**.

> Vehicle system design
> Vehicle network design
> ECU application software design
> ECU software system design
> ECU software detailed design
> Coding/ Implementation

Vehicle System Design

A certain system requirement specification was created for the evaluation project. We designed a system focusing on coordination between the EV-ECU and the extracted software. The evaluation included the allocation of functions to each ECU, the network configuration and choosing suitable AUTOSAR BSW. The system requirement specification has been revised according to the evaluation results. The allocation of functions to each ECU was realized by assigning software components (SWC) using the AUTOSAR method. It depends on the OEM and the project whether to define the SWC and BSW configuration during the vehicle system design, or during ECU design. Since we were dealing with a small-scale system and with BSW provided by the specific software vendor, we chose the former option.

Design tools	
AUTOSAR Evaluation Bundle [Vector]	
Network design	Network Designer
RTE design	DaVinci Developer
OS/BSW design (Configuration)	DaVinci Configurator Pro GENy
Model-based development tool [Mathworks]	
Model design	Simulink®
Code generation	Real-Time Workshop® Embedded Coder™
Other development environment	
Compiler [IAR]	IAR compiler
Microcontroller [Renesas]	R32C
Microcontroller evaluation board [Renesas]	Evaluation board for R32C

Table 1: Development and implementation environment used for evaluation

Vehicle Network Design

On the vehicle network of an i-MiEV, a gateway ECU was set up where the EV-ECU was located, and a local CAN communication network was constructed using the modified EV-ECU and the AUTOSAR ECUs **(Figure 2)**.

We used Network Designer for the CAN communication network design **(Figure 3)**. The CAN communication data was exported as a system description file in ARXML format. As far as the communication network design is concerned, the design method remains unchanged except for the tool and data transfer file format. In addition to the ARXML format, Network Designer also offers various other formats such as DBC and FIBEX which could be used instead of the AUTOSAR format. As network design also takes the entire vehicle into consideration, the process is carried out by the OEM.

ECU Application Software Design

We edited the software extracted from EV-ECU using Simulink®. According to the software model's inputs and outputs, we defined the AUTOSAR compliant ports/interfaces and runnable entities.

In terms of software reusability, the SWC input/output port and data configuration require caution, especially with regard to the collection of data and configuration of the port. If the data does not match other SWC port configurations when reusing SWC, the port configuration will have to be changed **(Figure 4)**. On the other hand, creating a port for each data makes the SWC configuration too complicated and is therefore not recommended. For SWC ports in line to be reused, it is important to review the port configuration beforehand.[1]

Regarding the interaction of tools, it should be taken into consideration that several tools might change the Software Component Description file (SWC Description).[2]

[1] In the latest AUTOSAR release 4.0, these problems are reduced since it is possible to explicitly define the mapping between port data.
[2] Therefore, the tools need to support a roundtrip and must not overwrite the data created by another tool.

Figure 3: Network Designer™

Figure 4: Points to note in terms of SW-C reusability

The model's software source code and the SWC Description file were automatically generated using Real-Time Workshop® Embedded Coder™. In this evaluation project, the ECU application design was carried out by Mitsubishi Motors. For series projects the division of roles should be discussed before design.

ECU Software System Design

The ECU software system design involves the allocation of SWC to ECU and designing the ECU input/output data. We used DaVinci Developer™ **(Figure 5)** for this purpose.

First, the SWC Description file generated from Simulink® was imported into DaVinci Developer and the SWCs were allocated to each ECU. Next, the System Description file generated from the network design was imported and the CAN communication data and SWC input/output data were mapped. DaVinci Developer features an automatic data mapping function, which is extremely useful when dealing with large amount of data. The data configured by DaVinci Developer for each ECU is exported as an "ECU Extract of System Configuration" file in ARXML format.

As the network design is frequently being updated with regard to the ECU development process, it is important for the Runtime Environment (RTE) design tool that the System Description can be simply updated and that the content of such updates can be easily verified.

The ECU software system design was carried out by Mitsubishi Motors. For small-scale development projects (such as closed projects with ECU units), it may be efficient for the supplier to carry out the design. However when designing software partially or designing a cooperative control with multiple ECUs, it is better for the OEM to do the design. By mapping the communication and SWC data, the OEM can verify the consistency of the data in advance. This is useful, as inconsistency of data can lead to unnecessary work between the OEM and supplier.

Figure 5: DaVinci Developer™

ECU Software Detailed Design

ECU software detailed design involves the configuration of ECU device drivers and design of OS and ECU basic functions (CAN communication, error diagnostic functions, etc). The design uses the ARXML-format "ECU Extract of System Configuration" files gene-rated above.

Figure 6: Vehicle testing

Specific design details include the allocation of Runnable Entity to OS tasks and the configuration of the relevant BSW parameters. We carried out the configuration in this case using DaVinci Developer, GENy and DaVinci Configurator Pro. The ARXML-format "ECU Configuration Description" file describes the specific configuration of the basic software modules and is shared between the involved tools. As ECU software detailed design is dependent on the hardware configuration, it is carried out by the supplier.

Coding/Implementation

The configuration code for the basic software modules is automatically generated by DaVinci Configurator Pro and GENy. This code is compiled along with the static code of the basic software into the executable code, which is then written to ROM. We used an IAR compiler for compiling. In the past, the codes were generated manually by hand coding. In contrast, AUTOSAR enables the usage of configuration tools and automatic code generators for coding. As is the case with legacy methods, coding/ implementation is carried out by the supplier.

Functional Testing

We equipped an i-MiEV with the AUTOSAR ECUs we designed and ran vehicle tests **(Figure 6)**. As the functions formerly processed by one ECU are now distributed to several ECUs, there was some delay in communication between the ECUs. Nevertheless, we were able to onfirm that the car was controlled correctly by the AUTOSAR ECUs.

Testing Reusability

We then tested the reusability of SWCs. By assigning the SWCs to an additional ECU (they were assigned to two ECUs during ECU development process testing) and changing the assignment pattern **(Figure 7)**, we estimated the process and man-hours needed in porting the SWCs. We were then able to compare the legacy design and the design according to the AUTOSAR method.
Here are the tasks involved in porting the SWC:

> Redesigning the network
> Assigning SWCs to ECUs
> ECU software detailed design

Redesigning the network is necessary for both legacy and AUTOSAR methods and the processes do not differ by much. The application software and SWC interface require no modification in porting the SWCs. Assigning SWCs to ECUs and data mapping with CAN communication data are both easy to do by using the tools. This has not been the case until now: adapting application software interfaces usually took much time and effort.
Regarding the detailed design of ECU software, it is possible to reduce the configuration settings when carrying over parts of the existing configurations. As a result, there is a great potential for reducing the time and effort that usually goes into adapting interfaces and designing middleware.

Figure 7: Evaluating reusability and ease of porting

Figure 8: ECU development process with AUTOSAR (Part 1)

Evaluating Tool Chains

As tool chains influence the quality of the software greatly, it is important to discuss issues thoroughly with the supplier. Considering the tool dependence, it makes sense to use tools from the same vendor throughout the entire development process. The tool chains evaluated in this project can be seen in **figures 8 and 9**. The pink areas show the work carried out by the OEM and the light blue areas indicate the work carried out by the supplier.

Figure 9: ECU development process with AUTOSAR (Part 2)

From network design to AUTOSAR RTE/BSW/OS detailed design and code generation, Vector offers tools encompassing the entire process. As the tool chain is pre-established, there is consistency and uniformity in the data throughout the process, increasing reliability in development.
The tools could be improved by expanding the compatibility of data exchanged with the most commonly used tools from other companies. As model-based development with AUTOSAR has proven to be a valid development method for the future, we hope this issue will be prioritized.
Also, GUI-based software design is efficient in the design phase, but it is not suitable for comparing and verifying the design data. An additional function to export the data designed by each tool in table format would be desirable. The process of testing the communication signal's configuration parameters and the transmission and reception of the signal was very complicated. This process would be much more efficient if the data could be exported in table format.[3]

Summary

By testing the ECU development process using AUTOSAR, we were able to gain an understanding of the workflow and division of roles. We were also able to clarify the necessary mutual consultation between OEMs and suppliers for each process. With AUTOSAR, the OEM can take over the designing process usually associated with the supplier side. Furthermore, there is room for optimization in overlapping areas such as ECU software system design, where both OEM and supplier should be more flexible in the future.
Regarding reusability, we found that this goal can be easier reached with AUTOSAR than with legacy methods. However, we focused solely on the sender-receiver port format for the SWC input/output, therefore the reusability for server-client port formats still needs to be evaluated.
Through the evaluation project, we were able to establish the technical details to some extent. In the future, we will actively exchange information and opinions with suppliers and tool software vendors to put in place a framework for the introduction of AUTOSAR at Mitsubishi Motors, from the introduction process to implementation.

Afterword

We received great support for this evaluation project. Compilers were kindly provided by IAR Systems, and microcontrollers by Renesas Electronics Corporation. Though we were not able to realize their offer for this evaluation, an offer from Fujitsu Semiconductor Limited for collaboration was greatly appreciated. And of course, Mitsubishi Electric's cooperation and Vector Japan's tool software were invaluable. We'd like to take this opportunity to thank everyone for their support.

Translation of a publication in the Japanese Vector Journal

Image rights:
All figures: Mitsubishi Motors Corporation

Literature:
www.autosar.org

Links:
Homepage Vector: www.vector.com
Information on Vector's AUTOSAR products:
www.vector.com/autosar

Author: Yuichi Kamei
Mitsubishi Motors Corporation
Electronic Control System Development
Electronics Engineering Dept.
Development Engineering Office

[3] In the latest generation of the tool chain, Vector provides according functions to compare the design data.

Integration of External Functions of a Microcontroller with vSBC Driver for System Basis Chips

Case Study vSBC

vSBC is a driver implementation for system basis chips. It is part of the Vector AUTOSAR solution MICROSAR.

The Challenge

Integration of external functions into an AUTOSAR project requires a suitable driver.
The AUTOSAR architecture of an automotive electronic control unit (ECU) contains several dedicated basic software components like transceiver drivers for CAN / LIN as well as for handling watchdog functionalities. Semiconductor vendors are combining these different functionalities to a solution, in so called system basis chips. A suitable driver is necessary to be able to integrate these external functions into an AUTOSAR project.

The Solution

vSBC driver from Vector.
Vector has many years of experience in developing basic software components and also drivers, especially for communication related peripherals. The vSBC driver offers the following functionalities:
> Designed to be used in an AUTOSAR 4 environment.
> For devices with built-in CAN / LIN transceiver functionalities, the vSBC driver offers abstraction of CAN / LIN transceiver related hardware features. This enables 1:1 mapping to AUTOSAR.
> Abstraction of watchdog related hardware features covered by watchdog driver. This enables 1:1 mapping to AUTOSAR.
> Additional SBC functionalities, which are not part of the AUTOSAR architecture, can be realized by using a RAW-API.

The Advantages

Only one driver for serial bus interfaces is needed.
> vSBC driver is developed according to Vector's certified development process to be able to realize ISO 26262 applications up to ASIL D.
> Independent of the microcontroller the software is executed with, only one driver for serial bus interface (e.g. SPI, I²C) is needed.
> Independent of other Vector software products. There is no need to use Vector's MICROSAR basic software.
> Configurable with any AUTOSAR configuration editor, e.g. DaVinci Configurator Pro from Vector.

Availability

Please contact Vector for detailed availability information of drivers for system basis chips.
www.vector.com/microsar

System Basis Chip is connected to the microcontroller via serial bus interface (e.g. SPI, I²C).

AUTOSAR METHODOLOGY IN PRACTICE

The partners Daimler, Hella and Vector recognised the great potential of AUTOSAR very early on, and report on their experiences in introducing AUTOSAR development methodology. From several production projects, they have learned where benefits can be easily attained but also note the hurdles that had to be overcome, and they identified obstacles that must still be addressed today.

AUTHORS

DR. RALF BELSCHNER
is Team Leader for Standard Software at the Mercedes-Benz Development Department in Sindelfingen (Germany).

STEFFEN HERZ
is Director of Engineering Standards Electronics at Hella in Lippstadt (Germany).

DIPL.-ING. (FH) JOCHEN REIN
is Group Leader Product Management and Sales for Embedded Software at Vector Informatik GmbH in Stuttgart (Germany).

FUNCTIONALITY AS DRIVER

For Daimler, the key drivers for introducing AUTOSAR were the large number of new functions and their progressive concentration in just a few ECUs. That has resulted in a high increase in communication requirements and made it necessary to use a FlexRay bus system. New basic software (BSW) and a new exchange format were necessary. To handle all bus systems uniformly, the course towards AUTOSAR was set. One of AUTOSAR's greatest strengths lies in the fact that many new functions are available in a standardised form. This applies to the support of new bus systems – like Ethernet – and functions such as partial networking, functional safety, security and multicore support.

NEW METHODS AND EXCHANGE FORMATS

In AUTOSAR, the automotive industry has created a standard that places the development of ECU software on a new methodical foundation. This enables a significantly improved systematic approach to collaborative work between carmakers (OEM) and suppliers, which in turn makes it possible to combine functional contents from different sources in a process-assured way. The new method requires standardised data exchange formats and above all extended formats. While familiar exchange formats like DBC, LDF or Fibex are somewhat bus-specific and only describe the communication, the exchange format AUTOSAR System Description represents a significant extension, ❶. The application software is described by its software components (SWCs), the relationship between the SWCs and the networks (data mapping) and the use of service functions of the BSW (service mapping) is defined.

The SWC definition lets the OEM formally describe its own contents in the application software and transfer it to ECU suppliers (Tier 1) in a process-assured way. Daimler recognised this and still sees this as a significant step towards consistent implementation of a system- and function-oriented approach. Previously, the interfaces description for the application software was completed over the course of development and was the responsibility of the supplier. Now developers in the various technical departments at Daimler are faced with the task of distributing an ECU's functional content to software components early in the development process and defining internal ECU interfaces, ❷. Formal description forces completeness and consistency very early on, and it shifts the effort in the development process forward – to the OEM. This is a good example of "frontloading"; however, this only brings the desired benefits after working through a learning curve.

For the Tier 1, early definition means a loss of flexibility at first, since local changes to the interfaces can no longer be performed so simply. These changes must be exchanged between the Tier 1 and OEM, ❸. This indicated the need for high-performance difference, merging and report algorithms. Vector has invested heavily in this area and now offers mechanisms that significantly reduce effort and errors.

One clear advantage that Hella sees in integrating application software and configuring the BSW is that specifications are more precise. However, this advantage is only realised when the System Description is updated frequently enough in early development phases. This is the only way to assure consistency of the data on both the OEM and Tier 1 sides. Over the course of development, it has been found that clear release planning is needed for data exchange in both directions. Along with the ECU extract of System Description (subset of the System Description for an ECU) this also involves consistent exchange of the diagnostic parameterisation. Currently, the compatibility of architectural levels is still a largely unresolved problem: The Tier 1 develops its own software architecture to efficiently handle the different OEMs. In turn the OEM develops its own software content for all of its Tier 1 suppliers. Since these two approaches are usually not identical, an integration problem typically results, which might need to be resolved by adaptation modules. AUTOSAR standardisation is desirable in this area but it runs into problems if it imposes limitations on competitive differentiation. If software components are also provided by third parties, synchronisation of the data becomes that much more important. Here too, the goal is to considerably simplify exchanges by formalised information – so that everyone is speaking the same language.

❶ The AUTOSAR System Description contains descriptions of the communication and other information on the software components and the basic software

❷ The OEM defines the structure of the software components with deep insight of the application

EFFICIENT DEVELOPMENT WORK

The tool producer is called upon to make the dramatically increasing number of configuration parameters manageable. The developer must be provided with as many algorithms as needed to correctly and automatically set dependent parameters and to recommend meaningful combinations. Both, the curse and the blessing of AUTOSAR are closely intertwined here: The high level of configurability of the BSW leads to the large number of parameters, but it also permits reuse in many projects. Therefore, the tools represent an important factor for success. A generic editor is insufficient here. The tool must offer specialised views and algorithms for maximum user support. For example, this makes it possible to edit dependent parameters consistently. Close coordination between the project partners has made a decisive contribution towards improving tools and processes. Although the work flow is roughly specified by AUTOSAR, only in practice does it become apparent which application cases occur and where the tools can be structured more efficiently by adding appropriate functions. All participants were faced with another challenge as well in the introductory phase of AUTOSAR: Over the course of the project, new requirements arose for the BSW modules and description formats. Sometimes it was necessary to switch over to new AUTOSAR minor or patch versions during development or even temporarily implement an extension to the AUTOSAR specification. At the same time, in-house tools that Daimler had used and extended for AUTOSAR were adapted to the tools from Vector. In addition, the application cases had to be considered from a Tier 1 perspective to permit an automatic transfer of the created configurations.

❸ To ensure the use of consistent data, the Tier 1 must regularly report its changes to the OEM and always get the latest version from the OEM

COORDINATED PROJECT HANDLING

In closing, it is necessary to examine the aspect of parallel work on an ECU project – a basic theme for the larger ECU developments that are typical at Hella. Central data storage makes it more difficult to work in parallel on an ECU project with multiple teams because everyone needs to access the data simultaneously. Arrangements between the teams which had previously pragmatic approaches of working together cannot simply be represented in the AUTOSAR description files. Therefore, it is advantageous to subdivide these files into elements, so that they can be stored separately in a configuration management system. Functions are also needed for representing differences and for consistent merging. The tools must relieve the user of as much manual work as possible. The quality of error messages and warnings proved to be very crucial here. Initially, too little attention was paid to this aspect. Now a significantly better level has been attained by intensive informational exchange between users and tool developers.

REVOLUTION OR EVOLUTION?

Today, all of the partners view the project in positive terms. For Daimler, the migration to the System Description was a logical and necessary step in mastering the increasing growth in the networking of vehicle functions. Combining the new AUTOSAR standard with multiple bus-specific configuration formats definitely would not have run nearly as smoothly. And with increasing experience consistently implementing this System Description step let all participants realise the full potential of AUTOSAR. Hella sees the greatest benefit in the improved and structured approach to collaborative work with OEMs. This method should be adapted on a broad scale, so that investments in AUTOSAR will lead to the anticipated gains in long-term efficiency and quality.

ON-BOARD DIAGNOSTICS MEETS AUTOSAR

As powertrains become more electric, the number of on-board diagnostics relevant Electronic Control Units (ECUs) continues to grow. Vector provides a summary of the on-board diagnostics functions integrated in the AUTOSAR basic software (BSW) and which use cases they support.

AUTHORS

DR. ING. THOMAS NECKER
is Team Leader in the Embedded Software Area at Vector in Stuttgart (Germany).

DIPL. ING. OLIVER GARNATZ
is Product Manager in the Embedded Software Area at Vector in Stuttgart (Germany).

BACKGROUND

For vehicles with combustion engines, increasingly more stringent emission limits have been set in recent years. To continue meeting these limits over years of vehicle operation, an on-board diagnostics (OBD) system must monitor components and systems to ensure fault-free operation.

Typically, three classes of ECUs are defined for each vehicle. The Master ECU takes on the role of collecting, conditioning and supplying central data and it controls warnings to the driver. Primary ECUs are ECUs that acquire local data in their fault memories and communicate with the scan tool, while Secondary ECUs only report to a Primary or Master ECU.

Functions for monitoring components and systems are subdivided into two categories: Major Monitors are relevant for systems that have a direct effect on emission values (for example fuel and exhaust recirculation systems). Comprehensive Component Monitors check the systems that are needed for the Major Monitors or systems that affect emissions indirectly. They include functions related to battery regeneration while braking the vehicle, battery management and climate control systems.

OBD FUNCTIONS AS PART OF THE AUTOSAR BSW

The rapid spread of electrically assisted drives leads to a growing number of Primary ECUs that have Comprehensive Component Monitor functionality. AUTOSAR defines the OBD functions that should be implemented in the DCM (Diagnostic Communication Manager), DEM (Diagnostic Event Manager) and FIM (Function Inhibition Manager) modules. To a high degree, they reference definitions in the OBD standards and directives. The concrete implementation and configuration, for example calibration strategy and relation to UDS (Unified Diagnostic Services) fault memory, must be defined by software suppliers in cooperation with the OEMs.

The DCM module implements OBD-specific services, ❶. Moreover, OBD requests are either prioritised over UDS requests, or they are processed in parallel. The DEM module also contains various OBD extensions that are additional to those of UDS. They include modified error status behaviour, storing persistent DTCs, saving freeze frame data and calculating the In Use Monitor Performance Ratio (IUMPR). In combination with the FIM module, incrementing of the IUMPR counter is deactivated as soon as certain malfunctions are detected.

Production-ready OBD extensions of the DEM, DCM and FIM modules for five different automotive OEMs are already available in the AUTOSAR 3 and AUTOSAR 4 basic software from Vector.

Additional services (modes) for OBD

$01: Request current powertrain diagnostic data
$02: Request powertrain freeze frame data
$03: Request emission-related DTCs
$04: Clear/reset emission-related DTCs
$06: Request on-board monitoring test results for specific monitored systems
$07 Request emission-related DTCs detected during current or last completed driving cycle
$08: Request control of on-board system, test or component
$09: Request vehicle information
$0A: Request emission-related DTCs with permanent status

❶ The OBD-specific extension of the DCM module contains nine additional services

VECTOR

Development of a Reference System for AUTOSAR ECUs and System Studies

Case Study

The Customer

A successful European heavy-duty vehicle producer that is playing a leading role in the introduction of AUTOSAR. Its mature E/E processes enable efficient development of AUTOSAR-based ECUs.

The Challenge

To develop a reference system with multiple networked ECUs based on a specified development process. The system must support CAN, J1939 and LIN based on the Vector AUTOSAR basic software MICROSAR.

The reference system should support key parts of the OEM AUTOSAR functional content and the following tasks:

> Validate correct implementation of the specifications and of the OEM's predefined development process
> Verify system behavior
> Execute performance measurements
> Validate new software revisions
> Reproduce error states

The Solution

The system environment consists of four exemplary ECUs, the CANoe test software and the VT System test hardware with various slide-in modules.

Four typical ECU classes were defined, and each was represented by an ECU. This produced a meaningful and well-structured mapping of the later overall system.

To represent the system functions, several demo applications were implemented. They are controlled by CANoe via a newly developed CAN-based remote protocol. CANoe uses this protocol to stimulate a function or a special system state and to check the response. All diagnostic requests run as in the real vehicle over a diagnostic tester of the OEM, which is controlled remotely by CANoe.

Examples of implemented AUTOSAR functions:

> Wakeup & sleep handling
> Different transmission modes, update bits, routings
> Diagnostic services, security level and DTC handling
> J1939 dynamic DLC and BAM
> Diagnostic session control and software download
> Parameter handling ("Power lost safe")

The Advantages

Compact reference system for ECU and system development with AUTOSAR.

> "Sandbox" to develop functional extensions of the AUTOSAR basic software or the application
> Simple benchmarking of different system states
> Clearly organized user interface for manually stimulating system states
> Easy to extend the reference system with real ECUs
> Minimal effort for executing regression tests after functional changes or updates to the AUTOSAR basic software
> Good expandability based on modular structured hardware and software
> Validated and optimized development process

www.vector.com/contact

6/13

AUTOSAR in Heavy-Duty Vehicles

Integration of J1939 in AUTOSAR

Now that AUTOSAR has largely become established in the automotive industry, the heavy-duty vehicle industry is showing interest in it. In this industry too, manufacturers want to apply AUTOSAR's modular concept to save on development costs. In contrast to the static networking that is used in the automotive industry, heavy-duty vehicles use dynamic communication based on the J1939 standard. This article describes the advantages of integrating J1939 in AUTOSAR as well as its limitations.

SAE J1939 has become the established standard for networking and communication throughout the heavy-duty vehicle industry. J1939 is sometimes used directly and sometimes in the form of derived standards. For example, ISOBUS is used in agriculture and forestry, and NMEA 2000 (National Marine Electronics Association) is used for maritime applications. In Europe and in certain other regions, only a part of J1939 is used in the heavy-duty truck industry. One aspect of the heavy-duty vehicle industry that contrasts with the automotive industry is its lower production volumes, which results in higher weighting of software development costs compared to hardware costs and warehousing. Therefore, it makes sense to focus efforts on the reusability of software. In practice, however, most developers generally write new code for each ECU. In a lot of projects, even the hardware and protocol drivers are developed repeatedly, because there is no common standard to enable the reuse of drivers and parts of the application software.

The Idea of AUTOSAR – Reuse of Software

Essentially, the development of the AUTOSAR standard was mainly based on two motivations: To standardize the hardware and protocol drivers, which would simplify a change of supplier for an ECU; and to modularize the application software, which makes it easier to handle the growing complexity of ECUs. The protocol and hardware drivers are standardized in the form of basic software modules (BSW modules), while the the application software is modularized and abstracted in the form of software components (SWCs). **Figure 1** shows part of the basic software for J1939 applications, as well as the runtime environment (RTE) and the application layer.

The Software Components Model of AUTOSAR

The application software in an AUTOSAR system is subdivided into components that are hierarchically organized in compositions. From the application perspective, the communication between the components runs over AUTOSAR's virtual function bus (VFB), which hides the actual topology of the ECUs. The components define their communication needs by means of interfaces (Ports), which group data elements or operations. The VFB connects these ports on an abstract level. After the application software components have been distributed to the ECUs **(Figure 2)**, the RTE takes on the role of the VFB for an individual ECU. This means that the RTE implements all communication interfaces of the application software components, both between these components and to the basic software and thereby to other ECUs as well.

AUTOSAR for Heavy-Duty Vehicles – The Beginnings

Initially, AUTOSAR was exclusively oriented towards the needs of passenger cars, which use a static communication matrix. Consequently, AUTOSAR was designed with a static model of bus communication. And this static communication model initially prevented AUTOSAR from being used in the heavy-duty vehicle field. The first step to open AUTOSAR towards J1939 was taken with version 4.0 at the end of 2009. For this version, Vector Informatik worked together with a European truck producer to specify an AUTOSAR basic software module for the transport protocols of J1939. By that time, however, it was already evident that the AUTOSAR 4.0 support of J1939 wouldn't be sufficient for most J1939 applications.

Figure 1: Excerpt of the AUTOSAR architecture for heavy-duty vehicles

Extended J1939 Support in AUTOSAR 4.1

AUTOSAR Version 4.1 was released early 2013. The most important change related to J1939 was the ability to access parts of the CAN ID from higher layers. This enables efficient communication in dynamic networks. When a message is received, parts of the CAN ID are appended to the payload data, which makes them available to other modules. In the other direction, the sender of a message appends variable parts of the CAN-ID to the payload data to be sent. BSW modules that access the CAN-ID in this way are the J1939 transport protocol, UDS transport protocol and UDS diagnostics as well as the PDU router.
Starting with AUTOSAR 4.1, the J1939 transport layer handles the reception of long messages regardless of the used protocol variant (BAM/CMDT). To transmit messages, the J1939 transport layer switches automatically between direct transmission and transmission using one of the two protocol variants based on the actual length of the message and the destination address. Besides, the use of dynamic CAN-IDs reduces configuration effort substantially.
Working together with two truck producers, Vector also specified three new BSW modules for J1939, which are also shown in **Figure 1:**

- The "J1939 Request Manager" for the request- response protocol according to J1939-21.
- The "J1939 Network Management" for a simple address allocation method according toJ1939-81.
- The "J1939 Diagnostic Communication Manager" for diagnostics according toJ1939-73.

The J1939 Request Manager of AUTOSAR 4.1 handles sending and receiving of request and acknowledgement messages. To this end, it offers interfaces to the relevant BSW modules or to application software components via the RTE. In addition, it handles timeout monitoring for sent requests. With the help of the J1939 Network Manager, AUTOSAR 4.1 lets users create non-configurable ECUs and ECUs that

Figure 2: Distribution of software components from a library to ECUs

Message name	PGN	Meaning
DM1	65226	Active Diagnostic Trouble Codes
DM2	65227	Previously Active Diagnostic Trouble Codes
DM3	65228	Diagnostic Data Clear/Reset for Previously Active DTCs
DM4	65229	Freeze Frame Parameters
DM5	65230	Diagnostic Readiness 1
DM6	65231	Emission Related Pending DTCs
DM11	65235	Diagnostic Data Clear/Reset for Active DTCs
DM12	65236	Emissions Related Active DTCs
DM13	57088	Stop Start Broadcast
DM19	54016	Calibration Information
DM20	49664	Monitor Performance Ratio SAE J1939-73 Revised SEP2006
DM21	49408	Diagnostic Readiness 2
DM23	64949	Previously Active Emission Related Faults
DM24	64950	SPN Support
DM25	64951	Expanded Freeze Frame
DM26	64952	Diagnostic Readiness 3
DM28	64896	Permanent DTCs
DM29	40448	Regulated DTC Counts (Pending, Permanent, MIL-On, PMIL-On)
DM31	41728	DTC to Lamp Association
DM35	40704	Immediate Fault Status

Table 1: J1939 diagnostic messages supported in AUTOSAR 4.1

can be configured by software update. If the address allocation for the desired address fails, the ECU goes offline after sending the "CannotClaimAddress" message.
The J1939 Diagnostic Communication Manager of AUTOSAR 4.1 already supports many J1939 diagnostic messages **(Table 1)**. Not yet supported are OBD messages and messages for memory access, calibration and flashing. A great advantage of J1939 diagnostics in AUTOSAR 4.1 is the shared access with UDS diagnostics to the fault information stored in the AUTOSAR "Diagnostic Event Manager". This assures uniform management of the diagnostic information for both diagnostic protocols.

Figure 3: Dynamic address allocation in ISOBUS: ECU A changes its address, because the just connected ECU C occupies address 10

Gaps in the Integration of J1939 in AUTOSAR 4.1

AUTOSAR has made headway in implementing J1939 with Version 4.1, but there are still some gaps – particularly from the perspective of agricultural and forestry machinery. The most obvious omission of AUTOSAR concerns the support of self-configuring and command-configurable ECUs, which can change their addresses at runtime **(Figure 3)**. Furthermore, the following transport protocols from the J1939-based ISO 11783 (ISOBUS) standard are unknown to AUTOSAR:

> FastPacket from NMEA 2000
> ETP from ISO 11783-6

Implementation of application protocols for universal terminals, farm management and similar components would also be desirable.
Moreover, AUTOSAR still does not know how to handle the extended request-response protocol with Request2 messages and Transfer messages, NAME management and other diagnostic messages. Finally, it lacks a solution for simple access to the standardized signals and messages of J1939.

Outlook

Currently, AUTOSAR is not working on closing the described gaps. It appears, however, that increasing numbers of commercial vehicle manufacturers are becoming involved with AUTOSAR, especially in the agricultural and forest management industries. As a result, it is anticipated that the ISOBUS extensions will also be standardized in AUTOSAR over the long term.
Until then, tool and basic software manufacturers will come up with interim solutions. For example, Vector Informatik is currently developing support for fully dynamic address allocation and for convenient linking the application to the J1939 catalog of messages and signals. An implementation of the ETP and FastPacket transport protocols is also already planned.

Translation of a German publication in Hanser automotive, November 2013

Image rights: Vector Informatik GmbH

Links:
Vector: www.vector.com
MICROSAR basic software: www.vector.com/microsar

Dipl. Inf. Martin Schlodder
studied computer science at the University of Tübingen, and is working at Vector Informatik since 2004. He develops J1939-software and is a member in AUTOSAR working groups for the integration of J1939.
E-Mail: martin.schlodder@vector.com

ECU Development with AUTOSAR Basic Software in New Generation E/E Platform

Case Study PATAC

The Customer

The Pan Asia Technical Automotive Center Co., Ltd (hereinafter referred to as PATAC) is China's first Sino-foreign automotive development joint venture between GM and SAIC. And it is also the first automotive joint venture ever entitled "National Enterprise Technology Center" in China.

The Challenge

New functions and concepts are introduced into ECU development requirement.
In the new generation E/E platform, the following functions and concepts are introduced into the ECU development requirement: AUTOSAR Network Management with Partial Networking, CAN FD, Functional Safety and Secured OnBoard Communication (SecOC).
Since almost all the ECU suppliers have not implemented these features before, it increases a lot of risk on product quality and leadtime. It's not only ECU suppliers' responsibility but also PATAC's to find a proper solution to reduce the risk.

The Solution

AUTOSAR specifications already define the new features which are required by PATAC.
As AUTOSAR premium partner, Vector starts to implement AUTOSAR specifications since 2004. The mature AUTOSAR solution MICROSAR has been applied by many OEMs and ECU suppliers.

After one year evaluation, PATAC decided to introduce AUTOSAR basic software as basis of ECU development and select Vector as sole AUTOSAR basic software vendor.

The Advantages

PATAC and ECU suppliers apply the same AUTOSAR basic software – Vector MICROSAR for ECU development:

> All new features have been validated and optimized during evaluation phase under the cooperation of PATAC and Vector.
> Some ECU configuration parameters are pre-defined in the delivery package with the common understanding of PATAC and Vector. PATAC and its ECU suppliers benefit from shortened development times due to reduced integration and testing needs.
> Issue management can be coordinated by PATAC for all ECU suppliers.
> PATAC can easily extend or update the requirement within MICROSAR.
> Graphic visualization with expert view makes parameter configuration much easier.
> MICROSAR including bus controller drivers is available for many commonly used microcontrollers, PATAC and its ECU suppliers can select the microcontrollers according to the application requirement.

The AUTOSAR tool chain from Vector supports the entire development process.

AUTOSAR – Equipped for Everything?

Over the past ten years, AUTOSAR has decisively shaped and advanced the world of automotive electronics. Today, AUTOSAR is the foundation for efficiently implementing distributed functions and integrating software components from different sources. But the desired productivity gains are only realized if the entire development process is based on the new standard.

In the early era of microcomputer-based automotive electronics, a separate ECU was developed for each functional unit such as engine management or transmission control. Coordination with the other ECUs was implemented via control lines, e.g. lines carrying pulse-width modulated signals. Growing functionality, however, was driving the number of control lines and their costs upwards. That is why, in 1983, Bosch began to develop a network protocol – the Controller Area Network, or CAN – which was implemented in a production vehicle for the first time in 1990. It enabled the exchange of large amounts of data between ECUs in real time. This, in turn, unleashed the creativity of development engineers in implementing many new convenience functions such as "Keyless Go" and enabled significant improvements in the areas of safety, fuel economy and exhaust emissions.

However, the approach was still "one ECU for each function" – and each ECU was developed and produced by one supplier company. Whenever multiple ECUs from different manufacturers were involved in implementing a function, this required additional coordination effort and very extensive integration tests. Problems also occurred when an alternate supplier or new supplier was added for an ECU (e.g. for a new vehicle generation). This incurred additional coordination and testing effort.

To improve this situation, automotive OEMs and suppliers founded the AUTOSAR consortium (AUTomotive Open System ARchitecture) in 2003. Its standardization would make it possible to integrate algorithms, in unmodified form, into the ECUs of different manufacturers, and this would support the distribution of functions to multiple ECUs much better.

More than Basic Software

Ten years later, AUTOSAR is still a hot topic, and with good reason. Over 100 companies have developed specifications in a number of working groups. Meanwhile, the fourth main version has been published. Vehicles with AUTOSAR are in production, many automotive OEMs are currently developing vehicles with AUTOSAR, and nearly all of the others are

Figure 1: Hype cycle according to Gartner Inc.

showing excited interest and will hop aboard the train sooner or later. As soon as it appeared, enthusiasm was very high, and this led to some inflated expectations **(Figure 1)**. Over time, the desire for a simple and cost-effective solution had to be tempered due to the growing volume of specifications **(Figure 2)**.

AUTOSAR is often understood primarily as the basic software. Undoubtedly, this basic software is an important component. It contains an operating system, communication and management services, and it offers a uniform interface (Runtime Environment) for the actual application. Often automotive OEMs and suppliers begin their migration to AUTOSAR with the basic software – a reasonable step, because the specification of the basic software is well validated, and many producers offer implementations. In addition, this step lets ECU suppliers use a uniform software platform for multiple automotive OEM customers. Nonetheless, from the beginning AUTOSAR focused primarily on the overall development process. Therefore, the interfaces between participating partners were standardized and a methodology model was defined. The full benefits of AUTOSAR can only be achieved when all of its elements are implemented systematically.

New Division of Labor

While automotive OEMs previously created a text-based specification for each ECU and subcontracted its implementation to a supplier, AUTOSAR makes it possible to derive and share precise machine-readable specification requirements that come from a design of the total "vehicle electronics" system. Different suppliers can implement these requirements with assured processes and high efficiency. This enables a division of labor that does not require increased effort at the interfaces between the partners on the following levels:

> ECU hardware
> Basic software
> Application software
> Integration and comprehensive testing.

This division of labor has many advantages. Specialized companies can be contracted for specific sub-areas, and this leads to better solutions that are also more cost-effective. They range from ECU hardware suppliers to suppliers of basic software components, developers of individual driver assistance systems and suppliers of just certain parts of a complex assistance system. The automotive OEM can under-

Figure 2: Even AUTOSAR is not all that simple: growing content of the specification

Figure 3: Universal process: software development with AUTOSAR

take tasks relevant to its competitive advantage to achieve complex innovations. This avoids having competitors quickly benefit from the development via suppliers who would otherwise be responsible for creating the innovative content. In addition, clear structuring of the total system makes it possible to re-use individual components to a great extent. The introduction of this new methodology requires a realignment of the methodologies of all participants in the supplier chain. Consistent frontloading is necessary with AUTOSAR. The relevant (sub-) contractor must formulate its specifications much more precisely and formally. Compared to today's methodology, this requires considerable rethinking and a shift of responsibilities and tasks. This, in turn, results in an industry-wide change process, which will be painful for many participants. Currently, the industry is in the midst of a changeover. In the transition time, much more coordination effort is often required in the supplier chain with its significantly larger number of participants. This harbors risks. A chain is only a strong as its weakest link. Short-term change wishes and quickly performed error analyses can become a Herculean task with so many participants. In some places, projects in this phase give the impression that AUTOSAR does not deliver what it promises, and that it leads to additional costs. As in any change process **(Figure 1)**, it is necessary to cross the "trough of disillusionment". Therefore, the goal must be to complete changes "on the fly" as quickly and systematically as possible. Those vehicle developments that were developed with AUTOSAR methodology – based on the maturity levels of the participants – clearly demonstrate that this effort is worthwhile. The objective must be to reach the "plateau of productivity" as quickly as possible. A powerful tool chain can make a decisive contribution here.

Tools

Development tools had to be created in parallel to development of the AUTOSAR specification itself. As a result, the first tools were not always very mature. The higher the level of performance that was required of the tools, the greater their development effort, and the later their availability and maturity. Since increasing stability has been achieved in AUTOSAR specifications in recent years, and it has benefited from experience from many production projects, very high-performance tools are now available for productive use. "PREEvision" might be cited as an example here. PREEvision is a development platform for the entire E/E-product development process according to AUTOSAR **(Figure 3)**. It integrates the design, evaluation, optimization and documentation of E/E architectures in one tool. It exports and imports the interfaces definition in AUTOSAR format. To achieve comprehensive support of E/E development areas, other components are necessary; they range from requirements analysis to the provision of a collaboration platform and finally the management of variants **(Figure 4)**.

Each standard, including AUTOSAR, harbors the risk that it could put the brakes on innovations. The current example of "Ethernet in the vehicle" shows that this can be avoided by pragmatic approaches. The standard is being developed in parallel with first implementations. This promotes quick definition of a practice-based standard, which enables quick market introduction. The down side is that first implementations are based on a temporary level of the standard, so they will need to be modified later with additional effort.

Figure 4: One tool for everything: contents of the "PREEvision" platform

Not an End in itself

AUTOSAR is a consistent continuation and standardization of a development methodology that has evolved in automotive electronics over several decades. AUTOSAR relies on established and proven methods, and it utilizes the latest software technologies to enable the development of a new generation of vehicle functions. The progression towards more automated driving sets new requirements with regard to complexity and safety. Nonetheless, standardization must not be an end in itself. It is important to precisely observe the extent to which its advantages outweigh potential disadvantages and whether disruptive technologies might force out established players. There are signs of this already; just consider some new e-vehicle manufacturer or the autonomously driving car from Google.

Translation of a German publication in Mobility 2.0, March/2014

Image rights:
Vector Informatik GmbH

Links:
Webpage Vector: www.vector.com
Product Information PREEvision:
www.vector.com/preevision

Dr. Helmut Schelling
is a founder and managing director of Vector Informatik GmbH.

User-friendly Configuration of AUTOSAR ECUs with Specialized Software Tools

The simple CAN ECU is a thing of the past. Now, a typical ECU utilizes many functions of the AUTOSAR basic software to perform its complex tasks. However, the more functions there are, the more difficult and extensive the configuration process is too. Without tool support, developers would be lost.

Three factors affect the complexity of ECU configuration:

> More standardization
A large share of ECU software has now been standardized as basic software by means of AUTOSAR. However, the AUTOSAR principle "configuration instead of implementation" forces developers to create a consistent overall configuration from the start. There are no provisions for making corrections directly in the code.

> More functions
New microcontrollers with multiple cores and memory protection features and new networks such as Ethernet have increased the scope of the basic software, and therefore configuration needs.

> New cooperation models
The AUTOSAR methodology encourages new roles in the development process **(Figure 1)**. In addition to the supplier of the AUTOSAR basic software (TIER2-BSW), there are also suppliers of the application software components (TIER2-SWC) and hardware-related Microcontroller Abstraction Layer modules (TIER2-MCAL). This means that more coordination effort is required initially in the configuration process.

How do Software Tools Help here?

The AUTOSAR standard continues to develop in a highly dynamic manner. There are changes with each new release: the functionality of existing BSW modules is changed and new modules are defined. All of these modules come with their own configuration parameters. However, the AUTOSAR configuration approach is fundamentally able to handle this dynamism with its formally described module structures and parameters. This makes it possible to quickly and easily define the Basic Software Module Descriptions (BSWMD) contained in the delivered BSW. At first, this formal method sounds enticing as an approach for tools: the tool company would develop a generic tool with a reasonable amount of effort, and then all parameters could be individually configured – those parameters that are known today as well as all future parameters. However, no one would want to configure the thousands of necessary parameters in such an editor without further support.

Specific Editors and Assistance Functions

There are a number of approaches for developing a software tool that can simplify this work. A specially developed editor displays the relationships between parameters and simplifies the configuration process, e.g. by bulk operations. In addition, such an editor can group parameters thematically – even extending beyond module boundaries. The editor's graphic views make it easier to understand the complex configuration. Such tools are helpful and necessary in all domains of the basic software such as communication, mode management, diagnostics, memory management and I/O. When using an editor in the area of memory domains, for example, the developer can rather simply insert a memory block, which is then consistently configured in the non-volatile RAM manager (NvM) as well as in the flash EEPROM emulation (Fee). It is also very easy to estimate the overhead from the ratio of user data to block size in a graphic display **(Figure 2)**.

Assistance functions are also helpful in the configuration process. When the developer changes a parameter, for example, other parameters that depend on it are automatically corrected by a rules system. For more complex tasks, such as mapping runnable entities to operating system tasks, assistants may be of help: The developer can perform tasks like selecting runnable entities of the SWCs based on similar trigger conditions, assign a task and finally define the execution sequence within the task.

Another example from the mode management domain: The BswM module in AUTOSAR 4 permits completely user-defined configuration of arbitration rules, logical expressions and actions, for reacting to mode changes in other BSW modules or for requesting mode changes itself. An assistance function now helps to configure the BswM such that it behaves similarly to the relatively easy to handle ECU State Manager (EcuM) from AUTOSAR 3.

Deriving Parameters from the System Description

Configuration of the communication modules for CAN, LIN, FlexRay or Ethernet must match the system description that originates from the OEM. AUTOSAR provides for derivation of a basic configuration of the modules (the Base EcuC) from an ECU-specific extract of the system description (System Extract). In practice, however, the situation is somewhat more complicated **(Figure 3)**: AUTOSAR has defined a standard format for the system descriptions. But in addition to this format, OEMs also use the traditional DBC, LDF and FIBEX formats. Furthermore, the TIER1 supplier might use its own system descriptions, e.g. for private CAN buses within the ECU or for LIN buses that connect sensors with the ECU. The software tool must therefore identify all possible types of input data, convert traditional formats by suitable preprocessing methods and generate ECU-specific extracts. It must also be possible to merge multiple, separate system descriptions into a common description. Only then is it possible to generate the Base EcuC.

Similar considerations apply to the diagnostic modules. The configuration of the modules must match the ODX specification of the ECU. Therefore, the ODX file must also be included into the Base EcuC.

Sometimes the TIER1 also has a standard configuration for its projects that is available directly in EcuC format. These configuration elements should also be contained in the Base EcuC. The BSWMD files that the tool needs in order to check the configuration can originate from various sources. They are either provided by the TIER2-BSW supplier,

Figure 1: Roles in an AUTOSAR development project

Figure 2: Advanced views simplify the ECU configuring

or they might be supplied by the microcontroller manufacturer so that they match the MCAL modules.
Finally, the SWCs might be contained in the system extract which the OEM provides to its suppliers. According to the AUTOSAR 4 method, the system extract is first used to generate an ECU extract, which represents a flat perspective of the SWCs.
Afterwards, for example, the ECU extract is extended to include other SWCs which might be provided by external suppliers (TIER2-SWC).
Using the ECU Extracts and the Base EcuC, the ECU developer creates the overall configuration of the BSW and RTE (Runtime Environment). Here, the tool should treat those parameters taken from the Base EcuC as write-protected to avoid deviations from the system extract.

Project Update

Over the life of the project, the developer continually receives updated input data. Typically, these updates arrive at different points in time and with varying frequency. The OEM distributes new system descriptions according to the specific milestones of vehicle development, while the diagnostic description is typically updated much more frequently. Often, the time span between receiving the system description and the delivery date of the ECU is very short. A tool must therefore be able to update the configuration with the new input data as automatically as possible. A project update function can do this by generating a new Base EcuC, and by updating the actual configuration to the new Base EcuC.
But what happens when there are errors in the system description? Even when the problems have been clarified with the OEM, a corrected system description is usually not available immediately. In such cases, the ECU developer might want to intentionally ignore some of the derived parameters and overwrite them with other values. This is a way to correct the error directly in the ECU configuration. Since such a deviation is always critical, the ECU developer should be able to explicitly set the parameter status as "user-defined" in the tool. As long as the parameter is in this status, the tool may not overwrite its value when performing a project update. Only when the ECU developer removes the status "user-defined", the parameter adopts the derived value.
The tool also helps in coordination between the ECU developer and the OEM, e.g. by generating a report on overwritten parameters.

Figure 3: Workflow for creating and updating an ECU configuration based on the example of DaVinci Configurator Pro from Vector

Figure 4: Development of "vertical" ECU functions

"Merge" Function for Multiple Developers Working in Parallel

Even on smaller ECU projects, multiple developers are always working on the project simultaneously. If competencies are clearly defined (e.g. "Colleague xy is always responsible for the operating system."), then it is possible to avoid making parallel changes to the same module or to the same SWC. Typically, however, the developers are working in parallel, and they develop ECU functions that run vertically through all of the architecture layers of the ECU software **(Figure 4)**. In the times of manual C-coding, the integrator would then merge the different code branches on textual level. With AUTOSAR, however, this would mean that the integrator would have to merge XML files in AUTOSAR format that could be many megabytes in size. It would be practically impossible to compare or merge an AUTOSAR configuration using general XML tools. The structure of the file and the many references within the file are much too complicated for that.
Only an AUTOSAR- tool that clearly displays the differences in the AUTOSAR configuration and offers merge options

Figure 5: Diff/Merge support in DaVinci Configurator Pro

would be of help here **(Figure 5)**. Ideally, the user could view the differences directly in the same editors with which the user created the configuration. Then the user would feel "at home" and would not have to work with a separate tool with different, unfamiliar view of the data.
Nonetheless, the following rule of thumb applies: The fewer the changes that need to be merged the better. Massive changes to the configuration might result from a project update to a new communications matrix (C-matrix), for example. Therefore, the following practical procedure is recommended: Starting from an integration milestone MS0, the feature developers each do their development work on a separate branch **(Figure 6)**. The branches are merged, one after another, in the main branch – possibly in several intermediate steps (MS1 to MS3). Only then can the integrator update a project to a new C-Matrix (MS4).

Figure 6: Working on the same ECU configuration in parallel

Afterwards, new branches can be added and functions developed that match the new C-Matrix. All of this can be supported by a Configuration Management (CM) system, which also manages the implementation files of the SWCs.

Available Tools

The tool DaVinci Configurator Pro from Vector already enables the described working method. Soon, DaVinci Configurator Pro will also permit the creation of configuration variants, which can be dynamically switched in the ECU. They are also known as Post-Build Selectable Variants. Here too it is important to intelligently implement the available AUTOSAR concepts in tool functions. Then the developer gets further valuable support in the development of AUTOSAR projects.

Translation of a German publication in Elektronik automotive, Issue 4 / 2014

Image rights:
Vector Informatik GmbH

Links:
Vector AUTOSAR Solution: www.vector.com/autosar

Matthias Wernicke (Dipl.-Ing. (FH))
After completing his studies in Industrial Electronics at the University of Applied Science at Ulm, he was employed at the Daimler Research Institute in Ulm for four years. Since early 2000, he has been employed at Vector Informatik in Stuttgart working on the development of methods and tools for designing distributed electronic functions in motor vehicles. Today, he is responsible for product management of the DaVinci AUTOSAR tools.

VECTOR

AUTOSAR ECUs
Your Key to Successful Development

From Requirements to Production with MICROSAR

Be one step ahead with the Vector AUTOSAR solution MICROSAR.

> Basic software tuned specifically to your project for AUTOSAR Classic and Adaptive
> Efficient development work in all phases with an intelligent tool chain
> Powerful security functions to protect the ECU from unauthorized access
> Safety and efficiency with ISO 26262 up to ASIL D certified basic software and RTE
> Reach your goals faster with professional and tailored AUTOSAR services

More Information: **www.autosar-solutions.com**

Take advantage of the practice-proven Embedded Software MICROSAR for your production-ready ECUs.

Vector Informatik GmbH | Germany · Austria · Brazil · China · France · India · Italy · Japan · Korea · Sweden · UK · USA | www.vector.com

AUTOSAR goes Multi-core – the Safe Way

Two major topics are currently at the center of software development activities for automotive ECUs: First, the trend in computer architecture towards multicore processors and second the safety standards demanded by ISO 26262. Each of these topics is already complex enough in its own right, so what will be the consequences of the two together?

The main reason for introducing multi-core architectures is to increase the computing power without having to have a higher clock speed. This is however only possible if enough of the application software can be parallelized. The classic reference in this context is Amdahl's law. Taking a dual core processor and software with a parallelizability of 50 % as an example, this yields a maximum increase in power of only 30 % as compared to a single-core architecture.
To achieve the best possible computing power, developers have to make every effort to minimize inter-core resource sharing when distributing the software modules. The resources concerned are hardware registers and – in most cases – data areas. The challenge presented by cross-core resource utilization is not so much access coordination but rather the avoidance of wait-states in the case of concurrent access to the shared resources. In such situations there is a loss of independent data processing and parallelization is not as useful.

Functional Safety as per ISO 26262

Implementation of the functional safety requirements stipulated by ISO 26262 is now an integral part of the development process in automotive engineering. The requirements for the functions of an ECU are assessed with respect to their relevance to safety on the basis of a hazard and risk analysis and the appropriate ASIL (Automotive Safety Integrity Level) is assigned. What are the consequences for a multi-core software architecture if safety-related functions have to be implemented? Before answering this question there are a few points to note with regard to the lockstep concept for multi-core processors used in many safety projects.

Lockstep Mode

In lockstep mode two cores execute the same code. An independent comparator compares the results and generates a trap in the case of a discrepancy. The next step depends on the hardware and the safety concept of the ECU. The design of the hardware must ensure that it assumes a safe state following the occurrence of the trap. Apart from error handling, no multi-core software extensions are required as both cores execute the same code. In other words: Even though use is being made of several cores, this is not a multi-core architecture intended to increase computing power.

Figure 1: The developer groups the tasks together in OS applications and distributes them to the cores.

Multi-core Architecture with Distributed Software

Multi-core architecture can be used to increase computing power or – in the case of safety-related systems – for the implementation of diversity algorithms such as ASIL decompositions. The developer assigns the software modules to the OS applications defined in AUTOSAR on the basis of parallelizability and in accordance with the safety concept. This corresponds to partitioning as defined in ISO 26262 and refers to areas within an ECU which have to be able to run without causing any mutual interference (freedom from interference). In multi-core ECUs the OS applications are assigned to the various processor-cores **(Figure 1)**. From the point of view of the developer it is irrelevant whether the purpose of partitioning is parallelizability or decomposition: The task is to ensure freedom from interference between the OS applications. This essentially requires runtime monitoring and the avoidance of erroneous changes to memory content of relevance to safety.

Runtime Monitoring

In scalability class 2, AUTOSAR features runtime monitoring. This is however not sufficient for safety-related applications. The AUTOSAR operating system checks that no task requires an excessive runtime or blocks interrupts for too long. However, configuration of the correct, safe task sequence and task triggering presents developers with a high degree of complexity which is difficult to secure. An alternative solution is available from TTTech and Vector Informatik. This takes the form of a program flow monitor developed in accordance with ASIL-D which monitors the execution times and sequence of the tasks and the func-

Figure 2: Together with the check points in the application, flow monitoring ensures the correct sequence.

tions contained in these. An instance of this program flow monitor is created for each core. The monitor receives information on function execution through the check points incorporated into the software **(Figure 2)**. These check points are only passed through in the correct order and with the correct timing if the safety-related parts of the program are correctly processed.
The central watchdog manager gathers the status messages of all monitors and triggers the watchdog. It is not possible to manage each core separately, as the tasks are often interdependent across core boundaries. In addition, the AUTOSAR specification only permits the joint re-starting of all cores. The watchdog manager therefore has to be developed as a central module for the entire ECU. In contrast, the actual monitors run independently on each core and are capable of informing the watchdog manager of their status across core boundaries.

Memory Protection and Safe Communication in the ECU

Perfect freedom from interference is only possible if the hardware provides a Memory Protection Unit (MPU). This ensures that the application parts can only access pre-defined memory areas **(Figure 3)**. These memory areas are defined separately for each core but have to share the hardware resources (RAM, etc.) with the other cores. The operating system plays a central role: On each task change it has to re-program the MPU to effect a change of memory partition. This part of the operating system, which is responsible for context change, is a safety-related component and must always conform to the highest required safety level.
Protection against the erroneous overwriting of data is one aspect. The other is the need for the correct exchange of data between the tasks and across various processor cores. In the AUTOSAR architecture model this data exchange between two tasks takes place by way of the Virtual Function Bus (VFB). The VFB is implemented by the Runtime Environment (RTE).
The RTE has to satisfy the following additional demands for a safety-related ECU with multi-core architecture: It has to permit communication across memory partitions and at the same time be able to distinguish whether a communication path runs across core boundaries (inter-core) or between two tasks running on the same core (intra-core). Inter-core data transfer requires additional coordination mechanisms. For this purpose, the operating system provides the RTE with a function known as IOC (Inter-OS-Application Communicator). This allows the exchange of data between tasks and interrupt service routines on various cores.

Use of Standard Solutions

Ensuring functional safety as defined by ISO 26262 on multi-core ECUs would therefore appear to be no easy task. It does not how-ever involve re-inventing the wheel but simply using it correctly. Vector Informatik and TTTech can provide ECU developers with operating systems, RTE and program flow monitoring for multi-core architectures up to ASIL-D.
Generally speaking, multi-core systems are far more complex than single-core concepts. This need not however apply to configuration of the basic software: Once the software has been distributed to the processor cores, the remaining configuration work for a multi-core system is no more difficult than for a single-core architecture with the aid of optimum tool support. The developer groups tasks, interrupt service routines, etc. together in containers, the OS applications and then assigns these to the processor cores **(Figure 1)**. With the DaVinci Configurator Pro configuration tool from Vector Informatik this can be done with just a few clicks. The integrated RTE generator automatically takes care of configuring the correct communication methods (intra-core or inter-core) between the software modules.
In other words, a lot of support is already available for multi-core ECUs in the form of basic software and configuration tools for example. It is far more difficult to realize good tool support for creation of the software architecture and distribution of the software components to the processor cores. In fact it is usually restricted to the evaluation of variants. In the foreseeable future, ECU developers will still be called upon for their expertise and experience.

Figure 3: The memory protection unit ensures that the application parts can only access pre-defined memory areas.

Translation of a German publication in Elektronik automotive, June/2014

Image rights:
Vector Informatik GmbH

Links:
Website Vector: www.vector.com

Dr.-Ing. Helmut Brock
has been with Vector Informatik since 1999 where he works as Operating Systems Product Manager.
helmut.brock@vector.com

Dipl.-Ing. (FH) Joachim Kalmbach
has been with Vector Informatik since 2006 where he works as Product Manager in the Embedded Software sector. His main fields are AUTOSAR and Multi-core.
joachim.kalmbach@vector.com

AUTOSAR-compliant Development of an In-car Radio Product Using MICROSAR for the European Market

Alpine Electronics, Inc. ("Alpine"), manufacturer of car audio and navigation systems, developed an AUTOSAR-compliant in-car radio product for a European automotive manufacturer for series production in 2013. Alpine completed development of AUTOSAR related software components for CAN, diagnostics and other functions within one year starting from October 2010 and then also completed development of other software components. By using MICROSAR from Vector Japan Co., Ltd. ("Vector") as the AUTOSAR platform and developing a proprietary wrapper that leveraged existing software assets, Alpine succeeded in its product development while meeting a tight schedule and high quality demands.

AUTOSAR Enables Sharing of Automotive Embedded Software as Assets

Electronics becomes more widely used in automotive systems, such as power-train, safety functions, and infotainment. The transition from conventional mechanical and hydraulic systems to E/E architecture will never stop advancing in the future. While evolving electronics enrich the automotive functions, the number of man-hours required to develop the control software are higher than ever before, creating a significant burden on automobile manufacturers and suppliers. To deal with this situation, the open and standardized automotive software architecture AUTOSAR (AUTomotive Open System ARchitecture) **(see Figure 1)** was jointly developed by automobile manufacturers, suppliers, electronics manufacturers, semiconductor vendors, and software vendors, to aim enhancement of reuse of automotive embedded software as assets. AUTOSAR is a system that abstracts the differences in hardware, such as microcontrollers, and enables a standardization and reutilization of in-car application software assets, in short.

MICROSAR Selected at the Recommendation of European Automobile Manufacturer

AUTOSAR brings certain advantages on high efficiency on automotive software development. However, for manufacturers and suppliers using it for the first time, it's not really easy to read through all the specifications and understand the full picture. It is also challenging to understand the detailed development activities.

Alpine was among one of those car audio and navigation system companies struggling to go with AUTOSAR. Tomohiro Kusano (Alpine's OEM Product Development Dept.) described the challenges. "We were asked to build an in-car radio for a European automobile manufacturer, and they specified as part of the RFQ that it must be developed in compliance with AUTOSAR Release 3.0. We had never developed any product using AUTOSAR, so it was our first time." Also this particular in-car radio uses a Renesas SH2A microcontroller, with existing software assets based on µITRON **(see Figure 2)**. Alpine immediately began work on introducing AUTOSAR. For the development infrastructure, the company selected MICROSAR, the AUTOSAR-based embedded software products developed by Vector. Kusano explained. "We selected MICROSAR because Vector is one of the suppliers recommended by the European automobile manufacturer and because we had worked with Vector in the past on CAN bus-related development."

Next, Alpine consulted with Vector and gained an understanding of the functions realized by the basic software (BSW) of AUTOSAR. After analysis to extract minimum set of BSW regarding with required functionalities and possibility on combination of in-house solutions, Alpine contracted MICROSAR BSW modules. Alpine used MICROSAR SYS for its error reporting, communication management, and ECU state management, MICROSAR MEM for managing flash memory, and MICROSAR CAN and MICROSAR COM for its CAN control. They did not use any AUTOSAR runtime environment (RTE), because a large number of changes were expected in reuse of the existing software assets. Also, because the conventional µITRON-based operating system would be utilized, MICROSAR OS was not used. The entire configuration is shown in **Figure 3**. MCAL (Microcontroller Abstraction Layer), which abstracts the microcontroller hardware, was not supplied from the microcontroller vendor at the start of development, so it was developed at Alpine.

As mentioned above, Vector's MICROSAR was chosen as the BSW modules. The Complex Drivers, which is highly depending on the vehicle variants, was provided by the automobile manufacturer. The software components (SW-C), which are equivalent to the application layer, were developed by Alpine using existing assets based on µITRON.

Figure 1: AUTOSAR structure

Figure 2: In-car radio for European automobile manufacturer developed based on AUTOSAR

Figure 3: Software stack of developed in-car radio

Leveraging Existing Assets with a Proprietary Wrapper without Using RTE

During actual development, the schedule became an initial issue. Kusano recalled the process. "We began development in October of 2010, but we had to provide the client with a prototype based on the evaluation board by February 2011, so we had to have big progress with AUTOSAR on a very tight schedule."

As a solution, training on the development flow of AUTOSAR, a BSW functional overview, and configuration of the MICROSAR environment and BSW were held at Alpine by instructors from Vector to improve the skills of the development leads. At the same time, development of MCAL was immediately started. Next, Alpine worked with Vector to implement an integration of products, starting with the MICROSAR BSW modules, Alpine-developed MCAL, and Complex Driver and CAN database (*.dbc) files that the client provided, and also combining EcuC (ECU configuration description), GENy and DaVinci Configurator Pro (both Vector's configuration tools[1]) **(see Figure 4)**.

"The schedule leading up to the provision of the prototype was tight, and in the end, we went to Europe, where the automobile manufacturer is located, and worked together with Vector to accelerating integration and perform testing," explained Kusano.

These efforts were worthwhile, as the team successfully provided the prototype based on the evaluation board by February 2011, as requested by the client. They were also able to release a prototype which is based on the hardware for series production by April 2011. The AUTOSAR-related development work was nearly complete, and next, the weight shifted to development of the SW-C, which is the application layer. The SW-C, which is comprised of functions including the radio tuner and USB audio playback, follows the conventional architecture that interfaces with µITRON using the communication server named Virtual MOST and interfaces with the BSW modules using a specially developed wrapper without using the RTE **(see Figure 5)**. Kusano explained. "The number of man-hours required to develop the wrapper was about five man-months, but that figure is smaller than it would have been if using the SW-C with an RTE." After performing a series of tests, the team completed a bug-free version of the prototypes incorporating all functions in October 2011, and series production was started at the end of 2013.

[1] In new AUTOSAR 4.x projects GENy is no longer used for configuring AUTOSAR BSW. The current solution uses just one tool – DaVinci Configurator Pro.

Launching Product Development with AUTOSAR Release 4.0

Although increased man-hours is a typical concern with AUTOSAR-based development, Kusano revealed that there was no significant increase, with only two people (it became one at the final development stage) required for configuring the MICROSAR BSW while each did other development tasks in parallel, only three people (only in the initial development phase) required for integration work while they also did other tasks too, and only about five man-months required for wrapper development.

Kusano also revealed that he was quite satisfied with the quality of MICROSAR provided by Vector. "During the evaluation process, we discovered several problems, but they were mainly caused by incorrect settings, wrong use or problems caused by other tools. There was only one problem within MICROSAR itself. Moreover, all detailed inquiries to Vector Japan were handled within 24 hours, and we were very pleased with their technical support."

Figure 4: Flow of Integration Using MICROSAR

Figure 5: Development of a proprietary wrapper to replace the RTE and reduce the changes to software assets

Alpine has already started developing new products using AUTOSAR Release 4.0, with series production scheduled for 2016. "Introducing MICROSAR RTE that we skipped this time will be next challenge in our next product. We will work to develop new products with the continuous cooperation of Vector," Kusano said.

Now, as AUTOSAR has been getting more mandated in RFQs of European automobile manufacturers, Alpine's example of development with AUTOSAR by introducing MICROSAR for short turnaround and reduced man-hours will surely serve as an exemplary model for other suppliers

Reviewing the Project

Tomohiro Kusano
OEM Product Development Dept.
ALPINE ELECTRONICS, INC.

"Thanks to the high-quality MICROSAR product and Vector's support, we were able to meet the tight schedule and quality demands of our European automobile manufacturer client. AUTOSAR-compliant development will certainly become more important in the future, and we would like to leverage our experience and knowledge to other parts of the company and to new products."

Tsuyoshi Sakurai
Embedded Software Dept. (PES)
Vector Japan Co., Ltd.

"The expertise of the Alpine engineers also helped us out. We were very happy that our AUTOSAR solution could efficiently contribute to their development. We will continue to provide products and services that best meet the needs of our customers."

Image rights:
Cover and Figure 1: Vector Japan
Figure 2-5: Alpine Electronics, Inc.

Contact in Japan for further information:
Sales Division (Embedded Software), Vector Japan Co., Ltd.
[Tokyo] Tel: +81 3-5769-7808
E-mail: EmbeddedSales@jp.vector.com

Find your local contact person:
www.vector.com/contact

High-Rate Task Scheduling within AUTOSAR

Vector Addresses the Challenges of High-Rate Application Task Scheduling within an AUTOSAR System

The electrical and electronic (E&E) functions inside a modern vehicle have grown both in number and complexity; this new level of complexity has led automotive vehicle manufacturers and their suppliers to form the AUTOSAR partnership, to define a standardised yet feature-rich software architecture for within the vehicle Electronic Control Units. A common misconception is that high-rate application tasks cannot be scheduled within an AUTOSAR system. This article explains the mechanisms in place within the AUTOSAR operating system to handle the application scheduling requirements and how a successful configuration of the OS allows the software engineer to continue running the high-rate task schedules within an AUTOSAR system.

The Drivers for AUTOSAR

In recent years, the electrical and electronic (E&E) functions inside the vehicle have grown exponentially, both in number and complexity. This trend has been driven by a number of factors within the automotive industry, including an increased demand for E&E-based end-customer features, the advent and proliferation of ADAS systems and increasingly demanding legislation in diagnosing E&E system faults and Electronic Control Unit (ECU) failures.

The newest vehicle functions have reached such a level of complexity that traditional automotive ECU capabilities and supplier development methods can no longer fulfil the demands imposed by the modern vehicle E&E architecture. In 2003, this increasing level of complexity led automotive vehicle manufacturers and their suppliers to form the AUTomotive Open System ARchitecture (AUTOSAR) development partnership; where their mutual goal has been to define a common and standardised software architecture for use within the vehicle ECUs, that would ultimately provide the software features, mechanisms and support necessary to meet the higher demands of the vehicle's underlying E&E architecture.

Before AUTOSAR was defined, the primary focus of the vehicle manufacturers for an ECU's embedded software was often limited to the network communication stack and diagnostics. But with AUTOSAR, the underlying software within an ECU has undergone significant expansion; AUTOSAR "Basic Software" covers not only areas such as network communications for CAN, LIN, FlexRay, Ethernet and diagnostics protocols, but also now includes system and watchdog initialisation and state-handling, a non-volatile memory software stack for various memory devices, an abstraction driver layer for the microcontroller peripherals and also the operating system (OS).

Increased Processing Demand from the AUTOSAR "Sweet Shop"

The AUTOSAR standards are still growing and evolving to meet the very latest vehicle E&E requirements – at the end of 2014 the AUTOSAR Basic Software attained its highest ever level of complexity so far, when version 4.2 was released. Over 90 scalable embedded software modules are defined and are supplied along with powerful configuration tools and code generators in conjunction with a set of further standards for the workflows, methodologies and data exchange formats.

When an ECU supplier chooses to equip their ECU with AUTOSAR Basic Software, they find themselves being offered a new range of feature-rich embedded software modules that become available as off-the-shelf commodities. Lengthy software development lead-times are no longer a constraint in deciding whether to build further features into the ECU, since including an additional AUTOSAR module simply becomes a tick-box within the Basic Software configuration. Software architects can find themselves like "kids in a sweet shop" with over 90 software modules to pick and choose from.

Of course, each of these software modules comes at its own cost; not only in terms of a license fee that is used to recoup the development efforts of designing and creating each module, but also in terms of processor load. Each software module will take up precious RAM and ROM resources within the microcontroller and also has functions that need to be scheduled and called, that consume valuable processor runtime overhead.

The AUTOSAR Operating System

In particular, the AUTOSAR OS offers many more features that were currently unavailable or not even considered by the ECU supplier. In the past, a simple single-tick or super-loop scheduler may have been adequate to host all the ECU application tasks. Now, with the alternative inclusion of an AUTOSAR OS, a new set of operating system features becomes immediately available to the software engineers.

The AUTOSAR OS is offered in a range of "Scalability Classes", extending from SC1 to SC4:

> AUTOSAR OS Scalability Class 1 extends the OSEK/VDX standard operating with the addition of schedule tables
> OS Scalability Class 2 extends AUTOSAR OS SC1 with the addition of timing protection
> OS Scalability Class 3 extends AUTOSAR OS SC1 with the addition of memory protection
> OS Scalability Class 4 extends AUTOSAR OS SC1 with the addition of both timing and memory protection

The additional features within the AUTOSAR OS do not come without a resource penalty and, like every other module within the AUTOSAR Basic Software, the increased functionality requires RAM and ROM resources within the microcontroller and consumes additional processor runtime overhead. These additional requirements would generally need to be supported with higher ECU processing power and greater microcontroller memory capacity.

A common misconception has subsequently arisen, that adopting an AUTOSAR software architecture will automatically require the ECU supplier to upgrade the internal microcontroller from a 16-bit to a 32-bit processor. Of course, this should not be the natural conclusion, since replacing an existing OSEK OS with an AUTOSAR OS SC1 would incur only a minimal additional overhead, and only if the schedule tables feature is utilised.

Figure 1: AUTOSAR OS Scalability Classes

However, the software engineer now has the option to upgrade the operating system to include timing and/or memory protection features, with just a simple selection within the required Basic Software configuration. The lead-time associated to develop these features no longer has to be a hindrance, or even a consideration, since the OS is available as a commercial off-the-shelf (COTS) component. If all the AUTOSAR Basic Software modules are handled in a similar manner, the inclusion of all the new AUTOSAR features can become technically trivial, however a significant and visible impact to the processor load and memory requirements would immediately be observed.

Scheduling a High-Rate Application Task

Just like the misconception that an AUTOSAR software architecture inherently requires a 32-bit processor, a similar misunderstanding occurs that high-rate application tasks cannot be scheduled within an AUTOSAR system. Many typical automotive applications, especially within the powertrain domain, require tasks that can run at rates of every 100 microseconds (10 kHz) or even faster. Such a challenging scheduling requirement puts a very high demand on the AUTOSAR OS, and in turn on the microcontroller processing load.

For these purposes, the AUTOSAR OS provides two types of interrupt "category":

> Cat1 interrupts are "unsupported" by the AUTOSAR OS. This means that the code to get safely into and out of the interrupt handler is not generated by the AUTOSAR OS and has to be generated in another way. [1]

> Cat2 interrupts are supported by the OS. This means that the OS's code generator and libraries abstract the interrupt from the hardware in a portable way. [1]

Cat2 interrupts are normally used to schedule all the tasks of the Basic Software and application level software components, where the AUTOSAR modules are specified to use the AUTOSAR OS and API (application programming interface). Where the application has a specific high-rate task scheduling requirement, the software engineer can make use of the AUTOSAR Cat1 interrupt, and hook the application task directly into the microcontroller's interrupt table using the configuration tool. The Cat1 interrupts are essentially "invisible" to the rest of the AUTOSAR system.

Cat1 interrupts offer a powerful way to interact with the hardware directly, and therefore a degree of care and diligence must be exercised when utilising this type of interrupt; the main advantage is that the latency of the interrupt is very low, and hence the required high-rate task scheduling can be achieved. However the normal protections offered by the AUTOSAR OS are not present, and therefore the interactions from within the Cat1 interrupt with the rest of the system have to be carefully considered prior to implementation.

Another aspect that has to be thought out is the allocation of the interrupt priorities across the system. The AUTOSAR OS is a fully pre-emptive operating system and therefore the AUTOSAR application tasks may interrupt each other. But it would be unacceptable for an AUTOSAR task to be allowed to interrupt, or block, the high-rate scheduled task and therefore the priorities need to be allocated such that the high-rate task has the highest priority in the system. This means that the high-rate task can interrupt any other task in the system, including the AUTOSAR OS, and every other interrupt in the AUTOSAR system must be at least one priority level below that of the high-rate task, this is termed the "system interrupt level".

Figure 2: Example High-Rate Task Scheduling within an AUTOSAR System

Conclusions

The AUTOSAR standards have reached a new height, but will continue to grow and evolve to meet the today's latest vehicle E&E requirements. For ECU software developers, embracing AUTOSAR in addition to enhancing their application features is often considered to be just one further burden that is too often undertaken only if mandated by the vehicle manufacturer. The proliferation of AUTOSAR architectures into powertrain areas has been particularly slower than in other domains (such as body and chassis applications) with one reason being the concern of handling the high-rate task scheduling demands in addition to all the new AUTOSAR features.

However, the mechanisms to handle such demands have already been considered and addressed by the AUTOSAR standards and many AUTOSAR applications are successfully running high-rate task schedules using the OS Cat1 interrupts. By analysing the processor load within such systems, Vector have observed that in many cases the high-rate task scheduling can still be achieved whilst also running the feature-rich AUTOSAR Basic Software.

But inevitably the additional features that are provided within the AUTOSAR Basic Software and AUTOSAR OS will bring further demands in load onto the processor, and therefore new differentiators have emerged within the embedded software market. The scalability, configurability, efficiency and optimisation of each new AUTOSAR module has to be considered when sourcing AUTOSAR Basic Software as off-the-shelf commodity parts.

Adopting AUTOSAR continues to be a concern for many ECU suppliers, often heightened by the perception of the complexity it can bring. But the complexity is inherently present within the system already, driven by the end-customer's feature requirements and the underlying E&E architecture necessary to support those required features. AUTOSAR only provides the mechanisms and structures necessary to manage the already existing complexity; therefore an AUTOSAR system should not be viewed as a problem – but rather, that it provides the solution.

Image rights:
Vector Informatik GmbH / Vector GB Ltd.

Literature:
[1] Explanation of Interrupt Handling in AUTOSAR, sourced from www.autosar.org on 09/07/2015: www.autosar.org/fileadmin/files/releases/3-2/software-architecture/general/auxiliary/AUTOSAR_InterruptHandling_Explanation.pdf

Links:
Website Vector: www.vector.com
Vector AUTOSAR Solution: www.vector.com/autosar/

Steve Waldron, Vector GB Ltd.
Local Product Line Manager for the Embedded Software Product Line

Tom Dalek, Vector GB Ltd.
Software Engineer for Embedded Software

Alex Ghinet, Vector GB Ltd.
Senior Software Engineer for Embedded Software

New Opportunities with AUTOSAR

AUTOSAR has been gradually adapted for use in heavy duty vehicles since 2009 and is already being used in the production of some truck model series. AUTOSAR is currently being introduced in agricultural machines, and with a few extensions of AUTOSAR this could become a success story as well.

For several years now, after the implementation of AUTOSAR by truck producers, manufacturers of agricultural and construction machines have also expressed interest in AUTOSAR. Compared to the automotive and truck industries, they build even more vehicle variants with even lower production volumes. Therefore, the problem of rising costs with the increase of electronics in new vehicles is particularly difficult in this area. To remain competitive, it is crucial for agricultural and construction machine manufacturers to re-use entire ECUs and individual functions implemented in software as often as possible. AUTOSAR was developed precisely for such requirements, and therefore it imposes itself as the natural choice for this domain.

Conceptual Advantages of AUTOSAR

The software component model of AUTOSAR promotes re-use of individual functions. To re-use entire ECUs, an sophisticated variants handling method and the option of reconfiguration (post build) are available. However, the greatest advantages are based on systematic use of AUTOSAR methodology. Converting development processes over to this methodology opens up the opportunity for rethinking the entire ECU architecture and therefore optimizing it for modular usage of all of its components – hardware and software. This makes the AUTOSAR methodology an important driver of innovation.

The Architecture of AUTOSAR

There were two primary motivations for developing the AUTOSAR standard. The first goal was to standardize the hardware and protocol drivers, which was achieved by specifying the basic software modules (BSW modules). One benefit of the basic software is to save costs by using standard software. Second, it makes it easier to exchange an ECU's hardware.

The next goal of AUTOSAR was to break down complex functions into manageable units. This was achieved by implementing software components (SWCs) that communicate via an abstract channel known as the virtual functional bus (VFB). The actual communication relationships are determined by the distribution of components to ECUs. The entire communication is implemented at runtime by the RTE (Runtime Environment), which accesses the basic software for external communication. **Figure 1** shows some of the basic software that is relevant to J1939 applications as well as the RTE and SWCs of the application layer.

Figure 1: Excerpt from an AUTOSAR architecture extended for agricultural machines

Software Component Model of AUTOSAR

The software components (SWCs) of AUTOSAR are embedded in a hierarchy of compositions. The SWCs functionality is implemented by runnables. SWCs define their communication needs by interfaces (Ports) which group operations and transferred data elements . The virtual functional bus combines these ports on an abstract level. After distributing the software components to ECUs, the abstract signals of the virtual functional bus are mapped to specific network signals, PDUs, and frames, and are thereby assigned to a physical network.

Use of AUTOSAR in Heavy Duty Vehicles

The first trucks to feature an AUTOSAR-based EE architecture have already been on the roads for several years. Individual ECUs from other branches of industry are also already based on this standard, e.g. engine controllers for diesel engines. The first agricultural machine producers are currently adapting their processes to AUTOSAR and have already developed initial pre-production ECUs. Considerable efficiency gains have been achieved in implementation by systematic migration to AUTOSAR methodology.

The use of AUTOSAR in the heavy duty vehicle industry was enabled by a step-wise extension of the basic software to include J1939 protocols **(Figure 2)**. In 2009, the cornerstone was laid for this effort when the J1939 Transport Layer (J1939Tp) was specified in AUTOSAR 4.0.1. Four years later, AUTOSAR 4.1.1 introduced meta data, and thereby the ability to access address information in the CAN IDs from higher layers of the basic software – which is important for J1939. Meta data makes it possible to handle or route a message regardless of its sender, which is necessary for correctly responding to a request. The J1939 Request Manager (J1939Rm) was introduced as an additional BSW module to perform this task. The J1939 Network Management (J1939Nm) was added for address arbitration, as was the J1939 Diagnostic Communication Manager (J1939Dcm) which handles J1939 diagnostics together with the extended Diagnostic Event Manager (DEM).

Currently, AUTOSAR is encountering its limits in fully dynamic address allocation **(Figure 3)**, which is widely used in agricultural and construction machines and is a prerequisite for communication on the ISOBUS. Also to be improved is the ability to handle several multiplexors in proprietary

Figure 2: J1939 milestones in AUTOSAR

Figure 3: Dynamic address allocation on the ISOBUS

messages (PropA, PropA2, PropB) and the transport protocols of NMEA2000 and ISO 11783. There is also no support yet for the protocols of ISO 11783 components such as the Virtual Terminal and Task Controller.

Further Development of AUTOSAR

Today, there are already solutions available for fully dynamic address allocation and for the extended transport protocol of ISO 11783 as well as for handling proprietary messages. In these solutions, the CAN Interface (CanIf), J1939Nm, J1939Tp and J1939Rm modules are extended accordingly. However, these AUTOSAR extensions are only partially coordinated between different suppliers. Standardization on the level of the basic software would be desirable here. Implementations also exist for the protocols of ISO 11783 components today. Standardization of these protocols in AUTOSAR could be achieved by "application interface" documents, which just define the interfaces to the application. They would then be implemented by complex drivers (CDD). Several manufacturers of heavy duty vehicles such as Caterpillar, CNHi and John Deere have recently become members of AUTOSAR, and this increases the chances that dynamic addressing and the lacking protocols will be added to the standard over time. Then there would be no remaining obstacles to implementing AUTOSAR in the area of heavy duty agricultural vehicles and construction machines.

Key Properties of ISO 11783 / ISOBUS

ISO 11783 was developed with the goals of combining any desired tractors with any implements over the ISOBUS and of promoting new developments in agricultural technology such as "precision farming." The standard is based on SAE J1939 with fully dynamic addressing, and it adds specialized components as well as satellite navigation from NMEA 2000. The most important components are:

- Virtual Terminal: for monitoring and manually controlling implements
- Task Controller: for automatically controlling implements based on position data.
- Tractor Controller: for exchanging data between the vehicle network and implement network.
- Sequence Controller: for logging and repeating identical sequences such as turning at the end of the field.
- File Server: for saving the data of other ECUs upon request.

Compared to J1939, ISO 11783 uses two additional transport protocols: extended TP for large amounts of data and FastPacket for navigation data. It also restricts diagnostics to the active error and last active error (DM1-DM3).

Translation of a German publication in Hanser automotive, November 2015

Image rights:
Vector Informatik GmbH

Links:
Vector: www.vector.com
MICROSAR basic software: www.vector.com/microsar

Dipl. Inf. Martin Schlodder
studied computer science at the University of Tübingen and has been employed at Vector Informatik since 2004. He develops J1939 software and is a member of AUTOSAR working groups for integration of J1939.
E-Mail: martin.schlodder@vector.com

VECTOR

Accelerate Your Embedded Development!

Standardized Basic Software for CAN, LIN, FlexRay, Ethernet, J1939

Take advantage of scalable software modules which are being used successfully worldwide.

- Benefit from the Vector AUTOSAR solution with smart extensions
- Combine software components individually and specifically for your projects
- Save on development time by using a perfectly tuned tool chain
- Update your ECUs securely with the Flash Bootloader on a high level of process reliability
- Boost your efficiency with our customized consulting and development services

Find out more: **www.ecu-software.com**

Vector supports you in your development work with ECU software, configuration tools and services.

Vector Informatik GmbH | Germany · Austria · Brazil · China · France · India · Italy · Japan · Korea · Sweden · UK · USA | www.vector.com

VECTOR

SilentBSW – Silent AUTOSAR Basic Software for Safety-related ECUs

Coexistence of Safety-related and Non-safety-related Software in one ECU by Protecting Memory Areas

In future vehicles, most ECUs will be "Mixed-ASIL systems" that contain both safety-related functions and functions without safety relevance. This article presents strategies for efficiently implementing these functions based on AUTOSAR. One solution is SilentBSW – an AUTOSAR basic software that monitors itself and blocks prohibited write accesses to safety-related software.

AUTOSAR and ISO 26262

AUTOSAR addressed the topic of "functional safety" early on. Current versions of the AUTOSAR specification [1] contain a number of concepts that support the development of safety-related ECUs. Examples are the end-to-end protection for exchanging information and the program flow monitoring with a watchdog. In defining these concepts, steps were taken to support ECU developments in accordance with ISO 26262 [2].

Software Development for Safety-related ECUs

AUTOSAR supports the distribution of functions to multiple ECUs. This simplifies functional development, which can be conducted without regard to the mapping of function units on specific ECUs. One consequence, however, is that a disproportionately high number of ECUs are then classified as needing to fulfill safety requirements **(Figure 1)**. These ECUs would have to be fully developed according to the highest ASIL (Automotive Safety Integrity Level) of their requirements, even if the vast majority of its functions might not be safety-related. Just a single ASIL-D safety requirement would mean that the entire ECU would have to be developed in accordance with ASIL-D.

To avoid this "ASIL lift-up effect" [3], ISO 26262 allows the coexistence of sub-elements with different ASIL levels. This coexistence requires verification that there are no undesirable interactions between sub-elements. Partitioning is a proven method for implementing this "freedom from interference" for software. The following aspects must be considered in this context:

1. Memory accesses
2. Timing behavior and execution order
3. Exchange of information

"Standard Case" of Coexistence

ECUs containing both safety-related and non-safety-related functions are known as Mixed-ASIL systems. For purposes of simplification, in this article any software with a low ASIL is referred to as QM software, and software with a higher ASIL is referred to as ASIL software. A prerequisite for the use of ASIL software is that the hardware design must be adequate for the maximum required ASIL.

The "standard case" category includes systems with moderate ASIL content, where the interaction between the different contents is rather low. In these cases, separating the partitions is implemented very efficiently by separate elements. Barriers are set up between the partitioned contents; they prevent – or at least detect – propagation of errors with sufficient probability and report them to the ASIL software. This means that only the barriers and safety-related software need to be developed to the relevant ASIL **(Figure 2)**. In MICROSAR Safe – basic software (BSW) developed jointly by Vector Informatik GmbH and TTTech Automotive GmbH – these protective mechanisms are available up to ASIL-D.

Protection against Memory Violations

The Memory Protection Unit (MPU) of the microcontroller is used to limit write access to the memory areas of its own partition. The software that drives the MPU and executes the context switch must support the highest assigned ASIL. This software is part of an AUTOSAR operating system per Scalability Class 3 or 4 and must be developed to ASIL-D. This function is part of MICROSAR Safe and known as SafeContext.

Protection against Violations of Time Behavior and Execution Sequence

Errors in the QM software must not interfere with proper timing behavior of the tasks of the ASIL software. This is achieved by monitoring the defined execution sequence and time intervals between different functions of the ASIL software. AUTOSAR 4.0 defines functions for this purpose. In conjunction with a safe hardware watchdog, the software detects any violation of requirements and changes the system to a safe state. This does not prevent errors, but their occurrence is detected. This approach has proven to be effective within typically allowable error reaction times. MICROSAR Safe includes this functionality under the name SafeWatchdog.

Figure 1: Example of a Mixed-ASIL system with function mapping to multiple ECUs

Protection against Communication Errors

ISO 26262 lists relevant error patterns in the exchange of information, and AUTOSAR has defined algorithms for detecting them by an end-to-end protection. As in program flow, the goal is to reliably detect each occurring error and change the system to a safe state. Monitoring is performed by inserting sequence numbers and a checksum.

To avoid having to assume a safe RTE (Runtime Environment), the E2E Protection Wrapper and E2E-Lib modules perform monitoring above the RTE in a safe partition of the application software. In MICROSAR Safe, these two modules are contained in option SafeCOM.

Coexistence with a Low Portion of Safety-related Software

If the portion of ASIL software is low, it may make sense to have the ASIL software protect itself against undesirable memory changes. The undesirable modification might be detected, or even corrected, by the following measures **(Figure 3)**:

> Redundant and possibly diversified computation – including validation of the ASIL software's algorithms
> Redundant and possibly diversified storage of data
> Deactivation of interrupts in critical areas, e.g. in areas in which comparisons and decisions are made.

If it is only possible to detect errors but not prevent them, the ASIL software must take appropriate measures to put the system in a safe state. Self-protection against memory changes is implemented in the ASIL software for the specific project. The methods described above are adopted for program flow and communication, because no self-protection is possible in these areas.

Figure 2: Barriers prevent errors from being propagated to the ASIL software

Coexistence with Strong Interaction

The method of protecting against memory violations based on an MPU, which is described above, is a very powerful mechanism for implementing any combination of partitions of different ASILs on an ECU. However, the required context switching between the partitions results in runtime overhead that can become critical. This could happen, for example, if a large number of signals are exchanged synchronously between the ASIL application software and the QM basic software, and it is not possible to group the signals.

In such cases, it is advantageous to have the QM basic software run in the same partition as the ASIL application software. But that is only possible if the basic software was developed for the ASIL of the application software. The development effort required for this is often not suitable, because the requirements of the actual functionality of the basic software are typically not classified as safety-related. This applies to sending out signals, for example, because an end-to-end validation of the communication is still always necessary to detect errors in bus transmission.

Therefore, one highly efficient approach is to leave the basic software on the QM level and enable coexistence with the ASIL software without the use of an MPU. MICROSAR Safe offers this type of solution under the name SilentBSW – which is a non-interfering basic software.

To prevent undesired overwriting of memory cells of the ASIL software by the QM software **(Figure 4)**, it is verified for each write command of the QM software that the write access is performed within a valid memory range. One difficulty here lies in a unique property of the basic software: Parts of the program code are generated, and it is not feasible to develop configuration and generation tools per ISO 26262.

The solution is to check the entire code after generation. This is not performed by tests, rather by a specially developed code checker. This validation covers the vast majority of error patterns. The remaining error patterns are covered by code inspections or runtime checks.

Summary and Conclusion

Approaches are already available today for developing ECU software per AUTOSAR and ISO 26262 up to ASIL-D. They include methods for the especially relevant Mixed-ASIL systems, as well as the new SilentBSW approach from Vector, which significantly reduces the execution time requirements for some application cases.

It is not necessary to assume that every requirement of the basic software must be generally classified as safety-related. Rather, it is much more important to implement individual requirements – which are derived from the safety concept for an ECU – in the basic software at the specific ASIL that is needed.

Figure 3: The ASIL software protects itself from errors

Figure 4: The QM software itself blocks unallowable write accesses to the memory of the ASIL software

The article is based on a speech of Günther Heling at the VDI conference Baden-Baden-Spezial, VDI report 2172

References:

[1] AUTOSAR GbR: http://www.autosar.org
[2] International Organization for Standardization (ISO), ISO 26262-1 bis ISO 26262-9, Road vehicles – Functional safety ISO 26262-1:2011(E) up to ISO 26262-9:2011(E), First Edition 2011-11-15
[3] Dr. Poledna, S et.al.: Ein ASIL D Plattformsteuergerät für eine elektrische Hinterachse mit Fahrdynamikfunktionalität; VDI-Bericht Nr. 2132, 2011 (English translation of the speech title: An ASIL D platform ECU for an electric rear axle with driving dynamics functionality; VDI report No. 2132, 2011)

Dr.-Ing. Günther Heling, Vector
is Global Product Line Manager for Embedded Software at Vector Informatik GmbH in Stuttgart.

Dipl.-Ing.(FH) Jochen Rein
is Manager Product Management and Sales for Embedded Software at Vector Informatik GmbH in Stuttgart.

Dipl.-Ing.(FH) Patrick Markl, Vector
is Team Leader Concept Development Embedded Software at Vector Informatik GmbH in Stuttgart.

Practical Implementation of Mixed-ASIL Systems

A Certified Operating System Simplifies the Development of Safety-related Software

The ISO 26262 standard describes a recognized and standardized process for developing safety-related ECUs in the automotive field. However, only parts of the software in these ECUs are safety-related. The goal is to restrict the additional, intensive development efforts for these safety-related components. It is possible to set up an ISO-conformant mixed-ASIL system, which may contain both ASIL functions and functions without qualification, using an advanced AUTOSAR operating system and two other basic software modules.

The ISO 26262 standard is increasingly being used to develop safety-related ECU software. At the beginning of a standard-conformant development process, the developer performs a hazard analysis and risk assessment of the system under development. The developer establishes safety goals and assigns each of them a specific Automotive Safety Integrity Level (ASIL), ranging from A to D, based on the probability the error will occur, the severity of damage that could result and the ability of the driver to control the vehicle in case of defect.
Assignment of ASILs to all software elements generally results in functional groups with different ASIL classifications. In principle an ECU's entire software must be developed to the highest ASIL determined for one of these functional groups. This intensifies the development effort to an extreme, because even non-safety related software must be developed to the high requirements of the safety process.

The approach of using mixed-ASIL systems provides a solution. In this case, the functional groups are isolated from one another by suitable protection measures, so that they cannot interfere with one another. This reduces the development effort for the individual functional groups to what is required for their specific ASILs.
The protection mechanisms needed are available in the form of a modern AUTOSAR operating system and two other basic software modules which the ECU manufacturer does not need to develop separately.

Technical Safety Concept

Developers must ensure that the overall system fulfills their safety requirements. None of the system's software components may put fulfillment of these safety requirements at risk. Therefore, the only safety requirement for software components without safety-related functionality is that they must conform to the principle of freedom from interference ([1] Part 9, section 6.4).
Freedom from interference of software components is defined by three properties:

> Safe memory accesses
> Correct time execution
> Safe data exchange

A component's freedom from interference can be verified by classic verification measures, e.g. by code reviews. There are also approaches in which a specially developed code checker is used to check the freedom from interference of the basic software [2]. Other measures may be taken in the software to ensure protection against hardware-related disturbances as well.
A modern AUTOSAR operating system like MICROSAR OS SafeContext **(Figure 1)** offers protection against faulty overwriting of memory contents. The protection is achieved by partitioning each functional group into a so-called OS application. Each OS application's data are allocated in separate memory partitions. Along with the application data, context-related data such as stacks and the contents of important registers are also located in such a memory partition. Access to these memory partitions is prevented by a Memory Protection Unit (MPU), which is part of the microprocessor hardware.
When switching the running task or the Interrupt Service Routine, the operating system executes a context switch. Here, the context data is stored, and the MPU is reconfigured so that it only enables the memory partition for the task or Interrupt Service Routine that is active after the switch **(Figure 2)**. This switch is only executed by the operating system and is safety-related. Therefore, the AUTOSAR operating system MICROSAR OS SafeContext was assigned ASIL D classification and was developed in processes defined for ASIL-D in ISO26262.

Comprehensive Validation Concept

It is important to monitor the program flow in safety-related systems. The Watchdog Manager specified in AUTOSAR **(Figure 1)** is used for this purpose. This module, which supplements MICROSAR OS SafeContext, is available in the form of the SafeWatchdog [3] developed by TTTech Automotive GmbH, which is qualified to ASIL D. As the name suggests, this component controls the hardware watchdog, and it safely ensures a reset of the ECU in case of error. In addition, this component monitors for correct time flow of the application's tasks. Developers can set a number of parameters for the monitoring such as program flow, cycle times, minimum/maximum execution times, etc.
The third requirement for freedom from interference, which is implementation of reliable communication, is fulfilled by end-to-end protection **(Figure 1)**. With the help of the E2ELib [4] specified in AUTOSAR, the SafeCOM product protects the data to be transmitted using a CRC and sequential message numbers. Strictly speaking, this does not ensure safe communication, rather just "integrity" in communication. Software cannot protect against data failure due to hardware errors, e.g. a bus line break. To ensure safe communication, additional measures must be taken in the hardware, e.g. in the form of redundant buses.

Figure 1: Layout of protective mechanisms in the AUTOSAR architecture

Integration of Application Software and Operating System

In the ISO standard, safety-related components developed by an external supplier and supplied to the ECU manufacturer are referred to as "Safety Elements out of Context" (SEooC). They include the operating system discussed above as well as the watchdog manager and the E2ELib. During development, the suppliers of such components must make assumptions about the expected safety goals without familiarity with the ECU project. Therefore, as part of integration work ECU developers must check whether the safety goals assumed for the supplied SEooC are sufficient for achieving the safety goals of their projects. Moreover, the ECU developer is responsible for following special integration instructions for the supplied software module. Therefore, each SEooC is supplied together with a safety manual, which contains the integration instructions and assumptions about safety goals.

At first, this may sound like more work. However, upon closer examination, it becomes clear that a similar level of effort must be planned for integrating components that are developed in-house, but the advantage of supplied components is that the effort required for creating these components is eliminated. Overall, this yields significant work savings.

Outlook

The TÜV Nord organization has certified the MICROSAR OS SafeContext operating system developed by Vector for the TMS570 microcontroller, making it the first AUTOSAR operating system to be certified to ASIL D. This implementation is currently being transferred to other platforms. They include multi-core processors which are being used increasingly.

MICROSAR OS SafeContext, used together with the SafeWatchdog and SafeCOM basic software modules, provides an up-to-date and safe development foundation for safety-related ECUs. In particular, it can be used to cost-effectively implement mixed-ASIL systems.

Besides protecting the application software, the safety process must also protect all of the basic software. Vector offers a variant developed in conformance to ISO 26262 which is distinguished by its ability to achieve the safety goal "freedom from interference in relation to memory access".

Translation of a German publication in the special edition "Funktionale Sicherheit" of Elektronik automotive, July/2013

Image rights:
Vector Informatik GmbH

References:

[1] ISO 26262 - Road Vehicles - Functional Safety, 2011
[2] G. Heling, J. Rein and P. Markl, "Koexistenz von sicherer und nicht-sicherer Software auf einemSteuergerät" ("Coexistence of safety and non-safety software in one ECU"), ATZ special electronics issue of electronica, pp. 62-65, November 2012
[3] AUTOSAR, "Specification of Watchdog Manager" V2.3.0
[4] AUTOSAR, "Specification of SW-C End-to-End Communication Protection Library" V3.0.0

Links:

Website Vector: www.vector.com
Product Information MICROSAR Safe: www.vector.com/vi_microsar_safe_26262_en.html

Figure 2: The MPU in the hardware protects the memory partitions from unauthorized access. Reconfiguration of the MPU is performed by the operating system.

Steffen Keul (Dipl. Computer Science)
is Product Management Engineer for Functional Safety at Vector Informatik GmbH

Dr. Helmut Brock (Doctor of Engineering)
is Product Manager Operating Systems at Vector Informatik GmbH

SEAMLESS IMPLEMENTATION OF ECU SOFTWARE BASED ON ISO 26262

Growing use of the ISO 26262 standard is producing clearly defined requirements for the development and validation of E/E systems. Vector describes a seamless methodology and tool environment for considering ISO 26262 in the development of ECU software that includes the Autosar platform. In particular, it is shown how references to safety goals and requirements are represented throughout the entire development. The Autosar basic software also supports Mixed-ASIL systems, which avoids the need to cost-intensively upgrade system functions with lower ASIL to a higher ASIL. In sum, this leads to clear cost advantages in developing safety-critical systems compared to not integrated method and tool environments.

AUTHORS

DIPL.-INF. STEFFEN KEUL
is Product Management Engineer for Autosar at Vector Informatik GmbH in Stuttgart (Germany).

DR. EDUARD METZKER
is Senior Product Management Engineer for Systems Engineering Tools at Vector Informatik GmbH in Stuttgart (Germany).

DR. DIETER LEDERER
is Partner and Managing Director of Vector Consulting Services GmbH in Stuttgart (Germany).

DEFINITION

As a standard for the development of safety-related applications in the automotive industry, ISO 26262 [1] defines a basic framework for development processes and methods. Functional safety is viewed as an integral component of system development and is integrated in this process from the start [2]. To assure full coverage of safety goals and to document their fulfillment, the traceability of requirements must be documented down to the level of individual software work products such as code modules or test cases. If more complex systems are being developed, an integrated tool support is almost mandatory. In the following sections, these aspects are explained in greater detail, and the benefits an integrated tool support for system and software engineering are described. The technical example used for illustration purposes is a Lane Departure Warning (LDW) system.

HAZARD AND RISK ANALYSIS

The next step is hazard and risk analysis, which is described in Part 3 of ISO 26262. The goal is to examine all system functions and determine the risks that could result from potential failures of the system. The identified risks are initially described based on operating situations and modes in which they could occur. Then they are classified in terms of their probability of occurrence (exposure), severity of their effects and the potential for risk controlling (controllability) in order to minimise any damaging effects. This is followed by assignment of an ASIL (Automotive Safety Integrity Level) to the specific system function.

For all critical system functions, i.e. those classified as at least ASIL A, safety goals must now be defined whose fulfillment reduces the specific risk so that unacceptable effects are mitigated. An integrated system engineering tool, such as Vector's PREEvision, provides an environment for this, e.g. in the form of a table view that can be used to conveniently perform a hazard and risk analysis, ❶. Operating situations and modes can be predefined and standardised, and automated ASIL assignment can be used.

FUNCTIONAL SAFETY CONCEPT

In creating the functional safety concept, safety goals are refined by functional safety requirements, and allocated to the system functions that should fulfill these requirements. This approach progressively leads to the functional safety concept, which is independent of later implementation of functions in hardware or software. ❷ shows which system functions must contribute towards attaining the relevant safety goals. Underlying them are the associated safety requirements.

TECHNICAL SAFETY CONCEPT

In creating the technical safety concept, functional safety requirements are refined

Lane Departure:HazardAnalysis

Level	Hazard	Hazard Description	Operation Scenarios	Operating Modes	Exposure Comment	E	Severity Comment	S	Controllability Comment	C	ASIL	Safety Goals
1	Insufficient reaction	Counteraction measures are performed but not sufficient to avoid an accident.	Country Roads (E3)	High Speed	Could occur in almost all driving cycles.	E4	Could have potentially fatal consequences	S3	In most situations the driver will rely on other indicators to realise that he is straying from the lane	C3	ASIL-B	Detect Lane Departure
2			Main Roads (E4)	Cruise Control Active								
3			Motorway (E4)									
4	Missing reaction	A lane departure warning is not activated, even though the vehicle is straying from the lane without indicating.	Main Roads (E4)	High Speed	Could occur in almost all driving cycles.	E4	Could have potentially fatal consequences	S3	In most situations, the driver will rely on other indicators to realise that he is straying from the lane	C3	ASIL-B	Detect Lane Departure
5			Motorway (E4)	Cruise Control Active								
6	Reaction too late	Counteraction measures are performed, but too late to avoid an accident.	Country Roads (E3)	High Speed	Could occur in almost all driving cycles.	E4	Could have potentially fatal consequences	S3	In most situations, the driver will rely on other indicators to realise that he is straying from the lane	C3	ASIL-B	Detect Lane Departure
7	Unnecessary counteraction	Counteraction measures are performed although the vehicle is not straying from the lane.	Country Roads (E3)	High Speed	Could occur in almost all driving cycles.	E4	Could have potentially fatal consequences	S3	High probability that driver cannot control the situation due to high speed and unexpected action	C3	ASIL-D	Detect LD Faults

❶ Hazard and risk analysis

❷ Functional safety concept

into technical safety requirements, and system functions are assigned to elements of the technical system architecture, ❸. In an early stage of development, it is sufficient to make assignments to the components of the system. In a later stage, the implementation in hardware or software elements is done. ③ shows an excerpt of the component network that is relevant for the LDW system. Safety functions and requirements are assigned to the involved components. Automated consistency checks immediately reveal, for example, whether the ASIL of the system function to be implemented and the previous qualification of components are incongruent with one another, see the components framed in red.

Once the development has progressed to refinement to the software elements to be implemented, the resulting software architecture can be exchanged and enhanced with other tools, based on an Autosar-conformant description. This ensures consistency and traceability beyond tool boundaries.

SAFETY ANALYSES

Safety analyses such as FMEA or FTA are conducted to check the technical safety concept. The goal of these analyses is to identify potential weaknesses in the safety concept and eliminate them by suitable improvements. ❹ shows the schematic of a system FMEA, which is applied to the safety concept. The advantage of an integrated tool is evident here as well: The FMEA is performed based on the data describing the existing system, and the resulting measures are assigned directly to system components or elements. This makes it unnecessary to model the system structure in a separate FMEA tool, which would lead to both increased effort and potential inconsistencies and gaps in traceability.

❸ Technical safety concept

SAFETY CASE

The goal of the safety case is to demonstrate – comprehensively and traceably – that the system was developed so that it is free of unacceptable risks. The advantage of the integrated tool support that was presented in the previous chapters is once again evident here: all of the information needed for the safety case exists, is consistent and can essentially be called up at the press of a button and be exported as a report. This can be done at any desired point in time during development.

IMPLEMENTING MIXED-ASIL SYSTEMS WITH AUTOSAR

Regarding the technical safety concept for a LDW described above, the following challenge arises: if parts of system functions with different ASILs should be implemented on one system component, e.g. the "Driver Assistance" component shown in ③, then – in the absence of other actions – this component must be developed to the highest ASIL. However, it is best to avoid this in practice, because this increases effort and costs.

An advisable solution is to "operate" the parts of functions with different ASILs next to one another with mechanisms of the Autosar operating system such that they do not interfere with one another. This is referred to as freedom from interference, which must then be verified. This verification must consider three requirements that are described in the following three sections.

ACHIEVING FREEDOM FROM INTERFERENCE

Implementation of freedom from interference can be achieved by three modules in the Autosar basic software, which fulfill the following requirements:

- Correct timing behavior of the software: The Watchdog Manager is used to check for correct timing behavior. The user defines checkpoints, and when a checkpoint is reached the application software calls a function. If a checkpoint is not reached within the allowed time frame, or if checkpoints are reached in an incorrect order, a system restart will be triggered.
- Correct communication: End-to-end protection is used to verify correct communication between software components. Corrupt or missing messages are detected by transmitting and verifying checksums and sequential numbers for each signal group.
- Avoiding faulty memory accesses: The Autosar operating system forces avoidance of faulty memory accesses at system runtime using a hardware MPU (Memory Protection Unit). This involves partitioning the software into different "OS Applications" and only assigning those software components to one OS application for which it is known that the risk of mutual interference is acceptably low. The hardware MPU detects and prevents unauthorised accesses from one OS application to the memory area of another OS application.

CONTEXT SWITCH VERSUS COEXISTENCE OF SW COMPONENTS

Partitioning into separate OS applications enables implementation of a Mixed-ASIL system, i.e. safe use of software elements with different ASILs on a common hardware platform. However, when there is frequent communication between different OS applications, this causes longer execution times due to needed for the context switches. To counteract this effect, an alternative approach is to

❹ Autosar architecture with safety mechanisms

strive for safe coexistence of software components within the same OS application. Coexistence can lead to improved performance, especially for basic software components. One method for verification of freedom from interference with regard to memory accesses of the basic software by using a code checker is shown in [3].

AVAILABILITY

Vector Informatik is the first manufacturer to offer an Autosar operating system called MICROSAR SafeContext, which is certified up to the highest safety integrity level (ASIL D). It implements safe management of contexts and thus memory protection for Mixed-ASIL systems. The operating system is supplemented by the certified mechanisms SafeWatchdog and SafeCom from TTTech Automotive GmbH, which are also capable of levels up to ASIL D. These mechanisms assure correct timing behavior and correct communication of software components. ④ shows an overview of these mechanisms in the Autosar architecture. The individual mechanisms are supplied as Safety Elements out of Context (SEooC). Sufficiently general assumptions were made for the safety goals of the SEooC, and they are specified in detail in the safety manuals. To integrate these mechanisms into their own safety concept, users can perform a check of assumptions, as specified in ISO26262.

FROM SOFTWARE ARCHITECTURE TO CODE

As explained in section "Technical safety concept" the PREEvision tool can be used to generate the specific Autosar extract for each ECU. Based upon this description, another tool such as DaVinci Configurator can be used to create a consistent and optimised configuration of the Autosar basic software, the code of the Autosar RTE and optional templates for implementing the software components. Finally, implementation of the application software is performed in the user's usual development environment or by integrating code that comes from other sources. At the end of the development process, a complete implementation of the ECU software results, including configured Autosar basic software and RTE. The software is strictly based on the initially defined risks and safety goals and exhibits traceability.

SUMMARY

Implementation of ECU software according to ISO 26262 using an integrated tool environment, that supports specific methods such as hazard and risk analysis and FMEA on the one hand, and contains all system and software describing data on the other hand, is clearly superior to a not integrated tool environment. Consequently, integrated tool environments are expected to take hold significantly more in the automotive industry over the next five years, than is the case today.

REFERENCES

[1] ISO 26262, Road vehicles – Functional safety, Parts 1 – 9, 2011 und Part 10, 2012

[2] Lederer, D.; Ebert, C.: Funktionale Sicherheit – Das Gesamtsystem Fahrzeug". In: Hanser automotive 10 (2008), pp. 20-24

[3] Heling, G.; Rein, J.; Markl, P.: Koexistenz von sicherer und nicht-sicherer Software auf einem Steuergerät. In ATZelektronik 7, S7 (2012), pp. 62-65

VECTOR >

Safe ECUs According to ISO 26262

Solutions for Functional Safety

Benefit from Vector's solution for functional safety in your ECU development projects:

> Develop and analyze safety-related E/E architectures with PREEvision
> Implementation of safety requirements up to ASIL D in ECUs using MICROSAR Safe
> Training and coaching to qualify technical specialists and managers, and consultation on safety-related development projects

Information and downloads: **www.vector.com/safety**

Vector is your competent partner in developing safety-related ECUs – based on experience, dedication and practice-proven products.

Vector Informatik GmbH | Germany · Austria · Brazil · China · France · India · Italy · Japan · Korea · Sweden · UK · USA | www.vector.com

Is This What the Future Will Look Like?

Implementing Fault Tolerant System Architectures with AUTOSAR Basic Software

Highly automated driving adds new requirements to existing safety concepts. It is no longer sufficient to simply deactivate a function to reach a safe state. In the future, a safe state will require energy and active functionality. This article shows available mechanisms and explains how they can be modularly combined to attain an effective safety concept. It also aims to create an awareness of the challenges of future fault tolerant systems and shows that they can be overcome effectively with AUTOSAR.

In safety-relevant systems of today's vehicles, the most frequent reaction to a fault is to deactivate or reset the faulty function. This is referred to as fail-silent. It is easy to implement this type of solution, and it is effective for achieving a safe state and maintaining it. However, E/E systems in the vehicle are increasingly assuming other functions that must remain available in case of a fault, e.g. when a microcontroller fails. This behavior is referred to as fail-operational and in the following as fault tolerant. In the future, the demand for fault tolerant systems will increase substantially in manufactured automobiles. One example: in some of today's heavy SUVs, it is necessary to keep steering assist systems active to assure that drivers can handle steering safely. While development for fail-silent systems is now mastered quite well with ISO 26262, issues in the design of fault tolerant systems are still difficult to resolve with ISO 26262. In particular, attempts to come up with a precise definition of a safe state are still causing headaches in this context. The second edition of ISO 26262, whose publication is planned for 2017 or 2018, will not achieve final clarity either. Aside from the requirements set in standards, the following chapters show how existing safety concepts can be extended to fault tolerant systems using AUTOSAR technology.

Modular Safety Concept for Fail-Silent Systems

Safety engineers use a modular concept to efficiently tailor the various safety mechanisms to a specific project **(Figure 1)**. Here they make a rough distinction between measures for microcontroller integrity, measures for functional monitoring and comprehensive measures.

Measures for establishing integrity of microcontrollers are selected according to the highest Automotive Safety Integrity Level (ASIL) of the software that is used. They are independent of the function to be performed, and they are determined by the required diagnostic coverage (DC) for a specific ASIL.

Microcontroller manufacturers often set specific requirements based on their safety analyses. For example, a DC for ASIL D requires built-in self-tests (BITs) that are executed periodically by the software. Generally, starting with ASIL B the probability of occurrence of so called Single Event Upsets (SEUs) must be considered. Microcontrollers in lock-step mode and memory with error detection and correction codes (ECC RAM, ECC ROM) offer effective protection against SEUs. Both safety mechanisms are realized in the hardware, are nearly transparent to software development, and are therefore very efficient solutions.

The developer normally implements additional mechanisms in the application to perform functional monitoring. They include monitoring tasks for sensors and actuators, as well as limiters and program flow monitoring (logic monitoring). Program flow monitoring can be achieved with an AUTOSAR watchdog, for example.

Functional monitoring and microcontroller integrity are defined and implemented according to the specific project. However, there are also mechanisms that are used in nearly every safety-related ECU and are independent of functionality and ASIL. Almost every ECU with ASIL software also executes QM software. To ensure coexistence according to ISO 26262, memory separation and monitoring of time constraints are needed [1]. Memory partitioning is realized by an AUTOSAR operating system with Scalability Class 3 (SC3) that controls a memory protection unit (MPU) with the required ASIL. The watchdog usually handles monitoring of time requirements by deadline monitoring. As soon as safety-related data is exchanged between more than one ECU, communication protection comes into play. AUTOSAR offers an effective safety mechanism for this purpose in the form of end-to-end protection (E2E). Products from Vector that are certified up to ASIL D are available for implementing these comprehensive measures.

Transition to Fault Tolerant Systems

For cost reasons, today's hardware is designed to be nearly redundancy-free. Therefore, a hardware fault generally leads directly to a serious functional degradation up to complete failure. On the other hand, mature methods exist for quantifying hardware failures, such as those defined in IEC 62380 and SN29500, which permit predictions of the target failure rate [2].

It is often difficult to quantify software faults, since they are exclusively systematic [3]. Timing protection is a suitable safety mechanism for boosting fault tolerance with respect to software faults. Timing protection guards against such faults as infinite loops in software components that prevent execution of the actual functionality. In timing protection, the developer assigns time budgets for the execution times of tasks and interrupt routines and for the blocking times of interrupts and resources. The time intervals between tasks and interrupt routines are also monitored **(Figure 2)**. In case of a fault, the AUTOSAR operating system can terminate the task or interrupt routine that is causing the fault and exclude it from further execution. However, timing protection is only a first step toward the fault tolerant systems that will be needed in the future.

Figure 1: A modular concept enables efficient tailoring of safety mechanisms to a specific project.

Figure 2: AUTOSAR timing protection offers early detection of a violation of the allowed time budget for tasks and interrupts.

Fault Tolerant System Architectures

Fault tolerant system architectures have been used for many years now in the aerospace industry. For the safety-critical flight control systems, three or four ECUs are redundantly combined to form a complex system. This redundancy in the hardware is of course extremely cost-intensive. Therefore, new approaches must be sought for using fault tolerant systems in the automotive industry. This industry also benefits from lower severity of the consequences of failure in risk assessments.

A feasible system architecture **(Figure 3)** always consists of at least two channels. In this example, one channel comprises a sensor, a logic unit and an actuator [4]. It is obvious that when the microcontroller of one channel fails, the associated software and its functionality fail as well. Due to their complexity, microcontrollers frequently have the highest failure rates in an ECU. Therefore, proper execution of a function cannot be assured for even a very short time period.

To make this two-channel system fault tolerant, each channel must detect all individual errors for itself and switch itself to passive [5]. Without this requirement, both channels are needed for safe operation of the function. However, in this case the failure rate would double and not be halved as desired. Of course, this system architecture requires a redundant energy supply for the two channels, just as it requires a redundant communications path to relevant partners. The IEC 61508 standard identifies such a system as a 1-out-of-2 with diagnostics (1oo2D).

Figure 3: Example of a fault tolerant system architecture achieved by redundant design.

Software Architecture with AUTOSAR for Fault Tolerant Systems

In principle, introducing redundancies into the hardware also increases complexity in the application. This creates new challenges in the area of control engineering, such as how to achieve controller stability and control actuators when redundant controllers are active at the same time. It is also necessary to reassess data consistency in networks ("Byzantine Generals problem" [6]). From a system architecture perspective, this complexity could be limited by a hot-standby mode, for instance. In this case, only one of the two channels controls the actuators at any given time. If an error occurs on this channel, the other channel immediately assumes control. The AUTOSAR basic software **(Figure 4)** is useful for simplifying the application development process for the following reasons.

> Re-use: The AUTOSAR components presented above that are used to achieve a modular safety concept can also be re-used by the developer for error detection.

> Use of existing mechanisms: There are two philosophies when it comes to implementing the software of redundant channels: diversity or homogeneity. The diversity philosophy uses different software on the two channels. With the same types of channels and microcontrollers, it is possible to use the same software that is simply parameterized differently for each channel. This is done with the Post-Build Selectable Mechanism of AUTOSAR, which is normally used to develop ECU variants. The use of the same types of channels requires examining errors with the same cause [7].

To enable a quick switchover of control to the other channel in case of a fault, sensor and actuator values as well as status information on the channel are exchanged between the channels **(Figure 3)**. The mechanisms of AUTOSAR make it possible to implement just one configuration of the basic software. The developer can either implement channel switching application-specifically as a software component (SWC) in hot-standby mode or exploit the flexible configuration options of the AUTOSAR manager components Basic Software Mode Manager (BSWM) and ECU State Manager (ECUM). Today, application-specific soft-

Figure 4: MICROSAR software architecture for fault tolerant systems based on AUTOSAR

ware is implemented to exchange the error status between the channels. In the future, however, it is conceivable to specify standardized basic software components for exactly this purpose.

Tool Support

To overcome the added degree of complexity in the future, effective and comprehensive tool support needs to begin in an early development phase. This frees up resources for focusing on application development and relieves engineers of tedious and error-prone work such as manual consistency checking of redundant signals in the system model.

Outlook

Today, on-board AUTOSAR capabilities already permit efficient implementation of safety-related projects. If increased fault tolerance of E/E systems is required, new system architectures will be needed that can handle the failure of the microcontroller as well. This presents new challenges for the application and the basic software. Nonetheless, the complexity of these types of systems can be mastered using AUTOSAR methodology. Here, AUTOSAR basic software offers an excellent starting point due to its configurability. The associated tool chain simplifies management of the necessary redundancy.
In the future, assistance will be further improved by the basic software and its tools. A first step, however, is to realize that fault tolerant systems also require new approaches in the system architecture.

Translation of a German publication in Elektronik automotive, 11/2015

Image rights::
Lead figure: Fotolia #76500668, Originator: Jakub Krechowicz, with modifications by Vector Informatik GmbH
Figures 1 - 4: Vector

References:

[1] Definition of the fault tolerant time interval (FTTI) in ISO 26262-1, 1.45
[2] ISO 26262-5:9 Evaluation of safety goal violations due to random hardware failures
[3] ISO 26262-10:4.3 Relationship between faults, errors and failures
[4] Definition of a system in ISO 26262-1, 1.129
[5] ISO 26262 Single-point fault metric (SPFM) for ASIL D
[6] The Byzantine Generals Problem, L. Lamport et al, ACM Transactions on Programming Languages and Systems, 1982
[7] Definition of a Common Cause Failure in ISO 26262-1, 1.14

Links:
Website Vector: www.vector.com
Product Information MICROSAR: www.vector.com/microsar

Jonas Wolf (Dipl. Ing.), Vector
has a degree in Aerospace Engineering from the University of Stuttgart. He has been working at Vector since 2012; currently he works in the role of Senior Product Management Engineer for Functional Safety and Cyber Security.

Safety and Performance with ASIL D AUTOSAR Basic Software

In electronic control units (ECUs) that are developed according to ISO 26262, safety-related and non-safety-related software is often used in parallel. To prevent them from interfering with each other, partitioning measures have previously been used. However, this partitioning often leads to overhead in runtime and higher complexity. The AUTOSAR basic software, developed entirely according to ISO 26262, reduces the number of partitions to a minimum. A report from Vector Informatik.

The part of ECU software that performs safety-related functions is growing constantly. This leads to a more frequent interaction between safety-related application software and the AUTOSAR basic software. If the two are located in different partitions, the interaction takes additional computing time because, at the very least, the active task must be changed and, depending on the hardware platform, the memory protection unit (MPU) may need to be reprogrammed as well. However, basic software developed according to the requirements of ASIL D, allows its coexistence with safety-related application software in a single partition. This allows a significant performance gain, since task switching, reprogramming the MPU, and additional copying of data can be avoided. Moreover, additional but frequently needed safety requirements can be implemented in the basic software, making it unnecessary to develop them repeatedly for the individual ECU projects. **Figure 1** compares the two approaches graphically.

Background

ISO 26262 defines the automotive safety integrity level (ASIL) as an attribute of a function implemented in software. It indicates the quality with which the function is to be implemented. The ASIL is derived from the hazard analysis and risk assessment of the item under development. It is assigned to the safety goals and propagated to the functional and technical safety concept, down to the safety-related software requirements. As soon as a software component implements a safety-related requirement holding a certain ASIL, the corresponding methods for its development, taken from Part 6 of ISO 26262, must be applied **(figure 2).** Well-known examples of such safety-related functions in the basic software are memory protection by the operating system, temporal monitoring by the watchdog stack, and the safeguarding of communication using end-to-end (E2E) protection as described in the AUTOSAR specification [1].

Figure 1: Partitioning concepts with QM and ASIL basic software.

The use of AUTOSAR E2E protection is the state of the art for safeguarding safety-related signals. Since it allows all relevant faults to be detected, no safety requirements are imposed on the basic software's communications components. In order to avoid partitioning between E2E protection and communications components, the methods from ISO 26262 Part 6 must be applied for the communication components as well. This is because it is assumed that software with a lower ASIL contains more systematic faults than software developed according to a higher ASIL. Without partitioning, these faults can affect safety-related software as cascading faults, thus violating the designated safety requirements. Interference with safety-related data and with the safety-related timing requirements of the software is possible. The potential hazards caused by interference are to be prevented. The solution ISO 26262 [2] offers is developing non-safety-related software components using the same development processes used for safety-related functions. Those software components are then considered to provide sufficient freedom from interference.

Consequences for Development

An implication of this argument is that, among other things, an extensive semi-formal design must be developed for the software in question. In general this implies additional development efforts to achieve the requirements for ASIL D. However, the increased effort during design and implementation is low, compared to the high demands on verification. This is especially true for testing.

The requirement of diverse implementation of algorithms as the result of the software safety analysis can be derived from ISO 26262 Part 6 Table 4. The algorithm is implemented in two different ways in order to prevent systematic faults. To avoid this diverse implementation, Vector builds on the excellent testability of its components. This is demonstrated by the low cyclomatic complexity of the software components under test. Cyclomatic complexity is a measure of the paths in the control flow graphs of a function. In order to meet the strict requirements of component testability, many components had to be refactored, i.e. their source code structure had to be improved. This process was also used to improve defensive programming, meaning the increased robustness resulting from additional checks, and the modularization within the components.

Another challenge is the extensive configurability of the AUTOSAR basic software. ISO 26262 only deals with this configurability partially. In particular in the area of unit tests, ISO 26262 implicitly assumes only one possible code variant. For the basic software to be used in a wide variety of configurations as a safety element out of context (SEooC) [3], an approach had to be found to cover this configurability. This configurability is partially reflected in the pre-processor conditions. Vector deduces possible configurations on the basis of use-cases and dependencies between configuration parameters. In its development process, Vector applied the same standards for the exit-criteria to its pre-processor conditions as it did for runtime tests [4]. Moreover, Vector has defined additional exit-criteria for testing in its component development: Complete code coverage for runtime tests in each configuration variant that is necessary for fulfilling pre-processor coverage, must be achieved.

Static code analysis is an important part of the process, since it allows certain types of errors, such as memory corruption, to be found significantly more reliably than a functional test. Specifically for the context of configurable software, Vector developed a static code analysis for detection of out-of-bounds access and invalid pointers.

Figure 2: Development of software components according to ASIL resulting from safety requirements or a desire for coexistence.

Because of the many variants of the basic software, this involves automated verification of the generated configurations that is performed by the user. This ensures freedom from interference with respect to memory between non-safety-related parts and safety-related parts of the basic software.

Safety Concept

The safety-oriented development process described above allows the definition of new safety requirements for the SEooC basic software because almost all relevant requirements from ISO 26262 have already been met. Vector has formulated additional safety requirements for the basic software: detection of loss, corruption, and masking of data in non-volatile memory – caused by the lower layers of the basic software or the hardware. Restarting of software applications and AUTOSAR timing protection can be used as a safety mechanism in the future, too. Vector has also assumed safety requirements to bring an ECU into a defined condition during initialization. The safety requirements that are necessary for partitioning are usually still required. For example, QM software, which is often present, must be integrated because a cost-benefit analysis has shown that new development or qualification (acc. ISO 26262 Part 8 Clause 12) is not economical.

Assumed safety requirements must be analyzed carefully and the underlying assumptions communicated before these requirements can be used in a safety concept of an ECU project. This means that only functions that are part of the assumed safety requirements may be used in the project-specific safety concept. For example, E2E protection is necessary to transfer safety-related data because Vector has no assumed safety requirements for the communications components, even though they have been developed according to the methods in ISO 26262. A list of assumed safety requirements should be made available in a very early phase of a given project.

Integration

After the user of the basic software has validated the assumed safety requirements with the project context, he uses the safety manual included in the delivery to perform a configuration and integration of the basic software. A unit test for the basic software is normally already performed by its supplier. The integration and verification step, as described in ISO 26262 [5], must be performed by the user, however. The basic software supplier cannot contribute to the safety case at these levels, since he has insufficient information about configuration, application, and the exact target environment during development of the basic software.

The user receives confirmation of compliance with an ISO 26262-conformant development process through the safety case for the basic software. The user references this safety case for the creation of his safety case then. The safety case is created on the basis of the configuration submitted by the user, among other things. This check of the configuration by Vector is necessary for validating the component test. Not all possible combinations of configuration parameters can be tested during development. In order to be certain that even exotic configurations are tested, project-specific testing is necessary.

The effectiveness of this approach has been confirmed in the exemplary assessment by the certification company exida in an independent assessment. During the assessment the operating system, the components for CAN, LIN, and FlexRay communication, and the components in the system and memory cluster were evaluated. **Figure 3** shows the current overview of the components developed according to ASIL D.

Figure 3: Current overview of the components of the AUTOSAR 4 basic software developed according to ASIL D.

Outlook

The approach that has been introduced shows how runtime overhead and the number of partitions can be reduced with ASIL D basic software. Additionally, the complexity in an ECU project decreases because safety mechanisms do not have to be implemented by the user again, but are already included in the basic software.

Vector's goal is to develop more AUTOSAR 4 basic software components according to its new, safety-oriented process, thus offering an optimum solution for as many application cases as possible. Challenges faced in the near future are AUTOSAR components, which undergo many changes. Another challenge is the extensions provided by OEMs and the hardware drivers of the various semiconductor manufacturers, which are not yet available at the necessary quality and can therefore be used with safe basic software only with increased effort and sacrifices to performance. In an additional step, Vector and exida are working on the implementation of an RTE that can be used in the context of safety-related software without costly qualification measures.

More information on Functional Safety and ISO 26262:
www.vector.com/safety

Literature References:

[1] AUTOSAR Specification of SW-C End-to-End Communication Protection Library
[2] ISO 26262 Part 9 Clause 6
[3] ISO 26262 Part 10 Clause 9
[4] ISO 26262 Part 6 Clause 12
[5] ISO 26262 Part 6 Clauses 10 and 11

Author
Dipl.-Ing. Jonas Wolf studied aerospace engineering at the University of Stuttgart. He has been with Vector since 2012 and now serves as the Product Management Engineer for Functional Safety.

Author
Peter Müller worked at TÜV Süd after graduating from the Mannheim University of Applied Sciences with a degree in functional safety. In 2002, he joined exida, where he has been Lead Assessor for Certification in the automation and automotive area since 2007.

Translation of a German publication in HANSER automotive, July/August 2016

Image rights: Vector Informatik GmbH

Has functional safety ceased to be a topic of interest?

The topic of "safety" is rightfully playing a dominant role in today's discussions about automated driving. It is proper and necessary to discuss the aspect of cyber-security intensively here, because highly automated driving requires integrating the vehicle into the world of the Internet – whether it involves sharing information, executing a software update or controlling functions remotely. This makes the protection of personal data important, but above all it increases the probability of an attack with the goal of manipulating the vehicle or using it maliciously.

No Safety without Security

Since such an attack can impair safety, the often cited principle of "No safety without security" is absolutely correct. Unfortunately, the converse statement does not hold true, namely "If we do not have any security problem, then functional safety is assured." Expressed differently: The aspect of functional safety deserves and requires our unlimited attention and must not under any circumstances be relegated to the shadows of discussions about security.

Functional safety is a topic that has been discussed for many years now, and it has also been fundamentally mastered. That is why, at first glance, it might seem justified to devote less attention to this topic. On a second look, however, it is a big step to advance from "fundamentally mastered" to "applied consistently and well over broad areas".

> The number of affected functions, ECUs, companies and therefore affected persons has grown in recent years and will continue to increase. It is no longer sufficient for functional safety to be relegated to a small group of specialists.

> It is necessary to analyze all levels and interfaces – from the total vehicle to individual ECUs and finally the individual ECU elements. Many parties are involved, and efficient solutions can only be achieved if there is a clear understanding of who will contribute what.

> The transition from driver assistance for the purpose of enhancing convenience to systems the driver can rely upon to autonomously drive the car requires substantial growth of functions. Even in case of failure, these systems must provide at least degraded functionality. That is , the standard is changing from "fail-safe" to "fail-operational".

No Safety without Safety

ISO 26262 forms a valuable foundation for the development of safety-relevant systems and for certain aspects also makes very specific practical guidelines. Many discussions have shown, however, that there are guidelines that require interpretation. In these cases, sharing of knowledge and experience between experts is helpful.

A working group of the ZVEI (German Electrical and Electronics Industry) has been compiling professional experience from a large number of safety projects at various companies. This has resulted in the best practices guideline "Software for Safety-Related Automotive Systems" which was published at **http://www.zvei.org/SSRAutomotive** on the ZVEI homepage. The first version introduces software safety concepts and architectures and provides advice on the classification and qualification of software tools.

This treasure trove of experience enables the dissemination of relevant concepts and approaches, and it thereby makes a contribution toward making ECU software functionally safe. That is because, as important as security may be, safety for the vehicle and for the driver is based on two pillars: functional safety and data security. The foundation for both is a consistent methodology.

Dr. Günther Heling
is director of the Embedded Software product line at Vector Informatik GmbH and leads the "Software ISO 26262" working group within ZVEI.

Translation of a German ZVEI publication in Automobil Elektronik, issue 07-08/2016

Image rights: Vector Informatik GmbH

Secure communication for CAN FD

Encrypted data transmission is not yet the norm in vehicle networks. Vector has conceived an implementation for secure communication over CAN. Protection goals were authentication and preventing replay attacks.

Author

Armin Happel

Principal Software Development Engineer
Vector Informatik GmbH
Ingersheimer Str. 24
DE-70499 Stuttgart
Tel.: +49-711-80670-0
Fax: +49-711-80670-111

Link
www.vector.com

In today's vehicle networks, data transmission is for the most part performed without any special security measures. Because of this, it is possible to read out the data transmitted in raw format or to even play it into the bus system in modified form if you have direct access to the vehicle bus. Encrypted data transmission would not only ensure that this information could only be evaluated by authorized recipients. At the very least, it would also make it much more difficult to intercept or alter the messages.

Media reports about vehicle manipulation [1], [2] raise the question of whether data in the vehicle network can actually be influenced by manipulation. Can a manipulated device or internally implanted device with a remote control function influence vehicle behavior? And what countermeasures can be taken to prevent such manipulations?

Today's vehicles are highly complex systems, which consist of networked sensors and actuators and continually transmit important data over bus systems. In the vast majority of cases, the information being transmitted is in raw data format. A plausibility check, if such a check is even possible, has limited effectiveness. The receiver is unable to verify whether the data was actually supplied by the desired sender or whether it was fed in by an outside electronic control unit, i.e. whether it is authentic data. The data is freely accessible as well, so an analysis of the bus information can be used to determine signal contents. The transmission is neither confidential nor authenticated.

Figure 1: Message transmission and timing of encrypted communication

This was the problem that engineers at Vector were confronted with. Their task was to come up with an implementation for secure communication over a CAN network which can be used flexibly and can also be integrated with Autosar-3.x basic software. Protection goals were authentication and preventing replay attacks. It was also desirable to implement communication that cannot be monitored

For the encryption method, the specialists chose the AES algorithm [3]. From today's perspective, this method is considered cryptographically secure. It involves symmetrical block encryption with a block length of 128 bits. It generates 16 bytes or a multiple of 16, which the sender transmits to the receiver. An additional advantage is that some microcontrollers already have very fast hardware-based implementations of this algorithm.

Since a CAN message can transmit a maximum of 8 data bytes per frame, a decision was made to utilize the ISO transport protocol (TP) that was already included in the communication stack for the transfer. To simplify the configuration and reduce protocol overhead, a unidirectional communication with a fixed 1:1 relation between sender and receiver was chosen.

Symmetrical encryption requires that both the sender and receiver have the same key. The software modules that are used permit dynamic allocation of the keys at runtime, so that the user or OEM can freely choose them. A higher-level method such as a (asymmetrical) key exchange method might be implemented, or a static allocation might be made such as in end-of-line programming. Whenever an ECU is replaced and a vehicle specific key is used, the new ECU must be set up by an authorization method, which keeps the key confidential under all circumstances.

Figure 2: ID keys of multiple receivers in the use of CAN FD

Preventing replay attacks

In this configuration, an encrypted transmission of messages is now possible, where the information is, however, still purely static, i.e. a unique key text can be assigned to the plain text signals. This means that replay attacks, i.e. recording excerpts of a desired communication and replaying it into the system at a later time, can still be made. That is because the receiver cannot check whether the message actually originates from the sender at this point in time. To make checking possible, at the start of communication the receiver generates a random value – which is referred to as the ID key in the following – and it communicates this to the sender. The sender increments the value with each transmission and appends it to the transmit message. When the message arrives, the receiver checks whether the ID key matches the expected value. If it does, it processes the message; otherwise it rejects it. To tolerate possible message losses, the receiver will also accept a slightly higher value. This means that the counter in the transmit message continually alters the encrypted data even if the signal contents remain the same (Figure 1).

Depending on the word width of the ID key and the frequency with which the message is sent, overruns of the counter value might be expected in the message, which would lead to repeated transmission of the encrypted message. To avoid this, the ID key is only valid for a certain time period. When this period expires, the receiver must generate a new value and communicate it to the sender. Immediately after receiving a new ID key, the sender transmits the encrypted message. This means that the receiver is also able to initiate repetition of a message, such as if the received ID key does not agree with the internal key, and this reduces latency times. Although the sending node receives and considers new ID key messages for a time T(offset), to avoid an overload of the bus system such messages do not immediately lead to resending of the encrypted message. To make the protocol more robust, the receiving side uses the timer T(Resent) to monitor the response of the sender with the new counter value. If it does not get an acknowledgment message from the sender, the receiver generates a new ID key and resends it. This makes it possible to detect even a brief failure of the sending ECU and shortens the time for resending. It also avoids storage of the ID key in nonvolatile memory.

Data transmission without segmentation

There is a significant disadvantage associated with segmented data transmission in CAN over the ISO-15765 transport protocol. Transmission time is increased, and this method is restricted to a fixed 1:1 relationship, because segmented data transmission over ISO-15765 is very difficult to implement with multiple nodes. CAN FD on the other hand enables simultaneous transmission of the entire encrypted message to multiple receivers [4]. Each receiver needs the same symmetrical key to decrypt the encrypted message. Two variants of the ID key for authentication come into consideration: either all receivers agree on a commonly agreed value, or all receivers independently generate and send their ID key to the sender. The sender manages all counters and appends them to the data message. The positions of the counter values within the encrypted message must be uniquely assigned to the receivers.

Figure 2 shows data transmission for multiple receivers. First, the receivers transmit their randomly generated start values to the sender. The sender then increments all ID keys for each send cycle and insert them into the encrypted message at the predefined positions. The relevant

Figure 3: Software components for encrypted transmission

Literature

[1] http://www.chip.de/news/CAN-Hacking-Tool-Autos-hacken-fuer-20-Dollar_67066892.html [only German]

[2] http://www.can-newsletter.org/engineering/engineering-miscellaneous/140822_list-of-potentially-vulnerable-cars_blackhat/

[3] Advanced Encryption Standard (AES), FIPS PUB 197

[4] CAN with Flexible Data Rate – Specification Version 1.0, Robert Bosch, GmbH; April, 2012 http://www.bosch-semiconductors.de/en/ubk_semiconductors/safe/ip_modules/can_fd/can.html

receiver then checks its ID key and accepts the data or rejects it (Figure 2).

However, as the number of receivers increases, this reduces the message space that remains for useful data. The number of useful data bytes is also highly dependent on the selected word width of the ID key. The communication timing illustrated in Figure 1 was applied. It only required a modification for the sender in receiving the ID key. Instead of immediately transmitting the encrypted message, the sender waits for a configurable time T(IdKeyReply) to allow time for any other ID key messages from other receivers. The special case T(IdKeyReply)=0 covers the original method.

Vector implemented the protocol for CAN FD in a CANoe environment. The specialists subjected the protocol to extensive tests using this software tool for development, simulation, and testing of ECUs and networks. Along with the required robustness against replay attacks, another focus was to study message losses, failure, and re-entry of sender and receiver as well as timing errors and burst attacks. In all of these cases, the encryption system provided a stable transmission.

Summary and Outlook

In CAN FD, in particular, it took relatively little effort to implement robust transmission of encrypted data with multiple nodes, and this method can also fit into an existing Autosar environment. One disadvantage is the serialization and deserialization of the data on the application level (Figure 3), which means that modeling properties of the RTE cannot be used any longer for individual signals. The classic points of attack on such systems must still be kept in mind. They include, for example, weak random number generators for the ID keys (at startup) or spying the symmetrical keys.

In the security technology world, the AES-128 algorithm is considered secure for the near future, and its implementations are mature or will even be supported by hardware accelerators. The method presented here makes attacks on the CAN (FD) communication much more difficult, and manipulation is hardly possible without "insider knowledge". It has already been in production use for several years, and it also has led to favorable classification of the relevant vehicle for insurance premiums. In this case, security not only protects data; it even offers a direct cost advantage to the end user.

In the near future, remote connections such as Car2x communication, WLAN, Bluetooth and Internet will continue to grow and will necessitate much more stringent requirements for IT security. These access modes must be made secure against attacks and must not permit any remote manipulation. This is especially true of information to driver assistance systems, which rely on reliable messages from other traffic participants and/or the infrastructure. ◀

VECTOR

Reliable Protection for Your ECUs and Networks

Automotive Cyber Security Solutions from Vector

In developing security ECUs, you benefit from a comprehensive Vector solution consisting of software tools, embedded components and services:

> Homogeneous solution for testing security-protected ECUs and networks
> Highly automated testing of security mechanisms by using fuzzy testing
> Efficient and configurable AUTOSAR basic software for implementing security requirements
> Flash bootloader for secure booting and updating of ECU software
> Systematic security engineering and creation and validation of security concepts

More information: **www.vector.com/security**

Develop your security projects reliably and efficiently with solutions from Vector.

Webinar series with valuable know-how
and free tips from experts.
Register now at: **www.vector.com/sec-webinars**

Vector Informatik GmbH | Germany · Austria · Brazil · China · France · India · Italy · Japan · Korea · Sweden · UK · USA | www.vector.com

Cyber Security – Challenges and Practical Guidance

Success Factors for Security Engineering

Cyber security is of a growing concern across industries. It is today not anymore nice to have, because systems are interconnected, and in one way or the other open for external penetration. Even worse security directly impacts functionality, user experience and safety, and thus has become subject to product liability. For instance, functional safety is not feasible without a concise approach to cover cyber security. Based on the specific challenges of automotive security, OEMs and suppliers have to realize an effective protection against manipulations of electronic systems. Key points in the development of protected systems are the proper identification of security requirements, the systematic realization of security functions, and a security validation to demonstrate that security requirements have been met. Based on our experiences inside Vector and with client projects around the world, we will show typical challenges and highlight suitable security guidance. We will show with concrete examples how these practices improve developing secure systems and how these activities can be performed efficiently.

1. Introduction

Night drive on the highway. The communication display flickers and suddenly annoying whistling sounds abruptly from of the speakers. The driver is surprised, looks to the display, tries to calm down the whistling tone – and causes an accident. Reality or fiction? With increasing complexity and networking, the use of standard components and open interfaces, the various electronic systems, especially in the infotainment, increasingly vulnerable and must be protected. Functional safety today needs information security. What exactly is information security (IS)? Basically IS is a system property and relates closely to availability, functional safety, robustness and performance. Quality in a system will generally mean that the system does everything that is expected of him. IS goes beyond this classical definition in the sense that the system itself in a malicious attack does nothing which is not expected.

Information security increasingly impacts today and future electronic systems. A growing number of automotive comfort, reliability, and safety functions are realized by software applications that are executed by distributed electronic control units (ECUs). Besides the further development of innovative sensors like radar and camera systems and highly complex infotainment and assistance systems, connected cars are pushing tomorrow's innovation. Internet connections will not only provide the need for information to the passenger. Functions like eCall or communication between cars or car to infrastructure (Car2x) will evolve traffic. This includes improved traffic flow controlled by intelligent traffic lights, OEM backend systems to provide latest mapping data, warnings from roadside stations or brake indication of adjacent cars. Enhanced driver assistant systems and automated driving are gradually appearing on our streets.

Figure 1: Modern car connectivity

More connectivity bears also the risk for attacks to the car. From painful experiences in standard IT and communication domains it is known that distributed systems and communication are subject to unauthorized manipulations, such as eavesdropping, manipulation of message contents, use of resources, etc. The data processing and communication activities in automobiles are no exception – in fact, the properties of vehicles and their E/E architecture can make unauthorized manipulations even easier than in many other systems.

Figure 1 shows car connectivity on-board in the distributed systems as well as from the car to its environment. Each connection has a potential risk for an attack, regardless if it is wireless or wired. Just the threat is different. The access through a connector shows limited risks, because it is only possible for single cars, whereas a far field connection is much more dangerous because it can be accessed from anywhere in the world. But also near field connections play an important role, such as tire pressure monitoring system, Bluetooth and wireless LAN which is accessible from passing by or parked cars.

Anything that can be exploited by an attacker will be if the gain is significant enough. Therefore, it is required to ensure that automotive systems can either not be exploited or the gain of manipulations is reduced below profitability. Security and reliability will be essential for the acceptance and success of modern automotive IT and electronic systems. In the standard IT and telecom domains, a plethora of methods, techniques, best practices, and patterns have been created in order to develop IT systems that are less vulnerable to unauthorized manipulations. This discipline of security-aware development is referred to as security engineering – it deals with processes, methods, and tools for designing, implementing, and testing secure IT systems.

In this paper, we depict how security engineering can be successfully incorporated into automotive systems development. Section 2 therefore summarizes the challenges that are specific to automotive security. In section 3 we present solutions that proved valuable in several automotive security engineering projects. Based on our experiences with these solutions we have identified several activities that leverage the successful application of security engineering – these are outlined in section 4. This article is continuously updated and in its latest version available through Vector Consulting Services[1].

2. Challenges of Automotive Security

Security directly impacts safety, and thus is crucial to protect the driver and its environment. Automotive OEMs and suppliers today are well aware that insufficient security engineering has direct and immediate effects on product liability. **Figure 2** shows how security attacks impact safety by creating hazards, either by intended malfunctions like in our introductory story, or by unintended malfunctions for instance due to DOS attacks. Security engineering is not rocket science and thus with sufficient competence and willingness can be implemented – and must be maintained throughout the lifecycle of any vehicle.

Automotive security has to deal both with coincidental events (e.g. defective ECUs becoming babbling idiots) and, especially, with events that are intentionally caused by persons with certain motivations, such as:

- Activation of features that can be enabled by software settings, e.g. "chip tuning".
- Car and vehicle parts theft.
- Component plagiarism.
- Manipulation of remote key systems, immobilizer systems.
- Manipulation of telematics systems, e.g. to obtain navigation systems' maps.
- Furthermore, new features such as car-to-X services might provoke further motivations for attacks.

[1] Contact us on the entire Vector security portfolio: mailto:consulting-info@vector.com or www.vector.com/security

Figure 2: Security Directly Impacts Safety

The exposure of automobiles and their E/E architecture provides several access points for attackers to mount an attack. It is often possible to physically access the E/E buses that provide for data transfer inside of the vehicle. Many of the relevant bus technologies are inherently insecure. Due to the growing connectivity of modern vehicles (e.g. car-to-X concepts, Internet connection), external communication interfaces such as LTE, Bluetooth, WLAN, and USB that allow for access to the E/E components might be used for manipulation. Additionally, the exposure of vehicles enables attackers to physically access the ECU hardware, which makes possible a wide range of manipulations, including passive (side-channel) attacks and active (non-) invasive attacks to compromise existing protection mechanisms.

Additional to the vulnerabilities of the automotive system, there is also a severe disproportion between the resources of an attacker and the resources of the vehicle, which places a strong disadvantage on software-based automotive protection mechanisms [2,3]. To compromise these mechanisms, an attacker has an arbitrary amount of time, whereas the time available for the execution of the protection mechanisms is short (e.g. for the encryption of a message). The same holds true for computational performance and memory capacity: an attacker can employ one or more high-performance computers while automotive ECUs possess only comparatively low processing and memory resources.

Automotive security is a relatively new issue. Hence there are not many experts available with expertise in both information security and automotive systems engineering. Moreover, the adaptation of the basic security building blocks (such as cryptographic algorithms) to the automotive environment and the composition of these building blocks to secure protocols are difficult.

To make matters worse, automotive E/E systems are not only complex but typically also developed and produced across different organizations. The car manufacturer (OEM) provides specifications of the system architecture and of the requirements on the different components. Multiple suppliers develop and manufacture the components. However, security itself is an emerging property of the entire system that cannot be overcome by one party alone.

3. Successful Solutions for Automotive Security Engineering

Automotive E/E systems comprise many different functions, ECUs, and communication systems, which can be attacked with different methods. The impact of an attack depends on the concrete function that is attacked. To reduce the complexity that is created by the huge number of targets, attack methods, and impacts, we suggest taking three different perspectives on automotive security, which each address a manageable, coherent set of questions. These perspectives and the corresponding questions are [2]:

> The product and its architecture: What can be compromised? What risks arise from potential manipulations? What has to be protected?
> Engineering for security: Which activities have to be performed to protect the vehicle? Which technical solutions exist for automotive security? How can these solutions be implemented according to the specific constraints? How can the security of the system be demonstrated?
> After sales: How are diagnosis, parts exchange, etc. affected by protection mechanisms? How to cope with vulnerabilities that are detected not until SOP?

These perspectives **(see Figure 3)** along with techniques that we successfully employed in client projects are depicted in the next sections.

3.1. The Product and its Architecture

The key to effective security is to know which assets have to be protected (and which do not). From an automotive viewpoint one is interested in what vehicle functions can be compromised by attackers and in the implications that result from an attack. Therefore, the security requirements have to be identified, before the development of security measures can take place. Without a thorough security requirements elicitation, critical assets may be left insufficiently protected or too much effort may be invested in protecting uncritical features. Like any other requirements, security requirements have to be complete, consistent, comprehensive, and testable. Unlike most other requirements, they are concerned with what a system should not do under any circumstances, including intentional manipulations. Therefore, specific methods are required to identify security requirements.

In a recent client project, we had to identify and prioritize security requirements for an automotive E/E component. Based on the size and complexity of the component development project and the given capacity for security engineering, we selected an agile approach [1] that was conducted in form of several workshops. The client's engineers provided expertise on the component's functions and their implementation. We moderated the workshop and provided expertise in security requirements analysis as well as knowledge on the used technologies' vulnerabilities and sensible protection mechanisms.

The key activities of the selected approach consist of the formulation and evaluation of abuser stories and the creation of security-related user-stories. Abuser stories are brief, informal descriptions of how an attacker might abuse the system. The abuser stories are written on index cards in a language understandable by developers. An example for an abuser story is: A chip-tuning firm flashes unauthorized software on the ECU. The index cards were then pasted to a white board with their respective positions determined by the probability of the abuser story and its impact, which gives a vivid impression of the risks of the abuser stories **(Figure 4)** – we defined the risk as the factor of probability and impact of an event.

Based on the risks, the abuser stories that have to be prevented were selected and security-related user stories were developed, which form the security requirements for implementing countermeasures to the identified threats. An example for a security-related user story could be: All ECU applications have to be authenticated before use.

The approach has proven valuable, because security requirements had to be identified in a very short time frame. The approach is lightweight, its results are concise, and it is well suited for application during workshops with domain experts, i.e. automotive engineers. However, it is less suited for complex systems because it lacks a systematic procedure – it strongly relies on the people and their knowledge both on the system and on security. Other approaches overcome these limitations, however at the cost of increased effort. An overview on security requirements analysis methods is given in [2].

It has to be noted for any approach that is used: To identify threats that emerge from technical features, an intimate knowledge of that technology as well as an attacker mindset are required.

3.2. Engineering for Security

While security requirements are concerned with what has to be protected, security engineering defines how the protection is realized. It affects all activities that are associated with "normal" engineering, such as system design, software construction, test and after sales. We regard each of these activities in the following paragraphs.

Design

In an ideal world, the design of a vehicle's architecture would consider security from the beginning on. Unfortunately, in reality many architectural decisions are determined by other factors, e.g. existing architectures of predecessor models, quasi-standard technologies, and cost factors. Additionally, security requirements are often not identified on product level but only on function level, e.g. in the case of security-affine functions, such as remote-key functions, immobilizer functions, or tachometer functions. As a result, many automotive E/E architectures are inherently insecure, what requires adding security as an after-

Figure 3: Perspectives on automotive security

Figure 4: Abuser story risk evaluation

thought. When designing protection mechanisms for specific functions, one is regularly faced with the following tasks:

> Ensure the operation of the function according to its specification. This requires the protection of the function's implementation against unauthorized change.
> Ensure the right operation of the function according to the current situation. This requires the protection of all information used as input by the function.

For both tasks, there are technical solutions available, such as cryptographic algorithms that provide integrity or authenticity on the basis of conventional and elliptic curve cryptography. However, we have to emphasize again that, even if established algorithms can be considered secure, designing a protocol that defines the communication, authentication, etc. is considered highly difficult and susceptible to failure, what might compromise its security.
While the protection of a function's implementation against unauthorized change usually does not influence other functions of the vehicle, the protection of the input information by a cryptographic protocol does. Therefore, information transmitted via the vehicle's E/E buses need to be secured. Because an attacker might be able to access these buses and to manipulate the messages, an implementation of a security protocol needs to be integrated into both the sending and the receiving ECUs **(Figure 5)**. Vector is very much engaged to orchestrate common security solutions starting with base software and even underlying hardware, rather than ad-hoc implementations specific to OEMs and components. For instance, Vector is supplying secure communication protocols within AUTOSAR for secure end-to-end communication [4].
In a project with an OEM we supported the development of a basic security architecture that defines a security protocol on system level for security-critical communications as well as reference implementations. A single security architecture can be tried and tested more intensively than a multitude of isolated functions and should therefore provide a higher degree of maturity and security. In the meantime we have developed and introduced a dedicated security protocol on top of the standard component requirements specifications [4].

Implementation

Not only the design of security-critical functionality is challenging, so is its implementation. First, there is a disproportion of the resources available to an attacker (long time span, high computing performance) and to the ECUs (short time span, low performance). Effective cryptographic algorithms are often resource-intensive and therefore usually have to be implemented specifically for the target ECU. Second, the code has to be robust against manipulations, such as inputs that result in buffer overflows. Third, a sophisticated attacker may perform hardware attacks such as side channel attacks or invasive attacks.
Resource efficient implementation is not specific to security, so we will not discuss it here. As we have seen in a recent client project, the introduction of a secure coding standard can reveal and subsequently reduce potential software vulnerabilities. Such secure coding standards, e.g. for the C programming language, ban insecure coding practices and undefined behaviors that can lead to exploitable vulnerabilities. As a matter of principle, we suggest the application of the MISRA-C guidelines and to use any measure one gets "for free", like setting the compiler to the highest warning level. The conventions have to be strictly checked and enforced, preferably with automatic analysis tools, such as LDRA's TBsecure.
For the protection against hardware attacks, one possible solution is to employ tamper-resistant hardware. In spite of such measures, we recommend designing the system in a way that the attack does not scale – i.e. compromised ECUs do not affect other elements. For secure implementation, too, the bottom line is that a strong fundamental knowledge of the language and the environment is required [4].

Verification and Validation

Security vulnerabilities can emerge both on conceptual level (e.g. insecure communications protocol) and on implementation level (e.g. susceptibility to timing attacks). Therefore a strict verification of the security concept is required, from the security requirements through the design of security-critical functionality (including protection mechanisms) to the implementation. For the development of security-critical software, the same verification mechanisms can be used that are recommended as good practices e.g. by Automotive SPICE, such as reviews, inspections, and tests on different levels.
In a client project, we performed reviews on the security requirements and inspections of the security architecture. The problems we identified in these early phases could be eliminated with comparably low costs. In this project we also created a simulation based on the design of the secure communications protocol using a widespread tool for ECU and network design and analysis, which allowed for an early evaluation of the security properties in a complex environment without the need for actual hardware **(Figure 6)**.
To validate system security (i.e. to check the actual system's performance against the system's security requirements), two approaches have to be considered: First, the functional test of the security mechanisms has to ensure their implementation conforms to requirements and that all of the security requirements have been realized. This does not differ from "normal" validation tests. Additionally, we recommend penetration tests, which simulate attacks on the system and may be independent of the security requirements. While the first approach can be performed by regular testing staff, the second one requires specific security expertise: the mindset of an attacker is required to design such tests and to interpret the test results.

3.3. After Sales

To enable efficient after sales activities in spite of constraining security mechanisms, several aspects need to be addressed. How can software updates be performed in the field with both security against unauthorized manipulations and justifiable logistical effort? Currently Vector is heavily engaged into secure communication protocols within the AUTOSAR base software to achieve a secure end-to-end communication on evolving protocol stacks. However, the concrete realization of the related logistical infrastructure needs to be considered.
An important aspect of after sales activities is the way OEMs and suppliers react when a security issue is detected

Figure 5: Security protocols have to be integrated into sending and receiving ECUs

Figure 6: Simulation and test of the security protocol using Vector modelling and test tools

in a fielded vehicle. Such scenarios have to be foreseen before the vehicle's SOP and procedures that define actions and responsibilities have to be set up. Actions to be planned are risk assessment of the issue, elimination of critical software vulnerabilities, and update of the software in the field. To achieve an efficient issue handling, a smooth cooperation between OEM and suppliers is required.

4. Levers for Security Engineering

Security engineering needs specifically tailored solutions. There is not one solution which fits to all architectures, products and practical use cases. For instance we are using with many clients a checklist which is adapted to the specific environment and security risks. The checklist though generic in nature, and for instance adopting lots of information security common criteria, needs to balance the risk exposure with cost for security. Some security measures can result in exploding cost and thus must always be taken with a good business sense. Too often we hear from clients that they had been forced to use IT common criteria or code checkers which create massive overheads. Similarly to a MISRA standard which hardly anybody uses without adapting to the specific environment, we need also to adjust security rules towards risks, cost and competences.
Independent of the approach used, we noticed several activities that support the introduction and the performance of security engineering in general **(Table 1)** [1,2,3,4]. We do not claim that these are the only valid solutions. Depending on e.g. corporate culture, existing experiences, and project size and complexity, other solutions may be preferred.

5. Summary

Security engineering is a challenging task for both OEMs and suppliers. While automotive security may not yet be in the customers' focus today, we should consider the lessons learned in other industries, where security is now a matter of course. In this paper, we depicted several security engineering approaches that we applied within the automotive industry.
We should take benefit from security experience in other critical domains

> Apply cybersecurity principles and methods to harden distributed embedded systems
> Adopt the IT standard "Common Criteria" for automotive IT infrastructure usage and create protection profiles

At the same time it is critical in this market to align the approaches for safety and security engineering to stay cost-effective

> Apply threat modelling as part of hazard and risk analysis
> Use security levels in relation to safety integrity levels, e.g., different security aspects, e.g. authenticity, integrity, confidentiality; relevance of assets and access options, e.g. attack intensity
> Align underlying development methods and processes, e.g., architecture design and evaluation, security design, coding rules, hardening, robustness and misuse testing, life-cycle robustness; tool support for combined safety and security engineering and consistency across the entire life-cycle

As a global market leader for many years, Vector supports all necessary automotive security engineering domains, starting with standard software such as AUTOSAR, providing a rich portfolio of modelling and test tools such as PREEvision, and also having the experiences to support consulting on how to build an efficient security and safety engineering in organisations around the world. **Figure 7** shows the Vector security portfolio and its levers to ramp up security.
The Automotive world has to catch up in security engineering. Looking to our recent experiences, organizations urgently need to build up some basic security expertise and obtain adequate external support, specifically where security meets safety. Mature development processes provide a good basis but need to be amended with security engineering activities.

This white paper is continuously updated and is in its latest version available through Vector Consulting Services: mailto:consulting-info@vector.com

References:

[1] Ebert, C.: Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing. ISBN: 978-0470636190-368. Wiley, Hoboken, NJ, USA, 2012.
[2] S. Burton, C. Ebert, et al: Automotive Information Security. VDI, Baden-Baden, 2007.
[3] Ebert, C: Functional Safety with ISO 26262 – Principles and Practice
https://vector.com/portal/medien/vector_consulting/webinar_podcast/Vector_FunctionalSafety_BestPractices_Webinar_EN.mp4
[4] A.Happel and C. Ebert: Security in Vehicle Networks of Connected Cars.
https://www.vector.com/portal/medien/vector_consulting/publications/Happel_Ebert_Security_Connectivity_2015.pdf

Dr. Christof Ebert
is managing director at Vector Consulting Services. He supports clients around the world to improve product strategy and product development and to manage organizational changes. He sits on a number of advisory and industry bodies and teaches at the University of Stuttgart.

Vector Consulting Services is a globally active consulting firm with focus on optimizing technical product development. Renowned companies from automotive, information technology, healthcare, transport and aerospace rely on the professional solutions for improving product development, product strategy and in organizational change management and interim management. A subsidiary of the Vector Group with over 1500 employees, Vector Consulting Services supports its clients worldwide with sustainable consulting solutions covering the entire product life cycle and the related processes and tools. The firm is managed by partners. This assures independent and customer-oriented consulting.
Details and further information:
www.vector.com/consulting

Activity	Benefits
Adapt the development processes to factor in security engineering activities.	Security engineering activities are known, scheduled, and executed smoothly within the "normal" development, not in an ad hoc way. Security is considered from the beginning on through the complete project. Additionally, synergies can be exploited (e.g. a configuration management process can prevent quick fixes that have not been tested against security vulnerabilities).
Systematically elicit security requirements.	Elements that have to be protected are known from the beginning on, allowing for stringent realization of their security. Additionally, security requirements can be used to deduce test cases for security validation.
Thoroughly review or test any security relevant artefact.	Reviews of security engineering artefacts such as security requirements and security concept as well as simulations of security functionalities and code analyses allow for the identification of vulnerabilities at the earliest possible time.
Use analysis and test tools.	Automated tools reduce effort and allow for efficient and comprehensive analysis and (regression) testing. For instance the Vector PREEvision PLM and modelling environment provides a strong collaborative engineering backbone for ensuring application of above measures along the life-cycle.
Manage embedded security competencies.	Many activities of security engineering require a specific embedded security expertise, e.g. identification of vulnerabilities, design of the security architecture, secure implementation, performance of security tests, and review of security-related work products. Without this expertise, effective security engineering is near to impossible. Therefore, build up embedded security competence in your organizational unit or obtain it from internal or external providers.

Table 1: Activities that leverage security engineering

Figure 7: Benefit from Vector Portfolio on Automotive Safety and Security

Cyber Security for the Automotive Industry

Practical experiences on the application of Cyber Security

Cyber Security highly impacts the automotive industry. The increasing complexity of systems way beyond the vehicle itself combined with open interfaces require rigorous risk management, continuous quality assurance, and a systematic development process. To foster a strong security culture, Vector has summarized here best practices in automotive cyber security. The article provides hands-on experiences from Daimler, Hella, Infineon, Porsche, Thales, ZF and Vector. Industry case studies emphasize how risk-oriented protection is implemented.

"Cyber security has arrived in all companies as a key challenge." This is the summary of Dr. **Thomas Beck**, CEO of Vector, from his experiences with companies worldwide. In 2007 when the Vector Group presented in Baden-Baden and spoke for the first time regarding the growing relevance of security in the context of functional safety with a practical contribution and own solutions, many companies had argued that security is no issue in the automotive area. This attitude has changed completely, as the recent Vector Symposium 2016 showed. The widely attended Stuttgart Liederhalle assembled car manufacturers, suppliers and IT companies to share their experiences on cyber security. Excerpts of the event and many practical tips are provided in this article.

There is no absolute security

"There will be no absolute security, since each complex system bears defects." This warning of Dr. **Christof Ebert**, Managing Director of Vector Consulting Services, gives a clear signal. Each company must establish a risk-oriented cyber security management that works reliably and traceable throughout the whole life cycle for its products, processes and supply chain. Experiences at Vector show that security is often limited to organically grown individual measures such as encryption. Crypto solutions, key management, code analysis, fuzz testing and firewalls are necessary, but their value is meager as long as vulnerabilities remain insufficiently addressed, be it consciously or unconsciously. About 90 percent of all security related incidents are due to human errors according to the experiences of Vector when surveying customers.

The answer is a risk-oriented culture where security is specified across functions, systematically developed and continuously verified and validated. Figure 1 depicts this value chain, emphasizing in particular on dangers arising internally or externally. It would be naive to assume that attacks come only from the outside. Vulnerabilities are caused by careless development processes, unsystematic releases, but also by deliberate errors, such as "back doors" and "master access". Companies have to secure the entire supply chain and in doing so deliberately change their perspective. It takes the ability to think like an attacker and on this basis take preventive actions as an engineer or manager. This protection must always work, especially when last-minute changes are implemented just before delivery, or variant regressions of a product appear years after the original release.

Holistic approach

Cyber security across industries has become a high commercial risk, since there is no absolute protection against hackers and abuse. **Lorenz Slansky** of Daimler AG underlines that cyber security is increasingly important for a brand, and consequently constitutes a competitive advantage. For critical systems, such as in-vehicle information, security is highly relevant and cannot be compromised by insufficient usability. Key issues in the development of protective systems are the complete identification of security requirements, the systematic implementation of security functions and security checks with the goal of consistent evidence that all the relevant security requirements are met. Security standards such as SAE 3061 help to learn from other industries and thus quickly implement essential solutions. Lorenz Slansky knows that proprietary mechanisms often pretend just supposed security and clarifies: "Security by obscurity does not prevent hackers, but delays good solutions" Dr. **Christian Meineck** of Porsche AG adds: "Security concepts must consider the vehicle and the IT networks inside and outside the vehicle." Figure 2 shows a subset of typical attack scenarios, and how they are evaluated by manufacturers. The times of individual functional units are gone, and the complex vehicle system demands solutions as they are already practiced today in complex IT systems, such as continuous software upgrades in a very short time and of course "over the air" (OTA). Standards foster fast learning from the IT by cross-fertilization and thus prevent that errors are repeated in every industry again and again.

Due to the high degree of interconnected functions in the vehicle but also to the outside by increasingly open interfaces not only a specific function is affected, but also other functions that belong to a system group and interact with this specific function. This includes non-electrical characteristics. For example, a warning of mechanical problems or wear-out may no longer function after a functional incident or attack on a bus system. **Masahiro Goto** and **Martin Prisching** of DENSO showed that safety and security aspects must be considered as a pair. To achieve this, the attack tree analysis (ATA) is supplemented by the Fault Tree Analysis (FTA) in order to systematically cover both type of risks. It is obvious for them that "all layers of the security architecture must be addressed for continuous security to achieve the well-known concept of 'Defense in Depth'. Figure 3 shows with the example of Cooperative Adaptive Cruise Control (CACC) how various security measures protect against threats in the network.

Katharina Lohmann of Hella observes specifically in verification and validation a huge need to move forward. While process frameworks such as SPICE are used to systematically prevent errors in engineering and maintenance, there is an urgent need to improve verification such as systematically scanning each new software regression before further testing and delivery, as well as introducing new testing procedures as fuzz testing and penetration testing.

Figure 1: Risk-oriented Security must consider the entire life cycle.

Figure 2: Attack scenarios in the vehicle.

Dr. **Günther Heling** of Vector highlights from his own experience in development and delivery of advanced AUTOSAR basic software that only an automatic supply chain with rigorous release criteria creates the necessary quality.

Cryptography as a solution

Cryptographic methods are today an essential part of security solutions. **Klaus Schmeh** of cryptovision assumes that the AES encryption procedure with 10 to the power of 38 possible keys will not be cracked in the foreseeable future even with quantum computers. A major challenge is to integrate this and other security mechanisms in a meaningful way to vehicle IT systems. High performance requirements often contradict and eventually inhibit security, as Dr. Andre Weimerskirch of the University of Michigan emphasizes. In the world's largest field test of Public Key Infrastructure (PKI) for car-to-car (C2C) and car-to-infrastructure (C2I) communication, he combined performance requirements and data protection. His logic is simple: "Priority one is functional safety, and priority two is privacy." **Dietmar Hilke** of Thales develops solutions for advanced security requirements. Design for security to his experience is an important tool, and he cross-fertilizes medical immunology with IT. Just as there are multi-resistant germs, the attacks are constantly developing new patterns. With systemic resilience the resistance will be expanded continuously. Security begins in design, but is a task for the entire lifecycle – no matter how long that is.

Figure 3: All Security layers are necessary for thorough protection

The companies agree that consistent risk-oriented security cannot solely be achieved by patches on ECU and network level. Security needs to address the overall architecture. Figure 4 shows the reference model of Vector with a clear separation of subnets. Between these subnets firewalls and intrusion detection systems are placed. Dr. **Eduard Metzker** of Vector sees such distributed security architectures under the current environmental circumstances as the best way, because they can be implemented in a stepwise mode. This requires a layered security architecture as it is propagated by Denso and other Tier-1 suppliers. Starting with a hardware-based anchor and AUTOSAR with its state-of-the-art crypto solutions, the individual components are individually hardened. Dr. **Achim Fahrner** of ZF emphasizes in this layering the importance of a crypto-co-processor and hardware security module (HSM) to build resilience and robustness bottom-up. The communication to safety-critical subsystems like steering and brakes should have no direct connection to infotainment and other vulnerable systems. This is especially true in 'Over the Air' (OTA) upgrades that are increasingly being used. **Axel Freiwald** of Infineon underlines that while OTA reduces the number of callbacks, it must also be carefully protected before it is used in existing networks. Not only are the IT requirements growing, such as rollback and availability, but also the secure distribution of software to the respective control units.

Companies must strengthen their competences in cyber security solutions, as well as systems and IT architecture in order to progressively modularize the organically grown architecture. At the same time they need to implement a strategy for sustainable support of various organizational units. Operationally this can be steered by a security manager, who is responsible for the release of architectures and component releases. The traditional change control boards need to be strengthened technically but also in the way they are working, to evaluate the threats on the basis of defined criteria and assess different solutions before implementation. Periodic architecture reviews and adjusting the test strategy with focus on security will deepen the mutual understanding among developers and managers that security measures are not a straw fire, but must be implemented in each single project. Such concepts cannot simply be imposed: They need strong anchor points in the organization. Vector reports from its projects that such behavioral changes make security sustainable because the teams have a tangible and concrete responsibility.

Security needs systematic handling

The cyber security projects of Vector give one clear message: The line between systematic implementation of automotive cyber security and ineffective ad-hoc measures is small and thus requires professional assistance. Security attacks exhibit a common pattern where strength does not exploit weakness but intelligence does exploit recklessness. This means for instance errors in the configuration of firewalls and gateways, not fully coherent software changes, and complicated user interfaces so that security measures remain in their default status and are thus vulnerable to attacks. On the basis of threat assessments, provisioning dedicated infrastructure components and tools for verification and monitoring, up to architecture reviews and

Figure 4: Security by design in practice: Different subnets should be separated as much as possible

security consulting Vector has gained wide experiences. Only a consistent life-cycle will facilitate cyber security: From the initial hazard analysis to architectural decisions, verification to regression tests at each new delivery and after-sales change management. That is also the impulse of Prof. Ebert of Vector: "Sustainable cyber security, that ought to be more than a flash in the pan, needs continuity and cooperation with professionals."

For more information on Cyber Security and Access to the mentioned presentations and videos, please visit:
www.vector.com/security

Author
Dr. Christof Ebert is Managing Director of Vector Consulting Services GmbH.

Author
Dr. Eduard Metzker is Solution Manager Security at Vector Informatik GmbH.

Translation of a German publication in Elektronik automotive, special issue on Software September 2016

Image rights:
Teaser image: Vector Informatik GmbH
Image 1: Vector Consulting Services GmbH
Image 2: Porsche AG
Image 3: Denso Automotive GmbH
Image 14 Vector Informatik GmbH

Risk-Oriented Automotive Cybersecurity

Industry Experiences and Case Study

Vehicles like any other open and distributed IT system are subject to security attacks. Cybersecurity matters for almost any automotive domain, be it dynamics or powertrain or infotainment, because without security, there is no safety. Therefore, advanced risk assessment and mitigation is the order of the day. Risk-oriented security engineering for automotive systems helps in both designing for robust systems as well as effective mitigation upon attacks or exploits of vulnerabilities. Security must be integrated early in the design phase of a vehicle to understand the threats and risks to car functions. The security analysis provides requirements and test vectors and adequate measures can be derived for balanced costs and efforts. We will show with concrete examples how risk-oriented cybersecurity can be successfully achieved in automotive systems. Three levers for automotive security are addressed: Product, i.e., designing for security for components and the system, Process, i.e., implementing cybersecurity concepts in the development process, and Field, i.e., ensuring security concepts are applied during service activities and effective during regular operations.

Automotive Cybersecurity

With the introduction of connectivity and increasing amount of IT in vehicles, precautions must be taken to increase the reliability and to reduce the vulnerability to the system [1]. The basic principle is simple to understand, yet difficult to achieve: Functional safety needs security [1,2,3]. Based on the specific challenges of automotive security, OEMs and suppliers have to realize an effective protection against manipulations of automotive E/E systems [1,4].

Automotive security has gained huge relevance in short time-frame. Attacks are reported today almost continuously and therefore systems must be protected and hardened. Safety needs security as a mandatory condition, which means that any safety-critical system as a minimum must also be protected for cybersecurity. Security must be integrated early in the design phase of a vehicle to understand the threats and risks to car functions.

Key points in the development of protected E/E systems are the proper identification of security requirements, the systematic realization of security functions, and a security validation to demonstrate that security requirements have been met. The following items need to be considered to achieve security in the car development process:

> Standardized process models for a systematic approach which is anchored in the complete development process. This starts on the requirements analysis through the design and development down to the test of components and the network.
> Quick software updates to close vulnerabilities in ECU software.
> Reliable protocols that are state-of-the-art and meet long-term security demands. Related to security this is often combined with cryptographic keys. So, a key management over the lifecycle of the vehicle must be maintained.
> In-vehicle networks and system architecture that provides flexibility and scalability and are designed under consideration of security aspects.

Figure 1: Security Engineering along the Life-cycle

Risk-oriented security helps to balance growing security threats with increasing complexity over the entire life-cycle. Unlike many previous attempts our research and many practice projects indicate that while design for security is good, it is not good enough. Effective security must handle the entire life-cycle. For covering the major risks related to safety hazards caused by security misuse and abuse scenarios, we combine for automotive risk-oriented security engineering activities of both safety and security engineering **(Fig. 1)**. While safety and security are very different methodologies, we realize that at present, both organizational infrastructures and governance are much better in many companies for functional safety than what they are for cybersecurity. So we see this path as an evolution towards full (independent) cybersecurity organization in the life-cycle.

Risk-oriented Security

Developing secure software is challenging for several reasons, namely because increasingly systems are connected, most software is developed in a global context in heterogeneous teams with various skills, systems complexity is exploding with embedded and IT systems converging such as IoT, and both budget and cycle times are continuously decreasing. For instance a modern car has almost hundred embedded microcontrollers on board and is connected over several external interfaces to a variety of cloud technologies. At the same time cyberattacks and vulnerabilities are increasing. Therefore, software technology and the underlying security engineering have to be constantly improved.

While there is a movement towards better understanding security from the ground up, many of the existing approaches in managing security have been focused around encryption, developing malware software, and to detect attacks to networks and systems. Existing methods and tools are limited by large number of false positives and inability to consistently trace such issues to the root causes. In this article we will particular draw attention to all aspects of security from specification to design and life-cycle support.

Over the past decade trends like connected car and driver assistance systems among others have led to software and connectivity playing an increasingly important part in developing vehicles and also for business models of OEMs and suppliers likewise.

Devastating impact of security issues is already known from industrial sectors like IT-infrastructure, aviation, information technology and telecommunications, industrial control systems and energy and financial payments. Virtually every connected system will be attacked sooner or later. A 100% secure solution is not feasible. Therefore, advanced risk assessment and mitigation is necessary to protect assets. Consequently, the typical solution to security in these industries relies on suitable risk assessment that projects threats on assets of interests. Thereby cost of implementing specific security measures can be compared with the probability of a particular threat that they counter.

Asset-based risk assessment is a suitable tool for companies to steer efforts for security engineering in a systematic and comprehensive way and thereby involve all relevant stakeholders in the organization. For example, a CEO may not find it very helpful to have a long exhaustive list with every attack vector or potential threat – they need to be provided with a ranked listing and useful decision-support tools which clearly shows alternatives and consequences. From the view of an automotive system developer, a flat listing of potential threats might not help to improve the system. To really help, they need to be able to map security threats, countermeasures and requirements to system/architecture elements in their scope of the project.

The systematic management of security threats and associated security goals is essential to actually providing safe and competitive products, and to protect valuable assets and business models.

But what makes security engineering so complex? Automotive developers face the challenge of securing a system against attackers whose capabilities and intentions are at best partially known. Some attacks might today appear infeasible, but todays impossible attacks might become more likely in the near future. An example of this is attacking a vehicle simply by exploiting wireless interfaces, 20 years ago would have been extremely unlikely, however today a cheap software defined radio and accomplish these types of attacks with little effort. On the other hand, an attacker might invest more effort into launching an attack the more valuable a successful attack is to him. Some attacks represent more effort to the attacker than others given the specific potential of the attacker. It is this risk/reward payoff that is analyzed in security engineering. Likewise during testing and verification, suitable methods to verify that the vehicle has the required security level and process goals like, test strategy and coverage, need to be chosen.

Furthermore, the assets to be protected from attacks are decided by stakeholders involved, e.g. drivers would indi-

Figure 2: Overview of risk-oriented cybersecurity analysis process with the major steps of asset determination, attack potentials and security goals derived from threats

cate different assets of their vehicle to be protected compared with what an automotive developer considers an asset. However, customers/drivers need to be satisfied with their vehicle in order to buy another one from the same company. Consequently, security engineering must seek tradeoffs between cost of security measures and benefit to assets in order to make sustainable decisions.

Security concepts must balance the cost of not having enough security and thus being successful attacked with all damaging consequences and the cost spent to implement appropriate security mechanisms and keep them updated along the life-cycle of the car – well beyond end of production. We therefore introduce here a strict risk-oriented approach to security **(Fig. 2)**.

The relationship between assets, attackers and threats is complex and dynamic (e.g. attacks are more probable the less effort is required and the more value successful attacks represent; attack vectors and effort change over time). Furthermore, common understanding of assets among all stakeholders of security engineering is mandatory in order to provide information for steering the security engineering. Choosing the right set of security engineering methods for analysis, concept and testing is challenging but required in order to enable goal-oriented and manageable security engineering.

Risk-based Security Engineering combines state-of-the-art methods for automotive security risk assessment in a practical framework and supports all involved stakeholders to develop "secure-enough" products. The method and our approach for proposing a concrete technical security concept is based upon security best practices such as:

> SAE J3061 (Automotive cybersecurity) being the first standard on the topic of automotive cybersecurity but also being aware that it primarily enriches ISO 26262 towards security for functional safety [3].
> ISO 15408 (Evaluation criteria for IT security) with its focus on IT systems, specifically the 7 evaluation assurance levels (EAL) for security requirements and guidance on common criteria a standardized practice translated by Vector to automotive common criteria.
> ISO 27001 (Information security management systems) with its governance requirements for security engineering across the entire value chain.
> IEC 62443 (Industrial communication network security) with its strong view on distributed systems and necessary security technologies and governance.
> ISO 26262 (Automotive functional safety) using its clear focus on automotive electronic systems with good coverage of entire life-cycle; revision in 2016 [2].
> IEC 61508 (Functional safety for electronic systems) while being aware that it is only a high-level functional safety guidance for electronic systems.

Our Vector Security Check and underlying security engineering methods have adopted the state of the practice in security evaluation and proposed mitigation [EB2016]. It is using significant research work from our worldwide security projects. It also uses external best practices, such as "E-safety vehicle intrusion protected applications" (EVITA) funded by European Union [Ev2017], HEAVENS [Is2014], and other proposed methods for security risk assessment in automotive development [Se2017, Si2017, Pr2017, ETSI2010].

We will furtheron show by examples how to use the risk-oriented security concept covering the entire security life-cycle with focus on the upper left activities, namely

> Asset Definition and Threat and Risk analysis
> Security Goals
> Security Concept

Two aspects related to security in embedded systems have to be considered: (1) Attack scenarios go well beyond individual components and functions. (2) While safety deals with avoiding critical failure modes, security has to cope with intelligently introduced causes of faults, which is far more difficult, given that the attackers' intelligence, willingness, determinedness, and creativity often exceed that of the engineers looking to a problem from the – different – perspective of how to solve it, and not how to find loopholes and strange feature correlations.

Risk-oriented cybersecurity connects security requirements directly with design decisions and test – following the triple peak model [4]. This ensures full traceability from the initial TARA and definition of security requirements. From a compliance and governance perspective we see this approach helpful as it instruments the necessity to prove that security requirements and decisions have been adequately verified – with each single regression. For instance we have introduced this risk-oriented cybersecurity to a leading tier-1 supplier in an ASIL-D environment and ensured that road tests cycle time could be dramatically reduced.

Figure 3: Risk-oriented cybersecurity connects security requirements directly with design and test

Figure 4: Increasing connectivity drives complexity and enables multiple attack paths.

Figure 5: Security zones in automotive networks

Case Study: Powertrain ECUs

With vehicle dynamics being a major safety risk, automotive cybersecurity has to also consider powertrain. Different attacks can be imagined, such as getting access to critical intellectual property up to manipulations to enhance vehicle functions, and not the least to impact the driver and thus directly the passenger safety. Considering security aspects in the basic structure from the very beginning can provide essential advantages for the flexibility and scalability of such a network. It can also reduce the risk for attacks. The major attack scenario sin automotive vehicles are described in **Fig. 4**.

Security components like firewall and router are also important parts and are useful to separate networks. For example, account computers will not be connected to the same network as that from marketing or development. Instead, computers are grouped to separate networks depending on their use case and traffic. A router interconnects the networks and provides data exchange between them. It passes only the relevant and allowed data from one network to the other. The access to computers is already restricted by the structure of the network. The router with an integrated firewall also manages the access to the internet. This device observes the incoming and outgoing traffic and can be configured that only allowed traffic will pass. Additionally, maintenance can be done easily on the central part by applying patches or re-configure the device if needed.

Let's consider a car network under the aspects shown above. At the very beginning, a safety and security analysis has been performed and the networks are partitioned so that the connected items are grouped under functional, safety and security aspects. The different networks are interconnected by a gateway. The ECU with a remote connection is considered particularly as unsafe. Even if great care was taken during the software implementation, a failure cannot be excluded and the potential risk is too high that someone could capture and take control of the ECU from outside. To minimize the risk, this ECU should not have access to any other internal networks. We locate this function into a separate ECU (inter comm. module) and connect it through the gateway. The gateway can now contain a firewall that has separate filter rules for each subnet. Only those messages are passed to other networks that are allowed. The traffic inside a network is not restricted and affected by the firewall.

We now take advantage of our threat analysis and risk assessment that has provided a detailed security analysis of our system. In the data flow diagram we saw the interaction of signals between partitioned functions separated in ECUs. This has already helped us to separate the ECUs to different networks, respectively partitioned the networks according to safety, security and functional aspects. We can now classify networks into security zones according to the safety and security requirements of the transmitted signals. If a network mainly contains signals with high security and safety requirements, that is classified as high security zone. The network with intermediate safety and security data is classified as a medium security zone. The network that contains just a few signals with safety and security requirements and many signals from remote connections is a low security zone. A network that contains an ECU with a remote connectivity must be treated as unsafe in principle and is therefore in a low security zone or even completely isolated **(Fig. 5, 6)**.

The origin and distribution of the signals influences the settings of the firewall in the gateway. The presence of signals from other security zones gives an indication to the security measures for the internal signals. If a network is physically isolated and signals from other networks are rarely used, the threat potentials are low. Unless other threats from adversaries are identified, reduced measures can be applied for signals in such a security zone. This reduces efforts and costs for security measures of these ECUs. It shows how partitioning provides advantages. The signal flow from high to low security zones is not critical. However, if threats for data manipulation on the network are given, security measures like authentication or confidentiality can be added to the data. Greater care must be taken in the other direction. The risk potentials for these signals must be observed carefully. Also, if filter rules of the firewall can influence the complete security zone settings. For counter measures, authentication on signal may be required.

Security analysis provides requirements and test vectors and adequate measures can be derived for balanced costs and efforts. The results are useful in the partitioning phase when functionality is distributed to ECUs and networks. Networks isolated under security aspects helps to reduce the risks and efforts. Security key management will become an important part and requires a key infrastructure (PKI) managed by the OEM over the production and maintenance phase of the vehicle. Additionally the secure key

Figure 6: Complete reference architecture with security zones for automotive networks

handling inside an ECU and the usage in development, production and maintenance phase must be considered. The PKI must be online to allow access by the workshops. Additionally, an OEM backend is needed that allows flash programming over the air, at least to provide hot fixes and patches. Such a backend can provide additional security and features to the car owners, but can also open new business divisions for OEMs.

Conclusions

Security and related measures demand well-founded concepts all along the life-cycle of both components and the car itself, especially if their effectiveness has to be proven at a later point due to legal actions. With this article we strive not only to provide guidance for specific misuse cases but to change the mentality of engineers of embedded systems towards designing for security – rather than for functionality.

Telecommunication and information systems have realized many years ago that isolated mechanisms (e.g., distributed functionality in proprietary subsystems, protection on component-level, gateways and firewalls between components, validation of critical functions) are insufficient. This article underlines together with concrete examples how automotive security can be achieved. We have taken three different perspectives, namely (1) the product and its architecture (e.g., specification of security requirements, misuse and abuse cases, vulnerability analysis, inherently secure architectures); (2) engineering for security during the development process (e.g., FMEA and hazard analysis as a basis for security, protection on component- and on system-level, systematic verification, code analysis, validation on product-level); and (3) the relevant after-sales activities in the field (e.g., fault analysis, patch and correction handling, emergency response and handling, distribution of corrections and protective mechanisms).

References

[1] Ebert, C. and A. Braatz: Automotive security engineering, Vector White Paper 2018.

[2] ISO 26262, ed.2, draft - Road vehicles – Functional safety, Last accessed on 12. Mrc.2018, www.iso.org

[3] SAE International: "Cybersecurity Guidebook for Cyber-physical Vehicle Systems, J3061_201601", 2016, www.sae.org

[4] Ebert, C. and J.Favarro: Automotive Software. IEEE Software, ISSN: 0740-7459, vol. 34, no. 3, May 2017.

[5] M. Islam et al.: Project overview HEAVENS – Healing Vulnerabilities to Enhance Software Security and Safety, Volvo AB, 2014.

Dr. Christof Ebert
is Managing Director of Vector Consulting Services.
He supports clients around the world to improve product strategy and product development and to manage organizational changes. He sits on a number of advisory and industrial bodies and teaches at the University of Stuttgart.

Translation of a German article for Elektronik Automotive on the special topic of Safety & Security.

Figure 1: Discussions between leading experts from industry and science concerning forward-looking trends, key technologies and current research results in the field of electromobility at the 4th Vector E-Mobility Engineering Day 2017.

E-Mobility Ready for the Mass Market?

Trends, Progress and Experiences

It has long been impossible to consider electromobility as a separate entity. Instead, it needs to be considered in the context of future megatrends. Continuing urbanization, the expansion of renewable energies, automated driving and new business models such as car sharing and shared vehicles will, in the near future, fundamentally change the way we travel. This was the conclusion that could be drawn from the many presentations and discussions on the 4th Vector E-Mobility Engineering Day in Stuttgart on April 27th of this year. Alongside many numbers and facts, it became clear just how suitable for daily use e-mobility has become. This is demonstrated, in particular, by urban transport companies which are introducing electric bus fleets together with the associated supply networks. However, there is a need for further clarification at the levels of charging infrastructure, standardization and cyber security.

Today, there are many signs that electromobility will assert itself during the next ten years, including in Germany. In coming to such an assessment, we must not be deceived by the current situation in Germany, as a look at other countries or other parts of the world shows. For example, in Norway, approximately a third of newly registered vehicles are electric – thanks to massive state support. In the Netherlands, the corresponding value is still as high as ten percent. The other western industrialized nations have barely started, with a share of one percent, and with Germany bringing up the rear at 0.7 percent.

China and Asia in the Fast Lane

By contrast, the level of activity in Asia and the breathtaking speed of the changes taking place there are highly impressive. In 2016, 500,000 electric vehicles were registered in China, of which 115,000 were buses. In the field of electromobility, China wants to present itself as a driver of innovation and world leader. In effect, Asia is characterized by increasing levels of urbanization and, according to UN forecasts, the number of megacities worldwide with more than 10 million inhabitants will grow from 28 in 2015 to 41 in 2030. Most of these will be located in Asia, China and India. The desire for mobility in these countries will not decline; at the same time, however, the cities cannot put up with even more vehicle-generated air pollution.

Another aspect of electromobility is taxi drones for personal transport and developments in this field make such solutions a realistic vision for the future. The state transport authorities in Dubai (RTA) want to launch regular test operation of autonomous air taxis in the fourth quarter of 2017. Interestingly, the German start-up E-Volo has been awarded the contract for this ahead of its Chinese competitor. Promoters of an initiative in Switzerland are hoping to set up air corridors for taxi drones above the country's freeways. Although projects such as this may not be representative of the general situation, they clearly show how far changes to mobility in other countries have come and what direction the trend here in Germany may take in the longer term.

The Country Needs Fast Charging Systems

Further reductions in battery costs, together with technical innovations such as a 48V drive that is significantly more efficient than the current high-voltage systems, promise to bring about further improvements at the level of vehicle range. Despite this, it is clear that electromobility cannot take hold in the mass market unless there is a nationwide charging infrastructure. In particular, a dense network of charging columns with fast-charging capabilities will certainly be part of the solution. That is why many different vehicle and infrastructure manufacturers are joining forces as part of CharIN e.V. This internationally active association is driving forward the large-scale implementation of the Combined Charging System (CCS) that supports both AC and DC charging. CCS is being constantly further developed, for example in order to provide higher charging capacities. Starting at a level of 50 kW, the next aim is to achieve 150 kW and, finally, the level of 350 kW. In the quest for standardization, vehicle-to-grid communication as per ISO/IEC 15118 is a central issue. The standard regulates the establishment of bidirectional high-level communication and permits intelligent charging that is intended to cover all conceivable requirements. These include not only pricing, cost optimization and billing processes but also, and most importantly, the consideration of intelligent grid functionalities. How much energy is currently available and under what conditions? Should the vehicle be charged immediately at the highest possible charging capacity or is it better to wait for a more economical time? The next version, which is planned for the end of 2018, will include further features, for example inductive charging systems with radio interface, energy feedback into the supply grid and the temporary emergency supply of households, as well as pantograph charging with wireless broadband communication.

Support for Inductive Charging

Inductive, wireless charging has the potential to significantly increase the comfort and convenience of electromobility. Charging operations start simply without the need to laboriously connect a cable. Alongside ISO/IEC 15118, which covers the field of communication, ISO 61980 defines the

infrastructure requirements and ISO 19363 the vehicle-side prerequisites. Despite this, technological challenges persist. These range from real-time wireless broadband communication through the detailed positioning of the vehicle and the identification of the ideal coil frequency and the geometry and location of the coils, and on to the detection of foreign bodies or living beings in the area of the coil.
As yet there is almost no detailed experience to allow us to answer these questions. As a result, it is hard for standardization boards to define binding specifications. The STILLE project, which is supported by the German Federal Ministry for Economic Affairs and Energy (BMWi), has the aim of constructing, measuring and evaluating the systems and technologies that are to be agreed on in the standard and making the results available to the various boards. The companies that are taking part in STILLE have an international presence and most of them actively contribute to standardization. The vehicle manufacturers' and infrastructure providers' aim for Germany is to bring an interoperable infrastructure for inductive charging to the market in 2020.

Electric Buses as Forerunners

While general progress in the field of electromobility depends on many constraints and circumstances and proceeds with the corresponding caution, the decision-makers in niche sectors such as urban transport can act more flexibly and swiftly. These are, practically speaking, closed "ecosystems" because the vehicles are all equipped identically with regard to charging technology. Vehicles, batteries and charging systems can therefore be optimally harmonized for the intended area of use.
In the case of electric buses, it is necessary to choose between charging the vehicles in the depot (depot charging) and opportunity charging via a pantograph. While overnight depot charging is less spectacular, fast charging at selected stops, such as terminals or interchanges, is worthy of closer examination. When extended, the pantograph on the roof contacts a charging rail in the charging system and charges as much energy as possible in just a short time.
A number of urban transport companies in European cities are already running buses of this sort in test operation. The spectrum ranges from two buses in Copenhagen through to a fleet of 43 buses in Eindhoven. Solutions offering 300, 450 or 600 kW are possible for fast charging systems operating at voltages of 600 to 800 V. The 600-kW system achieves an impressive charging current of 1000 A. The systems comply with the ISO/IEC 15118 standard and support both CCS and pantograph charging.
The fact that each of these projects faces its own specific challenges means that operators and manufacturers have been able to collect a wide range of different experiences. Eight 18-m articulated buses are on the road in Cologne. Here, the manufacturer has, for the first time, developed charging systems that transform the 10-kV grid supply voltage into the required 800/600 V DC voltage in a single step. In this way, it has been possible to reduce losses by approximately five percent.

Eindhoven the Pionier

Those responsible for the large-scale project in Eindhoven have been able to gather considerable experience. To ensure that sufficient energy is available for all 43 buses at all times, fast charging equipment has also been installed in the depot. Depending on the situation, low- or high-capacity charging is possible here. To permit the uninterrupted use of the charging infrastructure, the Dutch infrastructure manufacturer Heliox developed charging stations with dual-connection for the depot. Thanks to a special DC switch mechanism, a bus can connect to the system even while the bus next to it is still being charged.
Even though cable-based charging was originally planned for the depot, pantographs are now also being used here. On the one hand, problems were caused by cable breakages due to buses driving over trailing cables. On the other, the bus drivers in the depot frequently forgot to connect the charging cable since, as a result of their driving experience, they were used to charging at the press of a button. This meant buses were often left immobile. The uniform operating concept should now ensure that this can no longer happen.

Figure 2: The Vector VC36PLC-24 control unit provides an interface for standardized communication between electrically powered vehicles and the charging infrastructure. It is particularly suitable for small series and supports DIN SPEC 70121 and ISO/IEC 15118.

Charging Control Units and Cyber Security

Vehicle and infrastructure manufactures as well as service providers who typically only produce or supply small series do not generally want to have to get to grips with all the details of standardization, charging communications and electronics development themselves. Despite this, they still need to meet the high standards of the automotive industry. For them, the best option is to make use of existing solutions with the corresponding embedded software.

Figure 3: The MICROSAR.V2G embedded basic software for smart charging communication in accordance with DIN SPEC 70121 and ISO/IEC 15118.

Vector Informatik has already developed charging control units with CCS type 2 plugs that are ready for series production and are suitable for urban buses, postal and logistics companies and similar users of small pools of electric commercial vehicles. Vector's charging control unit **(Figure 2)** supports DIN SPEC 70121 and ISO/IEC 15118. The control pilot signal corresponds to IEC standard 61851. The tried-and-tested Qualcomm QCA7005 chip is used for powerline communication (PLC). For backend communication, the control unit uses a bidirectional HTTPS-based connection. Consequently, the ECU makes use of proven protocols and technologies to ensure the secure transfer of sensitive data. Because both the hardware and the MICROSAR.V2G software **(Figure 3)** come from Vector Informatik, it is easy to meet customer-specific requirements.
Another very important topic in connection with electromobility is cyber security, for example during billing processes at public charging installations. A great deal of specialist knowledge and experience is needed in order to master the details of establishing secure connections using TLS or the secure exchange of certificate data – in particular the secret private key. That is why it makes sense to opt for external solutions when dealing with embedded systems for charging in accordance with ISO/IEC 15118. Vector Informatik's security concept is based, among other things, on a hardware security module (HSM) that optimally protects control units against external access. In contrast to a software-based security concept with secure hardware extension (SHE), this HSM approach permits asymmetric encryption. This means that neither party makes use of permanently installed keys but that both negotiate new keys at runtime on every transaction. This vastly improves security.

Conclusion

So far, electromobility has played a fairly subdued role in the public consciousness. Despite this, the participants and experts at the Vector E-Mobility Engineering Day all agreed that this situation is the equivalent of the calm before the storm. It is only a question of time before the final technical obstacles are overcome. Standardization work for the necessary technological extensions for vehicles and infrastructure is proceeding at full speed.
Even small series manufacturers are now able to call on proven technical embedded solutions and existing control units. We shall therefore just have to wait and see how long it takes before we in Germany can look up from our electric cars and see the first autonomous air taxis driving overhead, or whether we will have to travel to Asia to see such a sight.

More information on the E-Mobility solution from Vector
www.vector.com/e-mobility

Author
Dipl.-Ing. Dirk Großmann is Senior Manager at Vector Informatik, responsible, among other things, for developing the Vector E-Mobility solution.

Translation of a German publication in emobilitytec 03/2017 (August 2017).

Commercial Vehicles and Electromobility

Charging Ahead

Electric cars, buses and trucks share the same problems: range, charging speed and charging infrastructure. Important steps have been taken. But how does charging work in detail?

In the mid-term, the German post office Deutsche Post wants to move its entire short-range fleet over to electrically driven vehicles and has secured Streetscooter GmbH as its partner [1]. This is just one example of the way businesses are equipping themselves for future tasks. However, it is not just the corporate world that has its sights set on electromobility; local authorities are also facing a major challenge. In many cities, the limit values for exhaust gases have not been respected for years, with the result that driving bans are being discussed. Bans of this sort would have far-reaching consequences: Urban transport operators would not be able to carry any more passengers and shops would not receive any deliveries. To preserve the environment and set a good example, the Netherlands has set itself the aim of approving only electric buses for use in city traffic as of 2020. The Dutch city of Eindhoven currently operates approximately 50 articulated electric buses and wants to further expand this fleet in the future. Local authorities in Germany are taking similar steps: Articulated electric buses are already in use in Hamburg. Sixty more are due to follow by 2020. And in the historic Old City of Regensburg, small electric vehicles are already being used for passenger transport.

The Challenges of Electromobility

In the face of the high demand for electric commercial vehicles, such as buses, manufacturers are being forced to make a move. Studies and prototypes already exist. However, the market is demanding vehicles for everyday use. These must meet the requirements in terms of range, comfort of travel and operation and must function as reliably as modes of transport that use conventional engines. This is precisely the challenge raised by electromobility. For example, if the fuel level in a diesel vehicle starts to drop then the driver can simply stop at any gas station to refill quickly and easily. The ranges of e-vehicles are currently still very limited and the charging infrastructure leaves much to be desired. But who makes sure that a vehicle can recharge at any charging station? This is the responsibility of boards that standardize charging specifications, while also taking account of the circumstances typical of the various markets. Europe and North America have adopted the Standard Combined Charging System, while China uses GB/T 27930 and Japan uses CHAdeMO **(Figure 1)**. The greatest differences between the standards lie in the plug/inlet combination, the underlying physical connection and the protocol used for communication between the vehicle and the charging infrastructure. While GB/T 27930 and CHAdeMO are based on the CAN vehicle bus, the CCS-2 standard relies on an Ethernet-based protocol named Powerline Communication (PLC).

Figure 1: Three different standards for electric charging are used around the world (Source: CharIN e.V.).

Figure 2: Structure of the charging plug for AC or DC charging.

Different Plugs for AC and DC Charging

Another challenge for standardization lies in specifying a standardized plug. Unlike in the now widespread Automotive Ethernet standard 100BASE-T1, Powerline Communication does not use a twisted wire pair. Instead, the so-called Control Pilot (CP) signal uses a single-wire cable. The plugs and inlets for AC and DC charging always contain three identical pins:

> Control Pilot (CP): Pulse width-modulated signal with modulated, higher-level protocol
> Protective Earth (PE): Protective conductor and ground for CP
> Proximity Pin or Plug Present (PP): An analog signal that confirms that a plug is present at the inlet and codes the current-carrying capacity of the cable (relevant for AC charging)

There are also further pins for AC charging and the faster DC charging. The CCS-2 plug **(Figure 2)** that is used in Europe contains the following pins:

> For AC charging: Control Pilot, Protective Earth, Proximity Pin, L1, L2, L3, GND
> For DC charging: Control Pilot, Protective Earth, Proximity Pin, DC+, DC-

Alternatively, an L-conductor and the ground (GND) of the AC plug can be used for DC charging at reduced capacity.

The high-level protocols used for communication are identical for AC and DC charging. It is therefore possible to charge a vehicle with alternating or direct current provided that the standard is supported by the vehicle. The geometry of the plug makes incorrect charging impossible. This is also prevented at the level of the protocol.

Release for Charging: The Control Pilot Signal

The CP signal actually releases the charging operation **(Figure 3)**. The carrier signal is the 1-kHz PWM signal. In a correct duty cycle, it refers to the high-level Ethernet communication which is implemented as a modulated signal in

Duty Cycle of the CP PWM	Description
0% ≤ Duty Cycle < 3%	Charging not allowed
3% ≤ Duty Cycle ≤ 7%	Use of high-level protocol according to ISO 15118 and DIN 70121
7% < Duty Cycle < 8%	Charging not allowed
8% < Duty Cycle < 10%	Maximum power consumption is limited to 6 A
10% ≤ Duty Cycle ≤ 85%	Available power = Duty Cycle * 0,6 A
85% < Duty Cycle ≤ 96%	Available power = (Duty Cycle – 64) * 2,5 A
96% < Duty Cycle ≤ 97%	Maximum power consumption is limited to 80 A
97% < Duty Cycle ≤ 100%	Charging not allowed

Figure 3: Low-level communication for AC and DC charging via the Control Pilot.

the bandwidth 3 – 30 MHz over the powerline. But why is such an intricate mechanism actually needed for charging? The vehicle must first identify itself at the charging station in order to exchange information such as payment details, charging status, type of charging and amount of charge consumed. As soon as authentication has been successfully concluded, the vehicle (master) draws electricity from the charging station (slave).
The refueling standard for cars with internal combustion engines is implemented in the form of different refueling nozzle diameters. However, to charge an electric vehicle, today's drivers also have to track down the "right" charging station. To ensure interoperability between the vehicle and the infrastructure, the standards ISO/IEC 15118, DIN 70121 and SAE J2847/2 are used in Europe and North America. These specify the charging communication and ensure that data is exchanged correctly before the actual charging operation starts.

Figure 4: The MICROSAR.V2G embedded basic software for smart charging communication in accordance with DIN SPEC 70121 and ISO/IEC 15118.

Communication Between Charging Station and Vehicle

Embedded software is required in order to implement the communication protocols defined in these standards. Vector possesses many years of experience in implementing vehicle-internal network communication via, for example, CAN, LIN, FlexRay and Ethernet. That is why MICROSAR – Vector's AUTOSAR basic software – has been extended to include modules that support communication between the vehicle and the charging infrastructure. Among other things, this extension, which is known as V2G (Vehicle to Grid, see **Figure 4**), contains the software component vSCC: Vector Smart Charging Communication. This implements the entire protocol sequence for smart charging as defined in the above-mentioned standards. It is also already able to work together with the Powerline transceiver. AC and DC charging are supported for all vehicle manufacturers by means of various protocols, such as External Identification Means (EIM), Plug`n Charge (PnC) or Signal Level Attenuation Characterization (SLAC). On this basis, it is possible to implement customer-specific functions such as Value Added Services (VAS) via HTTP and DNS in order to establish parallel, proprietary communication between the charging station and the vehicle and thus exchange additional information. The functions for this Vehicle-to-Infrastructure communication are located above the Ethernet transport protocol and can therefore also be combined with existing OEM-specific software architectures. In this way, the vSCC component can be used without it being essential to switch over the entire communication to AUTOSAR. The MICROSAR implementation contains the complete protocols for smart charging. They are configured via an AUTOSAR tool and are linked to other necessary components.
At the physical level, special transceivers are needed for Powerline Communication. These are responsible for processing the high-level signal while, at the same time, a digital input for low-level communication is connected to the microcontroller. In this case, the standards predefine precise values for the edge steepness of the signals. Due to the high-frequency transfer rate at the Control Pilot, it is necessary to pay particular attention to the EMC radiation when designing the board for a charging ECU.

Charging ECUs for Commercial Vehicles

The first charging ECUs for commercial vehicles are already available on the market. Vector's VC36PLC-24 (**Figure 5**) implements Powerline Communication between the vehicle and charg spot. It is particularly suitable for small series productions as well as for prototype constructions and developments. The ECU assumes a range of functions such as, for example, locking the plug at the inlet, plug temperature monitoring, or the display of the charging status by means of LEDs. The hardware platform of the VC36PLC-24 can be used independently of vehicle manufacturer and is suitable for cabled (conductive) charging using the CCS-2 standard. The ECU can be supplied with a software architecture that corresponds to the vehicle manufacturer or as a generic ECU for prototypes or development projects. This version implements both the low-level and high-level communication. This means that the ECU can be easily installed in any vehicle. The open, CAN-based J1939 interface for network connection is available to users without restrictions.

Figure 5: The Vector ECU VC36PLC-24 offers an interface for standardized communication of electrically powered commercial vehicles with the charging infrastructure. It supports DIN SPEC 70121 and ISO/IEC 15118.

Charging via Pantographs – The Better Plug?

The current standard ISO/IEC 15118 focuses primarily on cabled charging at a charging station. Here, drivers must do a number of things themselves in order to establish the connection between the vehicle and the infrastructure. In this case, it makes no difference whether the vehicle is a car or a commercial vehicle. However, this standard also supports electric charging via a pantograph located on the vehicle. To do this, the driver presses a switch to connect the pantograph to the charging mast. However, pantographs of this type take up space on the bus and have to be transported by the vehicle, increasing its weight and consequently also its power consumption. One solution here takes the form of the so-called inverted pantograph, i.e. a system in which the main element is located in the fixed infrastructure and the vehicle is simply equipped with contacts. It is precisely this type of application that is currently undergoing specification in ISO/IEC 15118 Part 2, Edition 2 and it should be available as an international standard in mid-2018. In this case, the connection between the two partners is established fully automatically by means of wireless communication. The WiFi protocol 802.11n is used for this.

Outlook

Alongside the minimization of emissions, the interoperability of all vehicles with the charging infrastructures of the various suppliers is an important element in ensuring the rapid success of electromobility. Every vehicle can "refuel" with electricity at every available charging station – just as easily as drivers now fill up at modern gas stations. The aim should be for charging standards to support and not slow down this change. The standards have now been defined and accepted by the manufacturers. The market already offers suitable solutions. Industry can get to grips with the challenges without reservations and make use of the available components.

B.Eng. Jonas Leserer
Business Development Manager for Embedded Software and Systems at Vector Informatik and responsible for small series ECUs.

Translation of a German publication in Elektronik Praxis, special issue Embedded Software Engineering, 11/2017

Image rights:
Vector Informatik GmbH
The World Factbook (edited by Vector Informatik GmbH)

Literature References:

[1] Press release Deutsche Post DHL (2014-12-09)

Inductive Charging – From Evaluation to Standardized E-Mobility

Enhancements and additions to Edition 2 of ISO/IEC 15118

Alongside wire-conducted charging, inductive charging will play an important role in the future of electromobility. The automotive industry is now converting the results from the evaluation phase of ongoing development projects into specific products. A considerable part of the associated development effort is going toward wireless communication between the vehicle and the charging infrastructure. The standards development activities for this are in full swing and essential functions are being added to the existing ISO/IEC 15118 standard. For easy development of vehicle and charging infrastructure electronics, both embedded solutions and tools for simulating the respective counterpart electronics during subsequent testing are available.

The standard charging system for wire-conducted charging of electric and plug-in hybrid vehicles is the Combined Charging System (CCS), which is mandatory for all future fast charging systems in the European Union. All charging systems in Germany must therefore be installed according to the CCS standard. CCS uses the Combo2 connector in Europe, which allows both AC and DC charging. The digital communication between the electric vehicle and the charging infrastructure in the CCS via HomePlug Green PHY™ runs over the control pilot wire and uses the SLAC mechanism (Signal Level Attenuation Characterization) for establishing the connection. Since 2015, the CharIN e.V. (www.charinev.org) has served as an official committee for promoting the CCS standard and for decision-making regarding further development.

"Booster" of E-Mobility: Inductive Charging

Alongside the charging of electric vehicles via a wire, wireless charging (WPT – Wireless Power Transfer) will become an absolutely essential component of E-Mobility in the future. Automobile manufacturers are currently developing the first series generation of electric vehicles that support inductive charging. This will make vehicle charging much more convenient since the vehicle can recharge energy automatically regardless of whether it is at home or out and about. Correct positioning, connection establishment, and authentication are required for this. As the vehicle approaches the garage at home, for example, it is connected to the charging infrastructure there via Wi-Fi. A positioning aid enables the incoming vehicle to be directed to the correct position over the charging pad (ground pad) after which the charging operation starts immediately. The driver no longer has to walk around the car, open the trunk, pull out the charging cable, and connect it to the charging system. Likewise, it is no longer necessary for drivers to unplug the cable and stow it in the morning when they are in a hurry.

For charging in public areas, services for reserving charging parking spaces will be available along with control systems that guide vehicles to available charging spaces. All of these convenience features, including automatic billing and payment transactions, will increase the acceptance of E-Mobility sustainably.

One of the most important prerequisites for uniform, compatible, and vendor-neutral implementation of this charging technology is the establishment of an international standard. This applies in particular to the communication. In addition to wireless power transfer, the information exchange between the vehicle and the ground pad must also take place wirelessly. Various ISO/IEC 15118 working groups are working on this task and are in process of integrating WPT and other interesting functions in the standard.

Additions in the Second Edition of the ISO/IEC 15118 Standard

Besides Wireless Power Transfer, the second edition of ISO/IEC 15118 will contain important new additions including energy regeneration and the Autoconnect Charging Device (ACD). The documents for ISO/IEC 15118-6 (Applications) and -7 (Network and Protocols) originally intended for the second edition will now not be stand-alone parts of the standard but will rather be incorporated into ISO/IEC 15118-1 and -2. Only ISO/IEC 15118-8 is being added to describe the Physical Layer and Data Link Layer of the Wi-Fi communication during charging.

With the possibility for energy regeneration, electric vehicles can now act as supports for intelligent power networks. This is likely to be feasible first in applications such as smart homes **(Figure 1)**, while connection to public networks (smart grid) is probably still a ways off. The focus is on balancing differences between energy supply and demand. The increasing proportion of regenerative energy sources will naturally lead to greater variations. Owners of electric vehicles are thus able to temporarily store energy or recover a portion of their energy costs by charging opportunely and supplying some of the energy back during expensive peak charging times (V2G – Vehicle-to-Grid). The standard explicitly defines another use case that might be of particular interest in Asia. Accordingly, electric vehicles can serve as an emergency power supply for households when public power supplies fail due to natural catastrophes or technical problems (V2H – Vehicle-to-Home). Finally, V2V functionality (Vehicle-to-Vehicle) gives electric vehicles the capability to perform roadside assistance. Stalled vehicles can thus be charged to the extent needed for them to reach the next charging station.

Figure 1: In a smart home, an electric vehicle will not only be charged in the future but will also feed power to its own charging station in return.

The new feature named ACD (Autoconnect Charging Device) targets operation of electric buses used in local public transit. For example, it allows charging at bus stops with high charging powers of 150 to 500 kW via a pantograph/DC busbar **(Figure 2)**. Wi-Fi is provided for communication so that the bus establishes the connection, performs a compatibility check, and receives positioning signals in a timely way while it is approaching the bus stop.

The wireless communication introduced with inductive charging is available not just for charging electric buses but is an option for cable charging as well.

Figure 2: Autoconnect Charging Device (ACD) is aimed at operation of electric buses used in local public transit. For example, it allows charging at bus stops with high charging powers via a pantograph/DC busbar.

Wi-Fi with TLS Encryption

A dedicated frequency band for E-Mobility cannot be implemented internationally. However, the frequencies specified in ISO/IEC 15118 must be usable worldwide. For this reason, the ISO committee chose Wi-Fi in the IEEE802.11n specification as the physical transmission medium for wireless communication. IEEE802.11n provides dual-band sup-

port for 2.4 GHz and 5 GHz and provides greater ranges and higher transmission rates than, for example, Bluetooth LE (Low Energy), which was also discussed. Further criteria for the wireless connection are low-cost characteristics, high reliability, and a maximum latency of 100 ms. A higher latency would cause frustrating delays in updating the direction indicators on the display during fine positioning.
The use of open wireless networks for charging communication or for billing and payment purposes leads directly to the issue of information security. A number of market participants are therefore calling for encrypted communication for the next edition of ISO/IEC 15118. The latest version (Version 1.2) of the Transport Layer Security (TLS) encryption method is well suited for this. TLS is the follower of SSL (Secure Sockets Layer) and protects privacy during charging communication, prevents data manipulation, and ensures authenticity of the charging station. A combination of various cryptography methods would be used in the proposed encryption. In detail, a so-called asymmetric key exchange takes place based on ECDH (Elliptic Curve Diffie Hellman) or ECDHE (Elliptic Curve Diffie Hellman Ephemeral). Only afterwards, is it possible to apply the symmetrical AES128 method for the actual encrypted data transmission as well as the hash algorithm SHA256 for checksums.

Fast Developing and Testing

The primary goal of standardization is cross-vendor interoperability, which means that every electric vehicle can ideally be charged on any charging system. To ensure this, vehicle and infrastructure manufacturers regularly participate in "Testivals" in which representatives of the respective companies test their ECUs, vehicles, and charging stations with one another over a 2-day period. Of course, engineers test their products in-house beforehand. Testing can quickly become a complex matter, especially since Parts 4 and 5 of the standard that describe the test modalities for conformance tests are not yet finished. Ideally, a suitable tool for simulating the respective counterpart is available, for example, from Vector, for testing in the lab. While developers of vehicle electronics need simulation of the charging station, manufacturers of charging infrastructure are dependent on vehicle simulations.
Electric vehicle manufacturers and charging infrastructure suppliers quickly come up with functional communication solutions when they use proven embedded modules. For example, Vector actively participates in the ISO standards development process and is able to promptly adapt its solutions to the current state of the standard. This is also the case for inductive charging. While MICROSAR.V2G is intended for charging ECUs in the vehicle **(Figure 3)**, vEVSE implements the ISO/IEC 15118-compliant counterpart electronics on the grid side. Tests can be performed during different phases of development using a model-based approach in which the test scope and hardware interface can be customized to customer requirements. The Vector test system can be configured in such a way that the vehicle or charging infrastructure is simulated, depending on what is to be tested. The tests optimally cover Vehicle-to-Grid communication according to ISO/IEC 15118-2, the SLAC mechanism (Signal Level Attenuation Characterization) according to ISO 15118-3, and the DIN-SPEC-70121 (DC charging). As soon as the test documents (Parts 4 and 5 of the standard) are available in final form, expected by early 2017, Vector will integrate a corresponding conformance test suite in its solutions.

Figure3: Integration of MICROSAR.V2G modules in an AUTOSAR architecture

Dirk Großmann
He is Senior Manager at Vector in Stuttgart/Germany where he is responsible for, among other things, development of the Vector E-Mobility solution. He studied electrical engineering and gained six years of automotive development experience in the area of ECU software before coming to Vector in 2003.

Translation of a German publication in Hanser automotive Special edition E-Mobility, issue 9-10/2016

Inductive Charging Gives Trigger to Future of E-Mobility

ISO/IEC-15118 Standardization of Wireless Power Transfer

In discussions and reports about electric mobility, the topic of inductive charging continues to grow in importance. Along with its primary benefit of convenience, this technology offers a number of key long-term advantages that cannot be overestimated. In particular, inductive charging makes it possible to conveniently extend the driving ranges of electric vehicles and design the expensive batteries to be smaller.
This article offers an overview of current issues related to the technology and the standardization of inductive charging as discussed in such forums as the 2015 Vector E-Mobility Engineering Day in Stuttgart, Germany, which was attended by automotive OEMs and suppliers.

Just as vehicles with a combustion engine make regular trips to the filling station, electric cars must regularly draw their energy from an electrical outlet – whether they are charged overnight by plugging it into an electrical outlet in a garage at home or on the road at an accessible public charging station. To date, there are no standardized methods that do not involve wire-conducted power transfer, which is always associated with manual actions for coupling plugs to a charging socket.

Inductive Charging Technology: Cutting-Edge Trend in E-Mobility

This will change in the future, because E-Mobility offers an alternative and supplement to wire-conducted charging in the form of the advanced option of inductive charging. A distinction is made here between stationary and dynamic charging. This means that along with usual charging while the vehicle is stationary, it is also technically feasible to perform dynamic charging while driving on specially prepared roadways. Numerous businesses, institutes and research projects are already working on advancing the technology of inductive charging to market maturity while simultaneously achieving global standardization. Charging without a wire connection leads to perceptible convenience benefits in everyday driving for users of electrically powered vehicles. Many other benefits are also realized: wireless charging avoids defects in plug connections and cables due to aging, wear, corrosion or wire breakage, and this leads to lower overall maintenance effort. In the public domain, the lack vulnerable points of attack in the infrastructure also means significantly less damage can be expected from vandalism.

In conventional charging stations **(Figure 1)**, a user interface is generally housed in the charging station along with the charging equipment and charging cable. This interface might include a display module, controls for user inputs and a credit card reader. In inductive charging, on the other hand, not only is a charging cable no longer necessary; this type of stationary user interface is no longer necessary either, because inductive charging is based on an operating philosophy that differs fundamentally. The drivers themselves decide, from inside the vehicle, whether to draw energy while underway, and they program their on-board electronics according to their wishes. The exchange of information between the vehicle and the inductive charging infrastructure is automated. When driving into a parking structure, for instance, a guidance system could guide the car to an available parking space that offers stationary charging. While driving the car, dynamic charging is automatically started when passing through a section of roadway designed for this purpose.

Pilot Projects Pave the Way

An Internet search will reveal that inductive charging is by no means new; rather it has existed for many years in the form of pilot projects and has been used in certain special applications. One area of interest is driverless transport systems, which have influenced processes in factory halls and warehouses for many years. This technology is also used to charge public electric buses in Turin and Genoa. In these two Italian cities, around 30 electric buses have been operating in traffic since the year 2002. They are able to "refuel" by quick charging with power from inductive charging devices embedded in the road pavement when they stop at bus stops **(Figure 2)**. This method lets the buses complete their daily routes effortlessly without having to explicitly charge their batteries during their operating hours or make extra trips to battery exchange stations.

Figure 1: Today's conventional charging via charging station and cable

Since the vehicles do not need to carry enough energy to last the entire day, developers were able to size the battery capacity to be significantly lower, thereby reducing battery weight and size. This also leads to smaller moving masses and makes them economical. Today, comparable projects based on inductive charging technology are being implemented in local public transportation systems in many cities worldwide. In Germany, for example, there are projects in Augsburg, Braunschweig, Mannheim and, starting in the summer of 2015, in Berlin as well.

New Approaches to an Old Transformer Principle

Similar to charging at bus stops, it would be feasible to charge private electric vehicles at stop lights, at least in the urban environment. However, many challenges would have to be overcome – with regard to both the technology and standardization – before such a scenario could be implemented.
Inductive charging is based on coupling between two coils – which is a principle from the classic transformer that has been known for a long time. Since a high level of efficiency is one of the key factors for success, one criterion for an implementation is that it should operate with a minimal distance between the primary coil in the roadway pavement and the secondary coil in the vehicle. Another important criterion is precise positioning of the vehicle in order to minimize stray losses. Assistance systems for fine positioning will be of valuable service here.

Overcoming Technical Challenges

Other technical issues relate to choosing suitable frequency ranges, charging current levels and coil voltages. These and many other parameters determine maximum charging power, efficiency levels, allowable heating and other per-

Figure 2: Inductive quick-charging of an electric bus in Turin, Italy – markings make it easier to position the vehicle precisely over the charging unit installed in the pavement

formance data. Furthermore, various physical effects, parasitic capacitances, isolation losses and many others must be considered. Despite all of the challenges, known projects on inductive charging are achieving efficiency figures of around 90 percent. While the Italian pilot projects that were installed in 2002 operate with a charging power of 60 kW, electric buses that went into operation in Braunschweig in 2014 obtain their energy at a power level of 200 kW. Both systems utilize a frequency of 20 kHz.

Charging Communication via ISO/IEC 15118: Standard for Wire-Conducted Charging

Any form of (public) inductive charging would be inconceivable without reliable communication between the vehicle and the charging infrastructure. That is the only way to answer questions of central importance – namely, who obtained how much energy when and where and under what conditions and to generate correct, legal billing documents. It should also be able to prevent fraud and misuse. The same requirements apply to charging via cable at charging stations. For wire-conducted charging (AC and DC), there is already the ISO/IEC 15118 standard that was fully released in early 2015. There is also the DIN SPEC 70121:2014 standard for DC charging. Together they represent parts the Combined Charging System (CCS – combined charging system), which is the standard for AC/DC charging systems in Europe.

The ISO/IEC 15118 standard specifies all details related to charging communication for intelligent charging or smart charging, along with establishing a connection and authentication, negotiation of price rates, automatic payment, charging management and considerations for future smart grid requirements. If electrical grid power is in low supply, charging management might have the electric vehicle temporarily feed energy back into the grid. ISO/IEC 15118 "HomePlug Green PHYTM" is planned as the physical layer for data communication.

ISO/IC 15118 Extensions for Wireless Charging

Of course, it does not make sense to use wire-based communications technology in inductive charging. Therefore, the physical layer (ISO/OSI Layer 1) must be changed. Moreover, it is necessary to modify the application layer (ISO/OSI Layer 7). Many ISO/IEC project teams are currently dedicated to these tasks and are developing the ISO/IEC 15118 standard for Wireless Power Transfer (WPT) further. Specifically, ISO/IEC 15118-6, -7 and -8 parts represent extensions of ISO/IEC 15118 for WPT. The draft for Part 6 of the standard addresses general information on wireless communication with WPT and defines application cases, while Part 7 addresses protocol requirements. Finally, ISO/IEC 15118-8 specifies the physical layer as well as the data link layer.

In addition to ISO/IEC, other standardization organizations are also working on communication solutions for WPT, in particular SAE. This means that for every part of ISO/IEC 15118, there is a comparable part in the SAE standard. China, however, has not yet announced any national Guobiao (GB) standard related to WPT. Nonetheless, harmonization of all national and international standards is necessary to achieve global compatibility of the charging technology and charging communication. Organizations relevant to North America and Europe, SAE and ISO/IEC, are in the process of creating a common standard for Wireless Power Transfer. Now it is necessary to get Asia on board, especially China.

A draft international standard (DIS) for ISO/IEC 15118-6 was released in early 2015, and DIS Part 7 and DIS Part 8 should follow by late 2015 or early 2016. A complete Final Draft International Standard (FDIS) is planned for early 2017.

Embedded Solutions and Test Tools for ISO/IEC-15118

Automotive OEMs, suppliers and relevant businesses do not, however, have to wait until then to develop the charging infrastructure. Because inductive charging per ISO/IEC 15118-6ff is in large measure identical to wire-conducted charging, a stepwise approach is recommended. Finished and tested embedded solutions for ISO/IEC 15118 are already available today from Vector, for example. Manufacturers of production ECUs can use them to develop intelligent charging by wire. While the MICROSAR.V2G product is used to develop charging ECUs in the vehicle, the vEVSE product implements the related ISO/IEC-15118 conformant communication for charging stations. The solutions support all software requirements of ISO/IEC 15118, including the hardware interface (HomePlug Green PHYTM) and SLAC (Signal Level Attenuation Characterization) process. The second step will be to perform migration to inductive charging as soon as the ISO standard has prescribed the necessary specifications for inductive charging systems. Because Vector is represented on ISO subcommittees and is actively involved in standardization, the Stuttgart-based specialist is able to offer solutions and extensions for WPT in a very timely way. The same applies to test tools from Vector that are used to test charging systems. For example, the company offers a special VT7900 rack card with a VT7870 adapter **(Figure 3)** for the VT System HiL tester. It can be used to support all of the mechanisms needed for Smart Charging Communication (SCC). They include HomePlug GreenPhy communication with SLAC, PWM control (IEC61851) and suitable resistors for simulating plug connections. The VT System can be used to simulate electric vehicles or charging stations.

Figure 3: VT7870 is the Smart Charge Communication module from Vector for ISO/IEC-15118 conformant tests of charging communication

Summary and Outlook

Global standardization is of tremendous importance in achieving a breakthrough in electric mobility and tapping the mass market, and this applies equally to both conductive and inductive charging. A uniform global market lets the automotive industry and the producers of charging infrastructure sell their products internationally. It eliminates the need to implement special solutions for each part of the world that drive costs upward. At the same time, this would increase the resolve to make further investments in E-Mobility and offer assurance that the enormous development costs would pay off. If electric vehicles could be implemented internationally – and a standardized charging infrastructure with broad coverage were to exist that included wireless charging – this would have a positive effect on user acceptance.

In practice, wire-based and inductive charging technologies will presumably complement one another. What is important is to first implement nation-wide coverage of basic infrastructure for wire-based charging. Inductive charging, on the other hand, offers a lot of potential for the future, but it requires substantial infrastructure modifications. After international harmonization efforts have been completed, it will become apparent just how willing investors are to support inductive charging infrastructure projects.

Dynamic charging solutions would let electric vehicles draw energy wirelessly while driving on specially designed sections of roadway. Over the long term, the positive effects of this approach can hardly be overstated. After all, this would significantly extend the driving ranges of electric cars, which are currently still limited. This might be accomplished by integrating charging segments at regular intervals on expressways or intercity highways. Today, there are already many places in Germany that have windmill generating facilities which are near expressways, and they could supply electrical power to sections of roadway with inductive charging capabilities at a reasonable cost.

Translation of a German translation published in emobilitytec, issue 4/2015.

Image rights:
Figure 0, 1, 3: Vector Informatik GmbH
Figure 2: IPT Technology

Dirk Großmann
Is a group leader in the area of Embedded Software at Vector Informatik. In his position, he manages such activities as the development of the Vector E-Mobility Solution. After obtaining a degree in electrical engineering from the University of Stuttgart and working in automotive ECU software development for 6 years, he joined Vector Informatik in 2003.

Smart Testing of Smart Charging

Consistent Test Case Coverage for Electric Mobility

With the increasing diversity of electric vehicles and charging station systems, interoperability between components and conformance to standards are ever more important. In order to test these issues, identify causes of charging interruptions, and test reliability and robustness against various disturbances, consistent test case coverage with an open test environment is required.

The Charging Process According to IEC 61851 and IEC/ISO15118

Four modes are defined in IEC 61851 [4] for conductive charging of electric vehicles.

> In Mode 1, the vehicle is charged with a single-phase power supply with a maximum current of 16 A without a pilot signal.

> In Mode 2, the charging operation uses a single-phase to three-phase power supply with a maximum current of 32 A and a pilot signal.

> In Mode 3, charging takes place with a single-phase to three-phase power supply with a maximum current of 63 A and a pilot signal provided by the charging station.

> Mode 4 describes DC charging with up to 400 V / 125 A.

While the Mode 1 charging operation does not involve communication between the vehicle and the charging infrastructure, charging in Modes 2, 3, and 4 always involves low-level communication based on pulse width modulation (PWM) via the Control Pilot (CP) connection. If the vehicle and charging station support high-level communication, the corresponding signal is modulated to the PWM signal of the CP based on the HomePlug GreenPHY standard. This is defined as power line communication (PLC) and is only possible in Modes 3 and 4 as described by IEC/ISO15118 [3]. Every PLC-based charging communication requiresis PWM-based communication, in principle. Therefore, a complete test system must handle both communication modes.

Function of the Components Involved

From the vehicle perspective, the functionality of the charging station essentially consists of the components described in **Table 1. Table 2** presents the functionality of the vehicle from the perspective of the charging station.

Charging station functionality	Realized through
Communication signal on the CP	Frequency signal based on pulse width modulation (PWM). The PLC signal is modulated to the PWM signal, if necessary
Maximum possible loading of the charging line	So-called proximity signal (PP), which is coded by a resistance between PP and PE dependent on the cable cross-section in the connector
Provided current	Coding by the duty cycle of the PWM signal
High-level communication capability	Duty cycle of 5%. The duty cycle in the range 3% – 7% is reserved exclusively for this purpose
Participant in PLC communication	Communication of more complex information, e.g., charging profiles, billing models, authentication
Power	1- to 3-phase AC circuit, between 100 V and 240 V AC, depending on country. Alternatively: DC voltage via separate cables

Table 1: Functionality of the charging station that the vehicle recognizes directly or indirectly.

Vehicle functionality	Realized through
Display of the plug connection	Resistance between CP and PE that lowers the CP level from 12 V to 9 V
Display of the connector interlock	Additional resistance between CP and PE that lowers the PWM signal from 9 V to 6 V
Display of required cooling	Resistance between CP and PE that lowers the PWM signal from 6 V to 3 V
Participation in PLC communication	Communication of more complex information, e.g., charging request, account data, authentication
Power reduction	Agreement of a profile by means of low-or high-level communication

Table 2: Functionality of the vehicle that is visible for the charging.

Test of the Charging Communication

In order to prove that charging is functioning properly at all levels, a special measuring and testing system is needed. It must be able to analyze and emulate both the load circuit and the communication signals, such as the Control Pilot and the PLC signal. In addition, creation, display, and if necessary manipulation of the message contents of the PLC are required. The ability to inject errors into the electric signals completes the test system. This includes both internal short circuits and short circuits to battery voltage/ground as well as variations of the resistances in the vehicle and charging station that take part in the CP communication. Finally, a complete test system must be able to measure and analyze the load circuit (AC or DC) in terms of current and voltage, and disturb it in a defined way.

Test Modes and Requirements for a Test System

For component tests and analyses of robustness, a test system must provide all interface variables in conformance with standards and with defined injected errors to each component to be tested.

For the charging communication channels shown in **Figure 1**, the possible issues to test include:

> Control Pilot signal with incorrect characteristics on the PW component (e.g., level, resistance values, duty cycles, time sequences) and PLC component (e.g., structure of communication, communication conforming to standards, and defined faulty messages)

> Proximity Pilot including incorrect coding resistance

For the component test or robustness test of an EV charging electronic control unit (EV-ECU) with simulated EVSE **(Figure 2)**, the following must also be provided:

> Vehicle wiring system voltage with overvoltage/undervoltage, ramp characteristics, erratic disturbances

> Vehicle communication (e.g., CAN) with message errors and electrical errors

> AC or DC load signal with all types of voltage disturbances

> Grid Simulator for emulation of various grids around the world (different voltages, frequencies, mains with disturbances, etc.)

> Consumption of the DC charging power, if necessary

> For an EVSE test, the vehicle is simulated with all limit values **(Figure 2)**. Also required are:

> Connection to a real or emulated power utility system (gridemulator)

> Simulation of the www environment, if necessary

For execution of interoperability tests, the test system is inserted between the charging station and the vehicle **(Figure 3)**.

Figure 1: Interaction of charging ECU (OBC – Onboard Charger) and charging station (EVSE).

There are two possible modes of operation: in pure passive mode, only measurements are possible in the electrical signals of CP, PP, and load circuit. In pure gateway mode, where analysis and manipulation of the PLC messages in high-level communication mode is targeted, it is necessary to disconnect, intercept, and modify as necessary the CP line since the messages exchanged are encrypted when the vehicle-charging station connection is established. In addition to measuring and influencing the communication and load channels, a test system must have an execution unit, recording unit, monitoring unit and ideally an authoring and managing unit for test cases. There is already a large demand for systematic tests [2] today. In order to address this issue, IEC/ISO15118 [3] will provide Parts 4 and 5 for standardized test cases in the future.

Overall Test System

To cope with the complex demands on an overall test system, a modular test system architecture was selected that meets all necessary requirements, while allowing realization of subtasks through targeted reduction of the system setup **(Figure 4)**.

The central software element is the CANoe tool of Vector Informatik GmbH with Ethernet option. CANoe takes over the following tasks: test execution, simulation of PLC communication, provision of measurement and analysis data of the vehicle and charging communication and all electrical characteristics, and control of the hardware components involved. It enables all relevant measurements and signals to be provided with uniform time stamps and to be processed, evaluated, and saved. The creation and management of test cases, whether user-created [2] or standardized, are completed in vTESTstudio of Vector Informatik [7]. The presented interconnection options enable test systems for subtasks to be derived from this architecture.

For simulation of the environment of the charging ECU and/or charging station, the presented system uses a combination of the EV Charging Analyzer / Simulator (EVCA) of comemso GmbH [5] as well as the VT System of Vector Informatik [6]. Both the EVCA and the VT System are able to uncouple the PLC signal of the high-level communication from the CP signal and to route CANoe over Ethernet. With the help of the Smart Charging Communication (SCC) add-ons, all necessary analysis and manipulation possibilities are available in CANoe [1].

The EVCA and the VT7870 module of the VT System provide the circuitry for CP and PP required according to IEC 61851-1 [4], including the possibility to inject errors **(Figure 5)**.

Figure 4: Interaction of charging ECU (OBC – Onboard Charger) and charging station (EVSE). overlay on video image in the Multi-media Window.

The VT System also represents the interface for vehicle communication and enables simulation of the vehicle power system and, if necessary, the control of an electronic load or external power supply. The EVCA of comemso GmbH is also able to detect all disturbances of the complex CP and load signals in real-time while simultaneously detecting the sources of the disturbances. Because the pilot signal together with the modulated PLC signal can violate the specifications for the pilot signal in the standard, specially developed analog filter circuits are required to achieve interaction-free separation of the high-frequency PLC signal from the pilot signal. For analysis of the CP signal, the EVCA uses a specially developed measuring procedure in which up to 150 different errors and permutations thereof can be measured in each period.

In addition, the current and voltage of the load circuit is measured precisely as true RMS on all three phases with the EVCA. Additional power supply system analysis for each of the three phases and each sine-wave period identifies asymmetrical mains utilizations as well as mains disturbances that can cause electromagnetic interferences of the communication signal.

While charging communication is based on Ethernet, all variables measured with the EVCA are forwarded to CANoe via a separate CAN channel. Only through the common and time-synchronized measurement and visualization of the load circuit, CP signal, and PLC signal, related disturbances and – if necessary – causes and effects can be identified.

For field use, the mobile version of the comemso EV Charging Analyzer / Simulator (EVCA) and Vector CANoe. Ethernet with vTESTstudio are used **(Figure 6)**. This provides test case consistency from the prototype stage to the series product both in the lab and in the field.

Figure 2: Possible component tests.

Figure 3: Possible modes of a "Man-in-the-middle" operation for interoperability tests.

Figure 5: Test system components for SCC tests relating to IEC 61851 and IEC/ISO 15118.

Figure 6: Test system combination for mobile use.

Summary

The test system architecture presented in this article enables systematic testing of components involved in conductive charging according to IEC 61851-1 [4] and IEC/ISO 15118 [3] for robustness, conformance to standards, and interoperability. The open vTESTstudio and CANoe test environment provides fundamental transparency and reproducibility of the test sequences. The modular architecture and the combination of the comemso and Vector tools allow operation with the real environment or a fully simulated environment and any intermediate stage in between as well as operation in automated test cycles. With this analytical approach at all levels, maximum test depth is made possible – the cornerstone for robust and harmonized components that will ultimately satisfy E-mobility customers.

Translation of a German publication in EmobilityTec, December 2014.

Literature:

[1] Albers, T. (July 2011). Komfortables Laden. Elektronik automotive – Sonderheft Elektromobilität, p. 60-63.

[2] Brosi, F., Reuss, H.-C., & Großmann, D. (2013). Communication Conformity Between Electrical Vehicles and Charging Equipment Pursuant to ISO 15118. 13th Stuttgart International Symposium Automotive and Engine Technology. Vol. 2, p. 115-126. Stuttgart: ATZlive.

[3] IEC/ISO15118. 2012. Road Vehicles – Vehicle to grid Communication Interface.

[4] IEC61851-1. 2010. Electric Vehicle Conductive Charging System – Part 1: General Requirements.

[5] EVCA comemso GmbH. 2013. EV Charging Analyzer / Simulator. Retrieved on 2014-08-01 from http://www.comemso.de/index.php/de/produkte/ladesystem-analysator

[6] VT System Vector Informatik GmbH. 08 2014 Modular Test Hardware: VT System Retrieved 08 2014 from http://www.vector.com/vi_vt-system_de.html

[7] vTESTstudio Vector Informatik GmbH. (2014). Vector. Retrieved on 2014-08-07 from http://vector.com/vi_vteststudio_de.html

Image rights:

Vector Informatik GmbH

Links:

Homepage Vector: www.vector.com

Homepage comemso GmbH: www.comemso.com

Dr. Kiriakos Athanasas
is Managing Director of comemso GmbH. He studied communications engineering at HS Esslingen and later studied information technology/electronics. He received a doctorate in distributed computer networks in the area of system safety and functional safety of autonomous driver assistance systems and driving dynamics systems. After 12 years of experience in these areas at Daimler Research & Development, he founded comemso GmbH in 2009. comemso GmbH develops complex measurement, testing, and electronic systems for e-mobility.

Dr. Heiner Hild
is Team Leader and Product Manager for the VT System at Vector Informatik, where he is responsible for development of the VT System and other I/O test hardware and their linkage to CANoe. He studied physics at the University of Stuttgart and received a doctorate in image processing and geographic information systems. Before joining Vector Informatik GmbH in 2014, he worked over 10 years in automotive development on driver assistance systems.

CANoe
Safety-Related Lifecycle Validation of Battery Systems

Case Study
ZSW Laboratory for Battery Technology (eLaB)

The Customer

The ZSW Laboratory for Battery Technology (eLaB) in Ulm, Germany is an independent development and testing center for batteries. It is internationally recognized for its testing of cells, modules and complete battery systems including battery management. The laboratory's researchers have over 25 years of experience with batteries and supercapacitors. Customers come from industry and research organizations worldwide.

The Challenge

Safety-related lifecycle validation of battery systems with increasing numbers of variants

Endurance tests, which typically take months to complete, represent the foundation for reliable operation of a Battery Management System (BMS) and for long battery life. The tests contain real and synthetic performance profiles, such as emergency shutoff and charge cycles. The challenges are to strive for full automation in test flow and to handle the large numbers of test variants and large volumes of data. The CAN communication data alone amounts to terabytes.

The Solution

Combining an active power controller and CANoe to create a high-performance endurance test bench for batteries

All power and communication interfaces of the System Under Test (SUT) are implemented in the battery test bench, which consists of the Battery Management System (BMS) and the battery (see figure). An active power controller can act as either a DC source or sink. Flow control in PC1 executes power profiles fully automatically. The CANoe test tool on the separate PC2 models CAN and LIN networks in a remaining bus simulation. The two systems communicate directly via the already existing vehicle bus. Visualization is provided on CANoe panels for continuous monitoring of the test progress. CAN-driven, CANoe subdivides the measurement results into binary log files of reasonable size and manages them. They can be traced back to their associated endurance test at any time.

The Advantages

Endurance logging, data management and visualization for continual access to everything "at a glance"

> Endurance tests of the battery and the BMS are very realistic
> Non-stop operation of the test bench of over half a year ensures smooth test flow
> Transparency achieved by clear traceability of log files to driving cycles and power profiles – both during operation and offline
> Many different logging strategies based on user-definable trigger conditions
> Graphic visualization makes key battery parameters immediately and continually accessible at a glance
> Targeted offline analysis is enabled by versatile and convenient search functions

Smart Charging – A Key to Successful E-Mobility

Development and Testing of Charging Electronics Conforming to Standards

Electric vehicle charging with AC current has been governed by the ISO/IEC 15118 international standard since it was fully adopted in spring 2014. For DC current charging, the industry has agreed on an applicable subset, which is described in DIN SPEC 70121 (Prestandard procedure). This article sheds light on some details of the two standards and points out ways that manufacturers of electric vehicles and charging stations can quickly develop and effectively test products conforming to the new standard.

Fueling a combustion engine vehicle is not very spectacular and is over within a few minutes. The corresponding operation for electric vehicles is quite another matter. Managing and optimizing the process of charging the battery is one of the primary tasks of vehicle and charging station manufacturers. In the context of the changeover to alternative energy sources and the future smart grid, the charging operation is much more complex than it might appear at first glance. It is necessary to consider not only basic information such as the power available at the charging station, the battery condition and the charging demand. However, more complex information, such as the energy resources available at the time of charging, must also be taken into account. Also important are aspects related to cost optimization, payment/billing modalities, and safety, as well as other questions, such as: How is the user authenticated? Does the vehicle need as much electrical energy as possible in a short amount of time or can the charging operation be extended over several hours until after work or the end of an extended shopping trip?

Smart charging according to ISO/IEC 15118 offers the opportunity to address these challenges in the best possible way in terms of vehicle owners, energy suppliers, charging station manufacturers, and last but not least, the environment. Uncontrolled charging processes in which users simply connect their vehicles to the available power supply system in order to charge them with maximum possible power will soon become a thing of the past or will be available only at a premium. Since electric vehicle charging re-quires relatively large amounts of power, locations such as parking garages that have a high demand factor can experience a significant power demand. Intelligent charging processes take this into consideration and grant the power supply company or smart grid the time needed to adjust to the changing load conditions. Blackouts during peak load times can be reliably prevented in this way.

HomePlug Green PHY: a Single Cable for Energy and Data

A basic prerequisite for smart charging is reliable information exchange between the vehicle and charging station. In this regard, ISO/IEC 15118 provides for HomePlug Green PHY as the physical layer. This is a PLC (Power Line Communication) method derived from HomePlug AV. The latter is widely used in homes for networking computers and audio/video components via a home's existing electrical wiring. Well-known Internet protocols from the IT world, such as TCP, UDP, IPv6, DNS, DHCP, are TLS, are all built on this. The initial establishment of a connection for the power line communication is described in more detail in the following.

SLAC Ensures Reliable Establishment of a Connection

Within the scope of high-level communication according to ISO/IEC 15118, the initial establishment of a connection between the vehicle and charging station is a critical event. As soon as a driver connects his/her vehicle to a charging station, the initial objective is to join a secured AV Logical Network (AVLN). Each AVLN is represented by a Network ID (NID). In addition, each node of an AVLN requires an NID-matching network membership key for encrypted communication.

Figure 1: MICROSAR V2G contains the software modules for Vehicle-to-Grid communication.

At the beginning of communication establishment, the vehicle sends an Ethernet broadcast message to all possible network nodes. All charging stations that receive this signal respond with an Ethernet unicast confirmation. Since multiple charging stations can be supplied by one physical supply cable, the protocol is responsible for identifying the relevant charging station to which the vehicle is connected. Unfortunately, other charging stations not physically connected to the vehicle can also respond due to undesired coupled-in noise. The so-called SLAC mechanism (Signal Level Attenuation Characterization) of HomePlug Green PHY ensures a correct and reliable process for identifying the physically connected charging station. SLAC works according to the request/response method in which the request always comes from the electric vehicle. At the same time, the vehicle and charging station agree on a RunID – an unique identification feature that must be contained in all subsequent messages of the same SLAC session.

The establishment of a connection via SLAC runs sequentially through various phases such as the "Sounding Phase", "Attenuation Characterization Phase", "Validation Phase", "Matching Phase" and "Amplitude Map Exchange Phase". Based on signal attenuations of various degrees, the vehicle is able, with the help of the SLAC mechanism, to identify which charging station out of the many responding charging stations it is physically connected to. At the end of this process, an electric vehicle and charging station form a shared AVLN over which the higher protocol layers exchange encrypted information.

Embedded Systems for ISO/IEC 15118 Charging Controllers

A major challenge for the automotive industry on the one hand and the manufacturers of the charging infrastructure on the other hand is the designing of products that conform to ISO/IEC 15118. The fastest way for automobile manufacturers to achieve functional results when developing the charging ECU is to use ready-made, field-tested embedded solutions such as MICROSAR V2G (Vehicle-to-Grid) of Vector. MICROSAR V2G **(Figure 1)** is a cluster of basic software modules and is part of an AUTOSAR-compatible product line, which allows users to assemble an ECU software package that is tailored to their exact requirements. MICROSAR V2G covers all technologies and protocols on the various ISO OSI layers that are needed for charging with AC and DC voltage current according to ISO/IEC 15118 and DIN 70121. This includes suitable hardware support for HomePlug Green PHY as well as implementation of a future-proof TCP/IP dual stack for IPv6 and IPv4. The smart charging communication counterpart for manufacturers of charging stations is called vEVSE. Based on Embedded Linux, it contains all ISO/IEC 15118 and DIN

70121-relevant components (SLAC, EXI, and message sequence).
Special attention must be given to the testing of smart charging developments. A charging ECU that ist equivalent to ISO/IEC 15118 and DIN 70121 not only has to be integrated error-free in the vehicle's own electronics. Rather, it is virtually open to the outside where it will confront a varying charging infrastructure involving different power companies and manufacturers. "Impasse situations" at charging stations due to systems that are not fully compatible would be highly counterproductive and a serious setback for acceptance of electric mobility. In order to track down all conceivable errors and incompatibilities, a high level of test coverage and testing depth is needed. This can only be achieved by using systematic and automated test runs with powerful HIL (Hardware in the Loop) systems, for example.

Mastering New Kinds of Test Requirements

Smart charging involves several new technologies that the test system must handle. It must provide hardware and software support for all methods used in ISO/IEC 15118 and cover all the test cases needed in this context. This extends across all ISO OSI layers from the physical layer of the PWM signal modulated for Power Line and HomePlug Green PHY and mechanisms such as SLAC to Internet and Ethernet technologies. A high-performance HIL system is able to completely and realistically simulate the environment for the system under test (SUT). In addition to the PLC and SCC (smart charging communication) methods, digital and analog values of sensors and actuators must be simulated so that the charging station side or vehicle side can be operated according to specification. A powerful remaining bus simulation additionally generates the events on vehicle buses, such as CAN, together with the underlying ECU logic.
For testing of ISO/IEC 15118 and DIN 70121 developments, Vector offers a special plug-in SCC board for its VT System HIL tester. The VT7870 Board **(Figure 2)** provides two operating modes and is suitable for testing both electric vehicles and charging stations. When vehicles are tested, the HIL system simulates the charging station, and when charging stations are tested, it simulates the vehicle electronics. The board contains a Devolo dLAN Green PHY module with QCA7000 chip set that is used not only to receive the signal of the power line communication modulated for the control pilot but also to generate this signal **(Figure 2, text box below)**. The VT System has a modular design, can be flexibly scaled, and is capable of simulating error scenarios, wire breaks, and short circuits. For charging tests with real amperages and voltages, controllable power supplies and electronic loads can be connected. A PC with CANoe software, Ethernet, and the SCC add-on package is used for configuring test scripts and for test sequence control, report creation, and results analysis. If necessary, CANoe also performs the remaining bus simulation. The automated test cases can be defined with vTESTstudio – a powerful test design and authoring tool **(Figure 3)**.

Functions of the VT7870 Hardware Module:
> Simulation of charging station or vehicle
> PWM Control Pilot (CP) communication
> High level power line communication (PLC)
> Error simulation on Control Pilot
> Simulation of component tolerances
> Connection and measurement of the proximity contact

Figure 2: VT7870 is the smart charging communication module for ISO/IEC 15118-compliant testing of charging communication

Figure 3: Example of automated test setup for onboard charger ECUs with simulated charging station

Outlook

The expected growth in the numbers of both electric vehicles and charging station systems, which is accompanied by an increasing penetration rate for high-level communications, will lead to enormous increases in combinatorial complexity. This will create challenges in terms of interoperability of charging stations and vehicles and in terms of conformity to applicable standards. To successfully overcome these challenges, a reliable and well-developed product and testing strategy is needed, as are mutually coordinated components and tools. The presented line-up of products and services by Vector represents a universal approach that assures successful expansion of electric mobility.

Translation of a German publication in Elektronik automotive, special issue E-Mobility, July/2014

Image rights:
Vector Informatik GmbH

Links:
More information about Vector's solution for E-Mobility on:
www.vector.com/ev

Dirk Grossmann
is Team Leader for Embedded Software where he is responsible for development of MICROSAR V2G and vEVSE. He studied electrical engineering at the University of Stuttgart, Germany, and gained 6 years of automotive development experience in the area of ECU software before coming to Vector in 2003.

Dr. Heiner Hild
is Team Leader and Product Manager for the VT System at Vector. He studied physics at the University of Stuttgart, Germany, and received a doctorate in image processing and geographic information systems. Before joining Vector in 2014, he worked over 10 years in automotive development on driver assistance systems.

ECU Testing for Electric and Hybrid Vehicles

Intelligent Measurement of Dynamic Current Consumption

Electrically powered vehicles pose special challenges to test systems: in particular, ECUs must utilize electrical energy efficiently in development and testing. This requires measuring the current consumption of the ECUs as a function of software states – this is the only way to attain the required energy efficiency.

The research conducted by the advanced development and production development departments of automotive OEMs and suppliers show that much work is clearly being done on electrically powered vehicles and related mobility concepts. Some products have already advanced beyond the experimental stage, while many others are prepared to follow shortly. This has put systematic testing of vehicle electronics into focus in accountable testing departments. During initial conceptual and prototype phases, existing components from conventional vehicles are often used, while the focus was on the electrical powertrain and innovative functions. However, in production development of initial electric and hybrid vehicles, it is now necessary to transfer important findings of the past decades related to E/E architecture validation to the new electric vehicles.
What is different in testing vehicle electronics in electrically powered vehicles? How do established test strategies need to be changed, supplemented or even replaced?

New Challenges in e-Vehicles

The most noticeable change in on-board electronics is the new ECUs for the electrical drive components and the battery **(Figure 1)**. In a fully electric vehicle, the engine controller is replaced by a controller for electric drives. The transmission ECU has been eliminated, but a new battery management system handles all aspects related to the battery cells, and an on-board battery charger enables "refueling". The testing departments can transfer existing test methods to these ECUs. For the batteries and electric motors in particular, suitable environment simulations must be integrated in the test benches. On closer examination, this turns out to be a great challenge, because the high voltages used in the electrical powertrain have a tremendous influence on the test bench equipment. The primary challenge is to attain the same testing depth and the same degree of test automation for the new systems under these conditions as for known electronic components.

With electric charging, a new communication interface is also assuming an important place in the vehicle: the communication between the charging station and the vehicle. Just a few simple electrical signals suffice if communication is limited to control and verification of the actual charging process. However, complex communication is needed for more extensive functions. Such functions include automatic billing for charging and integration of the vehicle in a "smart grid" – i.e. cost-benefit optimized charging for the vehicle that is temporarily connected to the infrastructure. All around the globe, various development groups and standardization committees are addressing these issues. Clearly assuming a central role here is ISO 15118, which describes intelligent charging with AC power. Communication is conducted over the power line (Power Line Communication), utilizes the Internet Protocol (IP) and is based on typical Internet technologies (TCP/UDP, DHCP, XML, JSON, to name just a few). These technologies are indeed widely used – every PC handles most of them – but implementation with the limited resources of an automotive ECU is new. The testers are faced with new challenges as well; they now need to analyze these protocols and set up suitable test environments.
Another aspect: While communication in the vehicle previously ran between known ECUs, now there is complex communication between the vehicle and a variable infrastructure. To assure troublefree operation in the field, a broad-based test of the vehicle interface is necessary. The same applies to the charging stations. In the future, uniform test content might be arranged (key word: Conformance Test). Certainly, some aspects are still in flux with regard to standardization; however, adequate tests must still be developed for ongoing vehicle projects.
Power-saving technologies play a central role in the development of electrically powered vehicles. In the conflicting priorities between range, available battery technologies and system costs, electrical energy is an extremely scarce resource. One might say fine, but energy-saving systems are under consideration for vehicles with internal combustion engines as well – the drivers here are goals and commitments to reduce CO_2 emissions. Each vehicle should therefore consume as little energy as possible on the ECU and network levels.

Energy Saving in the ECU Network

In the development of ECUs and networks, various ideas are currently being pursued. Not all of these approaches are new, but recently they are being addressed with high priority and are making their way into standardization (notably in AUTOSAR).

- In Partial Networking, logical networks are formed which must conform to the physical networks. The energy-saving sleep states are defined on the logical partial networks. They can be implemented more effectively in this way, because fewer ECUs need to leave the sleep state in many operating states.
- In Pretended Networking, individual ECUs are constructed so that they appear to one another as active on the communication network, but they can essentially go into a sleep mode internally. This is used whenever the actual ECU functionality is not needed. Extensions in the ECU hardware ensure that periodic messages continue to be sent, and the ECU core is awakened under certain conditions.
- The key word ECU Degradation covers all measures in which unnecessary parts of the ECU are deactivated. This ranges from simple switching off of electrical driver stages to fine-granular control of chip-internal logic units. GPS signals (optional).
- Merging of different functions into just a few ECUs not only reduces overall energy requirements, but also offers savings potential in hardware costs. AUTOSAR methodology already supports this approach in its distinct functional perspective. High-performance multi-core proces-

Figure 1: New in the e-vehicle are ECUs related to the electrical powertrain and the obligatory use of power-saving techniques on all levels.

Figure 2: The power supply module of the VT System determines the DUT's current consumption with high measurement precision and time resolution.

sors are also more energy-efficient than equivalent individual processors in dedicated ECUs.

> Development could even go a step further and combine multiple logical ECUs into one physical ECU by means of a virtualization layer. However, the savings potential comes at the risk of unknown mutual interactions.

The refined techniques designed to attain energy savings increase the complexity of ECUs and networks. That is why systematic tests introduced early in development phases are more important than ever.

Testing Current Consumption in a Differentiated Way

The current consumption of ECUs plays a key role in testing. Previously, current consumption was acquired as a mean total value, or it depended on relatively static operating modes, e.g. the key word Sleep Mode. In consumption-optimized systems, however, it must be determined dynamically in relation to the ECU's software states. Since the savings potential consists of many individual optimizations, the tester must test very precisely whether the current consumption meets expectations in the individual ECU states. The same applies to development: The developer must be able to evaluate the current consumption in reference to the momentary software state.

High-performance test systems such as the VT System **(Figure 2)** from Vector therefore enable precise acquisition of the current consumption of the ECUs under test and permit correlations to system states. The modular VT System test hardware simulates the sensors and loads, stimulates the inputs and measures the output signals, generates electrical faults such as short circuits and line breaks, and permits toggling between internally simulated and externally connected sensors and loads. Moreover, the VT System controls the supply voltage and measures the consumed current. Fast, automatic range switching ensures that very high currents while operating under load, as well as minimal currents in power-saving and sleep modes, can be measured with high resolution.

With the VT System, CANoe handles automation of these tests. This software simultaneously permits simulation of the rest of the network nodes (remaining bus simulation) and services software accesses to the ECU. Furthermore, it contains comprehensive analysis functionalities, so that developers can use the system for testing and debugging. CANoe was specially developed for analysis, simulation and testing of ECUs and networks, and therefore it is able to act as a communication partner for the various charging station interfaces. In testing a charging ECU, the charging station is simulated – and vice versa.

Testing and analysis require that various signals be used to determine the system states of ECUs and networks. They may be bus signals, hardware signals or information that is read out from the ECU via diagnostic or calibration interfaces. For example, a value on the internal state of the ECU read via XCP could provide information in the ECU's internal state. What is important here is that all values are available directly and with the same time base, as in CANoe. Suitable tool support makes it easy to evaluate ECU behavior **(Figure 3)**. This is particularly necessary in the development phases and during debugging. Using the same basic approach, it is also easy to formulate test programs and automate the tests. Even in E-vehicles, testing departments can thereby attain the same high level of automation that has proven to be a key component of validation strategies in recent years.

Figure 3: In CANoe, power consumption and the ECU state are clearly displayed and easy to analyze.

Translation of a German publication in: Automobil Elektronik, August/2011

Image rights:
Vector Informatik GmbH

Links:
Homepage Vector: www.vector.com
Product Information VT System: www.vector.com/vtsystem

Dr. Stefan Krauß
is a group leader at Vector Informatik GmbH in Stuttgart. As Product Manager he is responsible for the VT System.

New Bus Interfaces for Electric/Hybrid Vehicle Development

The many different bus systems used in the development of electric/hybrid vehicles (EV/HEV) are pushing conventional hardware interfaces to their performance limits. A crucial factor in determining testing depth, analysis quality and simulation options is the degree of real-time performance in data exchange. Communication between vehicle networks and the test and simulation tools is of primary importance here. To satisfy these heightened requirements, Vector has added a modular high-performance interface with integrated real-time computer to its interfaces product line. This interface's real-time performance leads to short deterministic cycle and response times and enables flexible use in the EV/HEV development field and in numerous new application areas.

An ideal hardware interface for EV/HEV development is characterized by excellent real-time capability, easy handling with such features as plug & play via USB, extendability and maximum versatility in potential uses. These somewhat contradictory requirements could not be satisfied by previous concepts.

Interface with High-Performance CPU Core

Therefore, in developing its latest generation of interfaces, Vector took the approach of implementing active hardware architecture with an integrated CPU. The resulting solution covers fundamental uses such as bus analysis, remaining bus simulations, high-performance measurement and calibration over CCP and XCP, execution of gateway routing relationships, quick flashing and execution of complex diagnostic services.

The VN8900 system's many different interfaces and its high-performance computing core let users execute remaining bus simulations and gateway applications in real time. New application areas are also emerging in ECU development; of particular interest here those that satisfy the stringent requirements of electric/hybrid vehicle development. It should be emphasized that rapid prototyping requirements (prototype ECUs and bypassing) are supported, which are located earlier in the development process (left side of the V diagram). This will be addressed in more detail later. A significantly longer time span is covered in the ECU development process than when conventional interface hardware is used **(Figure 1)**.

Real-time Execution of Remaining Bus Simulations and Gateway Applications

Great challenges are posed by requirements for real-time capability with very fast reaction times, a simple plug & play connection via USB, and preserving comprehensive use of widely used Vector tools such as CANalyzer/CANoe and

Figure 1: Comparison: range of uses of conventional interface hardware versus the extended range of uses for VN8900 systems.

CANape. These challenges are addressed by distributing work between the host PC and the intelligent interface hardware. Only relevant tasks are executed directly on the real-time capable hardware interface, while other standard tasks such as data preparation, storage and visualization are run on the host PC as usual.

Standalone Operation Opens up New Application Areas

If no user interactions are necessary, remaining bus simulations and gateway applications can be autonomously executed on the interface hardware. Remaining bus simulation capability is absolutely essential in developing and testing of individual ECUs, even if some of the electric or hybrid vehicle's ECUs are unavailable. CANoe is used with the relevant OEM extension to create the remaining bus simulations or gateway routing with just a few mouse clicks, and they are then loaded to the interface hardware. To integrate standalone applications in other systems, the Ethernet/UDP based interface known as FDX (Fast Data Exchange) is available. It permits quick write and read accesses to bus signals and system variables. Prepared CANoe/CANalyzer standalone configurations are loaded to the interface module from any desired PC via a manager. This process can also be automated for optimal integration in a suitable environment.

EV/HEV Rapid Prototyping: Universal Bypassing Solution

The bypassing solution from Vector lets users execute parts of the ECU software outside of the ECU with deterministic response times, i.e. on the VN8900. This especially makes sense in EV/HEV development if:

> Functions need to be modified quickly without running through the entire ECU code generation and flash process
> Functions need to be switched easily for comparative measurements
> New functions cannot be executed with the existing ECU hardware, e.g. if it lacks sufficient memory or CPU performance.

In the ECU, it is also necessary to have the function to be bypassed "cut loose," i.e. it is no longer executed. Input and output parameters of the bypass function are typically located in the ECU's RAM, and they need to be transferred as quickly as possible to the bypass computer. The standardized protocols CCP, XCPonCAN, XCPonFlexRay and XCPonEthernet are all candidates for the bypass transfer **(Figure 2)**. The VX1000 measurement and calibration sys-

Figure 2: Data flow diagram of the bypass solution

Figure 3: Measuring and calibrating VN8900 prototype ECUs

tem from Vector enables very quick access to an ECU's RAM. This process involves connecting a POD (PlugOnDevice) to the ECU via Data Trace or debug interfaces such as Nexus, JTAG or DAP; in the VX1000 base module, the information is converted to the XCPonEthernet protocol and is transferred to the VN8900. Performance measurements indicate a measurement data rate up to 100 times higher than is possible with CAN. To measure the up to 50 kHz drive signals of the electric motors, the VX1000 system is frequently used in electric/hybrid vehicle projects, which can then be easily extended to a complete bypass solution with the VN8900 hardware. An optional plug-in I/O board in the VN8900 gives the bypass runtime environment access to digital and analog signals.

EV/HEV Rapid Prototyping: VN8900 as Prototype Hardware Interface in Battery Control ECUs

This application case is of special interest when no ECU target hardware is available. Today, this is frequently the case for EV/HEV battery control ECUs, because it is impossible to re-use existing ECUs from previous models, and in most cases the ECUs must be completely redeveloped for electric/hybrid vehicles.

In the CANoe standalone runtime environment, the ECU software for this application case is executed on the VN8900 under real-time conditions in the form of Simulink or "C" executables. The communication with other bus nodes is via CAN, LIN or FlexRay bus interfaces. Other devices, e.g. sensors, might be connected via the digital and analog I/O interface.

Figure 4: Using the CANape Simulink Model Explorer, the model can be visualized, and internal model parameters can be displayed and calibrated.

Unlike existing solutions for ECU prototypes, no cross-compiler is used in the VN8900 approach. This means that all familiar and proven CANoe functions – such as network management, the interaction layer and transport protocols (e.g. for diagnostics) – can be used directly in ECU simulation.

In executable generation from Simulink models via a special CANoe target, an XCP driver can be compiled into the executable. As a result, the VN8900 prototype runtime environment over XCPonEthernet can be measured and calibrated quite easily with a calibration tool like CANape **(Figure 3)**.

Many VN8900 customers are developing more and more new application cases themselves. For example, the standalone operating mode permits use for data logging without a host PC by utilizing the integrated 2 GB flash memory (approx. 1 GB available) or external USB connections for mass storage.

Summary and Outlook

In developing ECUs for electric/hybrid vehicles, users benefit from the flexibility of the VN8900 interface solution, which has up to seven CAN, LIN and FlexRay channels as well as optional I/O hardware extensions. The integrated real-time core enables short and deterministic response times. Because it can be used as rapid prototyping hardware or for standalone applications, the VN8900 can be used for a wide range of tasks over a many project phases. Finally, the numerous application cases with one and the same hardware and associated production volumes contribute to making the VN8900 a cost-efficient solution, especially in the development of EV/HEV vehicle networks.

Translation of a German publication in Hanser Automotive, 03-04/2011

Links:
Homepage Vector: www.vector.com
Product Information CANape: www.vector.com/vn8900

Alfred Kless (Dipl.-Ing. (FH))
After graduating with a degree in Electrical Engineering at the College of Applied Sciences at Esslingen, Mr. Kless initially worked at ALCATEL, including as team leader for software development and business development of test systems. Since May 2004, he has been employed at Vector Informatik in Stuttgart as Business Development Manager for the "Measurement & Calibration" and "Network Interfaces" product lines.

Networking Heavy-Duty Vehicles Based on SAE J1939

From Parameter Group to plug-and-play Application

In networking ECUs in heavy-duty vehicles, it is the J1939 protocol that plays a key role. J1939 networks are based on the CAN bus (high-speed CAN per ISO11898); they are primarily used in powertrain and chassis components. The protocol creates a uniform basis for communication between electronic control units, and it supports the plug-and-play principle. Special J1939 tools and software components spare developers from needing to train in the details of the J1939 protocol, and they improve the quality of the development process.

The J1939 protocol – founded in the USA and defined by the Society of Automotive Engineers (SAE) – serves above all to preserve a uniform perspective and uniform handling of the most common vehicle components of various vehicle types and manufacturers. In this context, it is interesting to note certain distinct differences between the European and North-American heavy-duty vehicle markets. For example, heavy-duty vehicle buyers in the USA have prescribed to OEMs which components they need to install in specific vehicles. In Europe, on the other hand, it is the OEMs who fully define the design of the entire vehicle, including the components and their configuration.
Besides using uniformly defined signals and data formats to communicate, it is of course important that receivers know how to interpret the information. Ideally, it should be possible to interconnect individual J1939 components based on a plug-and-play scheme. Despite all of its standardization aspects, J1939 gives OEMs sufficient freedom for customized extension of communication. This is especially important in promoting innovations, because no OEM wants to announce or discuss plans in working committees before their implementation.

ISO Layers Model decouples the Application from Transmission Physics

From the perspective of the ISO/OSI network model, J1939 is essentially based on the Application Layer (Layer 7), Network Layer (Layer 3), Data Link Layer (Layer 2) and Physical Layer (Layer 1) **(Figure 1)**. This lets developers work with signals without needing to be concerned about communication details on the Application Level, such as details of the transport protocols. J1939 documentation and definition is oriented toward individual layers, and this is expressed in the names of the total of 14 documents of the standard. For example, documents of the 7 series such as "J1939/71" refer to the Applications Layer, document J1939/21 the Data Link Layer, etc.

Figure 1: The J1939 protocol essentially covers the Application Layer (Layer 7), Network Layer (Layer 3), Data Link Layer (Layer 2) and Physical Layer (Layer 1), so that it is no longer necessary to be concerned about communication details when working on the application level.

CAN Message Format in J1939

Although J1939 utilizes normal 29-bit CAN messages with up to 8 bytes of data, here the CAN identifier quasi defines the mask of a uniform J1939 scheme **(Figure 2)**. This is necessary to assure plug-and-play properties. The CAN identifier – besides identifying the useful data with the help of the Parameter Group Number (PGN) – also contains the J1939 ECU address of the sender and if applicable the address of the receiver too. In addition, the three most significant bits of the CAN identifier are reserved for the priority field; although these bits do not replace the implicit priority established by the CAN protocol, they make it possible to organize the Parameter Groups into up to eight J1939-specific priority levels. These priorities are used to adjust the priority to momentary application requirements at the time the Parameter Group is transmitted – or during an optional ECU configuration phase. This makes it possible to fine-tune communication to the application without the SAE protocol permanently setting the priority when the Parameter Group is defined.

The Name says it all: J1939 Device Names

J1939 defines device names that are each represented by a 64-bit number. When an ECU is switched to active in the

Figure 2: The J1939 message format – which is based on normal 29-bit CAN messages – requires some supplementation to achieve plug-and-play capability. The CAN identifier quasi defines the mask of a uniform J1939 scheme.

plug-and-play network, the device name serves to identify the device and its functionality. The device name is subdivided into different elements, between which certain dependencies exist. The independent fields include the Industry Group and Manufacturer Code. The Industry Group is used to establish the functions required in the network, since the J1939 protocol is not only used in conventional heavy-duty vehicles but also in vehicles in the agricultural and marine industries. Each ECU carries one of the SAE assigned Manufacturer Codes that can be requested from that organization. Since each device also has a serial number, the complete name over all fields is unique worldwide, and there are no overlaps.
Since addressability of the devices is inefficient in practice when 64 bit long device names are used, the SAE defines 8-bit addresses for the individual vehicle components in the heavy-duty vehicle field; practically these addresses never change over the life of the components. This does not apply to the agricultural and marine industries; there the addresses are dynamically negotiated at the start, based on the device name. The addresses 0 to 127 are assigned to the most commonly used ECUs such as engine, transmission, retarder, brakes, etc., while the range from 128 to 247 is reserved for agricultural, marine, construction equipment, etc. Service tools, OBD scanners, etc. occupy addresses from 248 to 253. Finally, there are the special addresses: 254 to identify devices that do not have their own address and 255 that is used as a global address for addressing broadcast messages.

Types of Communication: Point-to-Point or Broadcast

The J1939 protocol supports two communication types: point-to-point transmissions (1:1) are directed to precisely one target address; they are used for device configuration or ECU commands, for example. Broadcast messages (1:n), on the other hand, are simultaneously addressed to all bus nodes, which is practical when it comes to sending out measured values, error handling and diagnostic purposes.

Flexible Network Topology

J1939 works with a passive bus that is terminated at each of its two ends with 120 Ohm impedance. The advantage here is that individual ECUs can be connected to the bus via branch lines with a length of 1 to 3 m. This enables flexible wire harness design, provided that a total bus length of 40 m is not exceeded. Depending on the physical transmission layer, between 10 and a maximum of 30 nodes may be connected to the network. J1939 provides uniform diagnostic access for service testers and on-board diagnostics. Legal requirements specify that a branch line with a length of up to 5 m must be possible here, e.g. for road tests of the emissions control system. Bridges can be used to extend J1939 networks to include trailers/implements, enabling implementation of a separate network there **(Figure 3)**. In the EU, ISO 11992 has prevailed for this purpose, while in the USA the "Power Line Carrier" is state-of-the-art technology.

Timing Requirements in ECU Design

In designing J1939 ECUs, care should be taken to ensure that no messages are lost due to hardware or design limitations. At a baudrate of 250 Kbps, transmission of one bit takes 4 microseconds. With 8 data bytes, a typical message length of about 128 bits is obtained, yielding a transmission time of about 500 microseconds per CAN message. The shortest message length, however, is 64 bits, i.e. it must be possible to receive messages at intervals of 250 microseconds and process them sufficiently fast. In practice, this leads to a high interrupt load due to the CAN controller's often limited CAN identifier filtering capabilities. Filtering also usually needs to be implemented in software.

Figure 3: With regard to network topology, J1939 enables flexible design of wire harnesses. Individual ECUs can be connected via branch lines up to 3 m in length.
Bridges make it possible to realize separate networks on implements and trailers.

Testing and Diagnostics of J1939 Components and Systems

In view of the rising number of J1939 ECUs and the fact that software solutions in heavy-duty vehicles are becoming increasingly complex, a systematic strategy for testing and diagnostics also continues to gain in importance in the J1939 field. Tests are indispensable in all development phases, from functional tests to integration tests to driving trials in the total vehicle. It is well known that the later that errors are detected, the more complicated and expensive it is to correct them. However, it is generally only possible to test ECUs comprehensively after they have been integrated in the network structure. Consequently, weak points are often not revealed until very late, unless one relies on the support of proven software tools right from the start.
Given this situation, the use of specialized tools offers developers substantial simplifications in testing and diagnostic tasks. For many years now, Vector has been actively involved in SAE J1939 subcommittees and regularly participates in working sessions. With a universal tool chain for all J1939 projects, it is possible to efficiently solve the most challenging tasks in networking and communication in the heavy-duty vehicle field [1]. Besides development, testing and analysis tools, embedded software components tailored to the special requirements of J1939-based applications are available, and customized project work and training events round out Vector's products and services.
A J1939 extension is available for the widely used CANoe development and test tool; it spares heavy-duty vehicle developers from needing to train in the details of the J1939 protocol. The package from Vector extends basic software functionality to cover all necessary protocol-specific features. When CANoe .J1939 is used consistently, the models and databases created in the design phase not only serve as a foundation for simulation during development, but also for all tests accompanying development up to and including later diagnostic tasks **(Figure 4)**. With the help of simulated nodes, it is possible to set up and execute tests for the ECU to be developed. The tests are further refined during development and are used in verification of the total system.

Extensive J1939 Test Library

The CANoe Test Feature Set is made up of CAPL test modules, XML test modules and .NET test modules. They are able to cover all challenges arising in testing tasks of varying complexity, from simple to difficult test cases. While the C-like script language CAPL is ideal for creating extensive test scenarios, the primary focus of XML is on simple parameterization of test patterns and simple generation of test procedures. That makes it possible to implement application-specific test modules (function libraries) in CAPL and then generate test control that is individually adapted to the ECU configuration. The J1939-specific extensions in the Test Service Libraries allow the system react to Parameter Groups (PG) instead of typical CAN identifiers. They also offer test patterns for J1939 protocol functionality and checks (background monitoring) for protocol violations. For example, it is possible to test whether the ECU is able to send all Parameter Groups at the configured cycle time under high bus load. Furthermore, it is possible to send faulty transmissions via the BAM (Broadcast Announce Message) and CMDT (Connection Mode Data Transfer) transport protocols for test purposes.

Figure 4: Tests conducted with the help of simulations during development make it possible to detect and correct errors early on in all development phases. The CANoe Test Feature Set offers extensive testing and analysis capabilities.

To create the test modules – besides the J1939 Test Module Manager and the convenient Test Automation Editor – the Option DiVa is useful. DiVa creates a connection between CANoe and the diagnostic specification tool CANdelaStudio, so that specifications created there can be ideally used in further ECU-specific diagnostic tests.
Other functions of the Test Feature Set relate to test flow control and automatic report generation, including statistical information in XML or HTML format based to individual requirements. Further options for automating test processes are enabled by the COM interface, e.g. options relating to flow control, parameter changes or status queries. CANoe Option J1939 provides a trace window, J1939 diagnostic monitor and J1939 diagnostic memory access for diagnostic purposes. The diagnostic monitor supports various J1939 diagnostic messages, such as DM1 and DM2, and it serves to display and clear active errors. Also possible is access to memory areas, objects and parameters as well as periodic object updating for monitoring purposes.

Integrating Matlab/Simulink Models in J1939 Network Simulations

Generally, various function models are created for mechanical components such as transmission, powertrain or even the entire vehicle during the different heavy-duty vehicle development phases. ECU architectures are initially saved in virtual CANoe function models and are implemented step-by-step on the final target hardware platform. CANoe .J1939 can also integrate Matlab/Simulink models in ECU and network simulations **(Figure 5)**. With the Real-Time Workshop from Mathworks the user generates a *.DLL for CANoe so that variable names and units are compatible.

Progressing through the various stages of the V development model, individual tests and subsystem tests are possible through final verification of the overall system. This enables early detection and correction of errors. If an error is found, the automated tests can be restarted at any time; they minimize the risk of side effects in error correction. As a result, development is characterized by short verification cycles, enabling a seamless transition from MIL (Model in the loop) to SIL (Software in the loop) and then to the real ECU (HIL – Hardware in the loop). If there are exceptional real-time requirements of the simulation platform, a special real-time version is available with CANoe RT.

Realizing Goals quickly with standardized Embedded Software Components

Use of CANbedded J1939 software components leads to quick development results. These components largely relieve developers of the need to handle all of the details of the J1939 standard, and they avoid duplicated developments. A key aspect is a central OEM-managed database containing all elementary information related to ECU communication. Depending on requirements, this information might be distributed to other working partners, producing flexible distribution of tasks between the OEM, network specialists from Vector and suppliers **(Figure 6)**. The latter can use the GENy configuration tool for specific settings and parameterizations. The results are reduced cost and timing for implementation and testing, compatibility on the CAN bus based on unambiguous signal interpretation and maximum quality and flexibility in the J1939 communication stack. CANbedded J1939 supports all relevant microcontrollers and is characterized by low ROM and RAM memory requirements as well as high runtime efficiency.

Figure 5: Not only is CANoe able to simulate functional models of ECUs during the development process and integrate models created with Matlab/Simulink in the scenarios; at the same time it also serves as a convenient user-interface.
(Source: Renault Trucks)

Image rights:
Vector Informatik GmbH

Internet links:
[1] J1939 solutions from Vector – www.j1939-solutions.com
[2] Download of presentations from J1939 User Day – www.vector-worldwide.com/ud [most of them are German]

Peter Fellmeth
studied at the University of Applied Sciences in Esslingen, Germany, majoring in Computer Engineering and specializing in Automation Technology. He is team leader and product manager at Vector Informatik GmbH, where he is responsible for the development of products and customer-specific projects related to J1939, ISOBUS, Ethernet and DeviceNet.

Thomas Löffler
studied Automation Technology at the University of Applied Sciences in Reutlingen, Germany. He has been employed at Vector Informatik GmbH since 2000, initially in the DeviceNet area, and since 2002 in the J1939 and ISOBUS area. His areas of specialization are configuration and generation tools for embedded software, support of customer projects and product and protocol training programs.

Figure 6: Standard software components of the CANbedded J1939 package lead to quick development results without developers needing to be concerned with all details of the J1939 standard. A centrally managed database avoids duplicated developments and enables optimal work distribution.

Quo Vadis SAE J1939 Standardization

Due to new application layer requirements, SAE is continuing to develop the J1939 standard, which is primarily used to network powertrains in commercial vehicles. However, optimizations and extensions are being made in the other communication layers as well, right up to the physical transmission layer. This article summarizes the current state of discussions within the SAE J1939 working committee, such as the planned introduction of the 500 kbit physical transmission layer and changes to network management. Moreover, it also explains ongoing efforts to standardize J1939 in AUTOSAR Release 4 and WWH-OBD diagnostics.

Based on the CAN bus (High-Speed CAN per ISO 11898), the SAE J1939 standard is used primarily to network the powertrain and chassis in commercial vehicles. The protocol creates a uniform foundation for communication between the electronic ECUs and operates by the plug-and-play principle. The J1939 standard is an active standard that currently consists of 19 documents **(Figure 1)**. The responsible SAE subcommittees generally meet four times a year to decide on changes and further developments. The current versions of the documents may be purchased either individually or together as a package in so-called "JPaks" from the the SAE website [1].

More Bandwidth

For years now, the maximum 250 kbit bandwidth specified in the standard has forced commercial vehicle developers to work at the limits of performance [2]. From a communication perspective, development of the 500 kbit data transport layer is a long overdue step. European commercial vehicle producers in particular are seeking a final decision in the near future. The specification will be released in a separate document, J1939-14, and its key aspects are:

> Twice the baudrate, 500 kbit instead of 250 kbit
> Use of shielded and unshielded cable as defined in [2] and [3] is still possible.
> Topology is essentially a bus that has branch lines with a max. length of one meter. To connect diagnostic tools, a branch line (from the diagnostic socket to the diagnostic tool) with a length of 5 meters may sometimes be used.
> The bus is terminated at both ends with a characteristic impedance of 120 Ohm. Up to 30 nodes are possible.

The specification for the diagnostic plug [4] was adapted to 500 kbit operation. In addition, a new "Type II" diagnostic socket is being used in the vehicle which has a green color coding, and its connector keying prevents use of the previous 250 kbit "Type I" diagnostic plug. A "Type II" plug is compatible with a "Type I" socket. Another change is that the "Type II" diagnostic socket defines pins previously used for SAE J1708/J1587 as reserved. Consequently, a J1708/J1587 network can no longer be addressed via a "Type II" diagnostic plug.

SAE Gets Serious about Dynamics

Changes are also being made in the area of Network Management [5]. For a long time now, the J1939 committee has been deliberating over ways to handle the short supply of permanently assigned ECU addresses. This is especially problematic for manufacturers of sensors with a direct bus connection. The number of new devices is growing rapidly due to heightened exhaust emissions requirements and the addition of assistance systems. Many alternatives were proposed but then rejected. They ranged from a dedicated network for sensors to implementation of a new protocol – e.g. by using previously reserved data pages. Meanwhile, the fact was that SAE has not assigned any more new addresses. That was an unsatisfactory situation for ECU suppliers. Often, they did not know whether their product designs would have lasting value. In the latest version of Network Management, SAE recommends implementing "Address Arbitrary Capable" ECUs. These ECUs are able to compute their own addresses based on the momentary vehicle configuration – and indeed at runtime. Essentially, this approach aims to utilize the mechanism of dynamic address allocation that has always existed in the commercial vehicle field, but has never really been implemented or used before.

In conjunction with Network Management, the newly added Name Management should be mentioned for the sake of completeness. This is a standardized interface for changing specific components of the 64 bit device name. This might be necessary if the relevant function or measurement parameter is derived from the device name. So, the device name can be used to identify the position of an exhaust gas temperature sensor – upstream or downstream of the catalytic converter, on the right or left side of a dual-flow exhaust system. Changes can be made to multiple ECUs in sequence and activated at a specific point in time. This could be helpful, for example, in a case where multiple ECUs of a network need to be assigned a new function simultaneously.

These changes in Network Management are supported in Version 7.5 of Vector's CANalyzer .J1939 analysis tool and its CANoe .J1939 development and test tool.

AUTOSAR and J1939 Come Closer Together

The introduction of AUTOSAR in the passenger car industry is ramping up quickly. Yet, there is also interest in exploiting the benefits of AUTOSAR in commercial vehicle and the agricultural machine markets. However, the special requirements of these markets have not been a focus in the development of AUTOSAR. Therefore, the AUTOSAR versions released so far have very limited potential in these markets. In particular, the requirements of SAE J1939 cannot be mapped to the current AUTOSAR concept, or only in a very limited way.

Dokument		Stand	Status
J1939	Recommended Serial Control and Communications Vehicle Network	Released status Feb. 2010 with contents through Feb. 2009	New version in process
J1939	Companion Spreadsheet	Released status Feb. 2010 with contents through Feb. 2009	New version in process with contents up to May 2010
J1939-01	Recommended Practice for Control and Communications Network for On-Highway Equipment	Released status September 2000	New version in voting process
J1939-02	Agricultural and Forestry Off-Road Machinery Control and Communication Network	Released status August 2006	Regularly scheduled revision after 5 years, scheduled from 2011
J1939-03	On Board Diagnostics Implementation Guide	Released status December 2008	Revision scheduled from November 2010
J1939-05	Marine Stern Drive and Inboard Spark-Ignition Engine On-Board Diagnostics Implementation Guide	Released status February 2008	Revision scheduled from August 2010
J1939-11	Physical Layer, 250 kbits/s, Twisted Shielded Pair	Released status September 2006	Regularly scheduled revision every 3 years, scheduled after release of J1939-14
J1939-13	Off-Board Diagnostic Connector	Released status September 2006	New version in voting process
J1939-14	Physical Layer – 500 kbit/s	Not yet released	In process, voting scheduled in 2010
J1939-15	Reduced Physical Layer, 250 kbits/s, Un-Shielded Twisted Pair (UTP)	Released status August 2008	Revision scheduled after release of J1939-14
J1939-21	Data Link Layer	Released status December 2006	Voting ended. Release scheduled for 2010.
J1939-31	Network Layer	Released status May 2010	No activity
J1939-71	Vehicle Application Layer	Released status Feb. 2010 with contents through Feb. 2009	New version in process
J1939-73	Application Layer - Diagnostics	Released status February 2010	New version in process
J1939-74	Application - Configurable Messaging	Released status November 2006	New version in voting process
J1939-75	Application Layer - Generator Sets and Industrial	Released status June 2007	Regularly scheduled revision every 5 years, scheduled from November 2010
J1939-81	Network Management	Released status May 2003	New version in voting process
J1939-82	Compliance – Truck and Bus	Released status August 2008	No activity
J1939-84	OBD Communications Compliance Test Cases For Heavy Duty Components And Vehicles	Released status December 2008	New version in process

Figure 1: Status of SAE J1939 documents (status in September 2010)

Figure 2: Layout of a 29-bit identifier in J1939 networks.

The "static" approach of AUTOSAR stands in contrast to the "dynamic" behavior of J1939. The AUTOSAR architecture only allows fixed CAN identifiers, i.e. there is a fixed allocation between precisely one CAN identifier and one message layout. In contrast to this, a J1939 specific message layout is only allocated to a specific part of the identifier, known as the Parameter Group (PG). Some of the other components of the 29-bit identifier are dynamic and not defined at the time of configuration. Such a dynamic identifier can be modeled in AUTOSAR by creating a separate static identifier for each combination of priority, source address (SA) and destination address (DA) that can occur in a network **(Figure 2)**.

When all nodes of a J1939 network are known, and node addresses are already defined at the time of configuration, it is relatively easy to map J1939 PGs to AUTOSAR: In case of such a static network, the ECU addresses are fixed. Therefore source and destination addresses are fixed, and so it is possible to work with static identifiers. To transmit data that is longer than the 8 bytes available in a CAN frame, J1939-21 specifies two transport protocol (TP) variants. They are the Broadcast Announce Message (BAM) variant and Connection Mode Data Transfer (CMDT; also known as RTS/CTS) variant. Both of them are already defined in AUTOSAR Release 4.0 and have been available since December 2009. Therefore, AUTOSAR Release 4.0 already covers the requirements of many European commercial vehicle producers.
More in-depth support of J1939 requirements is planned for the end of 2012 in AUTOSAR Release 4.1. The target group here includes European and some North American commercial vehicle OEMs.
The primary extended features being added to AUTOSAR:

> Support of multiple messages with the same layout (the same Parameter Group)
> Network management per SAE J1939/81 without dynamic NM, i.e. without AAC (Arbitrary Address Capable)
> Responses to a request message
> Support for diagnostic services
> On-board diagnostics (WWH-OBD) via J1939

Together with large European commercial vehicle OEMs Vector actively participates in efforts to specify these J1939 extensions for AUTOSAR. Today, Vector already offers an AUTOSAR solution with a J1939 extension based on AUTOSAR Release 4.0 **(Figure 3)**. It will soon be used in production at one large European commercial vehicle OEM. The extension for AUTOSAR Release 4.1 is currently in the development stage.

Figure 3: AUTOSAR basic software from Vector contains the two J1939 transport protocols BAM and CMDT.

Description / contents of the specification	Document
General information and definition of use cases	ISO 27145-1
Definition of diagnostic data	ISO 27145-2
Definition of diagnostic services	ISO 27145-3
Communication between vehicle and external tester	ISO 27145-4
Tests for compliance to the standard	ISO 27145-5
External tester	ISO 27145-6

Figure 4: The WWH-OBD is specified in the six documents of ISO 27145.

Figure 5: On-board diagnostics in commercial vehicles is implemented by CAN-based protocols.

Commercial Vehicle Diagnostics Using WWH-OBD

On-board diagnostics (OBD) is a diagnostic system standardized by ISO; one of its applications is to monitor systems related to exhaust emissions control. Over the course of time, regional standards were derived from this standard (e.g. ISO15031), which have now been merged again into WWH-OBD (World-Wide-Harmonized On-Board-Diagnostics). This standard was initiated by the United Nations and documented in its Global Technical Regulation 5 (GTR 5). ISO 27145 represents the technical implementation of GTR 5. It establishes technical constraints for WWH-OBD. WWH-OBD initially targets the commercial vehicle market, but eventually it should be extended to other vehicle industries as well. ISO 27145 consists of six parts **(Figure 4)**. The current document status is a "Draft International Standard" (DIS), and a final version is anticipated by the end of 2011. The first step was to establish requirements for emissions control and diagnostic communications. This involved specifying the vehicle-side implementation, data access and OBD data. At this time, regional authorities are still defining limit and threshold values; harmonization will not occur until a later point in time.
For on-board diagnostics in commercial vehicles, the two CAN-based protocols "Diagnostics on CAN" (ISO 15765-4) and J1939-73 are widely being used today **(Figure 5)**. To enable a cost-effective transition to WWH-OBD, diagnostics over CAN will continue to be used at first. For the long term, diagnostics over the Internet Protocol (DoIP) should also be possible which would enable access that is either wired over Ethernet or wireless.
Different than in the current OBD-II standard, WWH-OBD only utilizes services already defined as ISO 14229 "Unified Diagnostic Services" (UDS). No additional OBD-specific services are needed. Specifically, WWH-OBD requires support of the UDS services shown in the figure **(Figure 6)**.
Effective at the beginning of 2014, all newly registered heavy-duty commercial vehicles must conform to Euro VI standards, and so they must have WWH-OBD diagnostic capability. Developments involving new types of vehicles must fulfill standards one year earlier by 1-1-2013.
UDS diagnostic services can be implemented with CANbedded communication software that also supports J1939, which is available from Vector and is practice-proven in many production implementations. Recently, a production-ready AUTOSAR solution has also become available for implementing WWH-OBD diagnostics via UDS.
The described developments in the J1939 field show how the standard continues to be adapted to current requirements on a regular basis. The transfer, extension and modification of concepts from passenger car technology aim to make the development of J1939 ECUs more cost-effective. Vector, with its many years of networking expertise, is making a contribution by actively participating in standardization committees. In turn, developers of commercial vehicle electronics benefit from early implementation of new standards in embedded software and development tools.

Diagnostic service	Diagnostic identifier	UDS service	Specification
Clear error memory	0x14	ClearDiagnosticInformation	ISO 15031 Mode 04
Read out error memory and environment data	0x19	ReadDTCInformation	ISO 15031 Mode 03, 07 and 02
Read out ECU data and test results	0x22	ReadDataByIdentifier	ISO 15031 Mode 01, 06 and 09
Execute routines	0x31	RoutineControl	ISO 15031 Mode 08

Figure 6: The WWH-OBD uses these diagnostic services from the UDS.

Translation of a German publication in Elektronik automotive, 12/2010

Image rights:
Vector Informatik GmbH

Literature:

[1] http://standards.sae.org
[2] "CAN und offene Protokolle im Nutzfahrzeug" ["CAN and Open Protocols in Commercial Vehicles"], Elektronik automotive 5/2005, page 60
[3] SAE, J1939-11, Physical Layer, 250K bits/s, Twisted Shielded Pair
[4] SAE, J1939-15, Reduced Physical Layer, 250K bits/sec, Un-Shielded Twisted Pair (UTP)
[5] SAE, J1939-13, Off-Board Diagnostic Connector
[6] SAE, J1939-81, Network Management

Links:

Homepage Vector: www.vector.com
Solutions for J1939: www.vector.com/j1939
Solutions for AUTOSAR: www.vector.com/autosar
Product Information CANoe .J1939:
www.vector.com/vi_canoe_j1939_en.html
Product Information CANbedded J1939:
www.vector.com/vi_canbedded_j1939_en.html

Peter Fellmeth
is Group Leader and Product Manager at Vector Informatik GmbH, where he is responsible for the development of products and customer-specific projects related to J1939, ISOBUS, Ethernet and Car2x. He is an active member of various working groups involved in the standardization of SAE J1939 and ISO 11783 (TC23/SC19/WG1).

Holger Söhnle
is Product Manager at Vector Informatik GmbH for Embedded Software in the area of J1939 and ISOBUS.

Integration of J1939 Systems in Practice

Commercial vehicle producers are continually being confronted with problems in integrating networked systems with the J1939 protocol. Weaknesses include information exchange between OEM and supplier and different variants of the J1939 standard. Initial approaches to improvement: establish a uniform data exchange format for the CAN communication, introduce a tool for conformity testing and define mandatory device profiles.

At the suggestion of key American truck manufacturers, the Vector US subsidiary hosted a conference to discuss improvement options for J1939 networking. At this conference, a number of presenters described their concepts and experience in integrating J1939 components. In addition, weaknesses were identified, and specific optimization potentials were discussed.

J1939 is not always J1939

The presentation by Vector showed how the SAE J1939 protocol takes on a different meaning in the USA, than it does in Europe [1]. The market structures that have evolved and the different technical requirements both play a role here. For example, the relationship between customer and OEM differs; a US customer not only selects the vehicle functions but the customer can also choose whether an installed ECU – a brake ECU for example – comes from supplier A or B. Therefore, the E/E design and communication protocol used must be as flexible as possible and must be based on standards. Typically, individual components are used across OEMs. This means that the OEM in the US has less influence on functionalities or the component manufacturer's development processes. The role of the OEM is often limited to that of an integrator.

In Europe, on the other hand, OEMs offer the vehicles in different variants. Therefore, European customers do not have the option of selecting components like their counterparts in the USA. European OEMs typically use their own chain of fixed suppliers who individually develop or intensively modify ECUs for them. The OEM specifies the entire E/E design and sometimes the development process as well, and they are structured so that individual components can be used in different model series or brands/markets. Standards or open protocols are only needed where external interfaces are provided, e.g. in emissions-related diagnostics (OBD), fleet management (FMS), Toll Collect Modules (OBU) or trailers (ISO11992). This is also the case when components, e.g. the engine, is supplied to other industries such as agricultural or construction equipment. In practice, European OEMs tend to view J1939 more as a "toolbox" and they only use those properties actually needed in the vehicle. A J1939 conformity test would be inadequate for these implementations – but the OEMs do not consider this a disadvantage.

Reducing Development Costs with Standard Software Components

The presentation by Ford [2] indicates that the network architecture is not viewed as a crucial competitive advantage. So, in this area it makes sense to use standardized and largely generated software components. In its FNOS (Ford Network Operating System) initiative in the automotive area, for example, Ford took up the idea of making an implementation available to all of its suppliers. This reduced quality problems and their propagation to a minimum. The commonality between this approach and J1939 is that FNOS is like a standard for the suppliers. In contrast to FNOS, however, many J1939 users implement the standard quite differently. Participants report that this situation always leads to problems in integration. For example, messages might not be received, because sender and receiver priorities do not match, specific ECU addresses are assumed or signals are not fully implemented. Navistar [3] noted that even the exchange of information between OEM and suppliers on which signals are available is a recurring source of errors due to outdated information and incomplete information. Instead of using an available quasi-standard – such as the DBC data format – to describe the CAN communication, text documents are often used.

Ford's experience with FNOS demonstrates that de-facto compatibility increases the availability of products and significantly reduces integration effort. For OEMs, advantages are realized if they do not develop the reference implementation for different hardware platforms themselves. By outsourcing this work to a specialized company – in this case Vector – savings were attained on the order of about 800,000 US-dollars at Ford.

Figure 1: Development models in contrast (green, blue) and a potential merging (red).

Vector CANtech [4] discussed the use of standard software components versus in-house solutions. In particular, the use of off-the-shelf components offers a number of advantages in areas that are not competition-relevant and are not typically core competencies. Purchased and in some cases pre-certified components, such as the J1939 CAN communication software or the operating system, lead to greater assurance in the development process and increase interoperability. Model-based development also contributes toward reducing sources of errors **(Figure 1)**.

Optimizing J1939 Integration

In its presentation, Navistar showed just how all US commercial vehicle OEMs might realize such savings potential [3]. In the context of its "Blue Diamond Program," Navistar acquired experience with both J1939 and FNOS. This involved marrying a Ford cab to a Navistar chassis. A gateway (Blue Diamond Gateway) had to be developed that would act as a link between the FNOS cab and the J1939 chassis. It became evident that the advantages of FNOS could not simply be transferred one-to-one to J1939, but that the underlying principles could. Navistar has identified a number of items that could be improved for future J1939 integration:

> The communication description should be made with an OEM-in dependent data format such as the DBC format. This enables automatic detection of incompatibilities or missing signals **(Figure 2)**.
> The OEM should have more influence in selecting the number of communicated parameter groups and signals. This offers a better way to avoid or entirely eliminate potentially critical latency times or excessively high bus loads.
> Suppliers can continue to write their own communications software. A DBC database is used for improved knowledge transfer related to the communication behavior and monitoring of communication.

Figure 2: Use of standard software components can reduce the share of application-specific software.

Figure 3: Shifting effort to the beginning of the development process reduces risks and integration effort.

> Exchange of simulation models between OEM and suppliers for all communication modules enables simulation and testing of the entire network.

Vector CANtech [5] has analyzed these items from a cost-benefit perspective and with regard to a timeline for their introduction. The use of a common existing data exchange format between OEM and suppliers is highly recommended, since it requires little implementation effort, and the benefits are realized immediately. The use of a conformity test offers additional advantages. It supports the user at various points in the development process, e.g. in implementation, verification and system integration, and it improves both product quality and process effectiveness. Development tools such as Vector's CANoe .J1939 support automatic generation of conformity tests utilizing DBC databases. This addresses critical paths in development much earlier in the process, so that they do not first appear in system integration **(Figure 3)**.

Conclusion

The conference showed that all of the US-OEMs in attendance were working on the same problems, although at different levels of intensity. This fuels the hope that these issues might be addressed as a group. The most obvious and promising approach can be implemented with little effort: "Use of a uniform data exchange format for CAN communication." This offers direct benefits to both OEMs and the suppliers. A reference implementation or official conformity test tool could be defined and implemented in coordination with the SAE organization. Mandatory device profiles could also be established in this framework.

Translation of a German publication in Hanser automotive, 9/2010

Image rights:
Figures 0, 2 and 3: Vector Informatik GmbH
Figure 1: Vector Informatik GmbH based on documents from Navistar [3]

Literature:

[1] Fellmeth, P.; Vector Informatik GmbH: "E/E Development and J1939 in Europe, Overview about Current Status". JEIM Congress 2010.
[2] Paton, E.; Ford Motor Company: "Ford Network Operation System, The OEM Perspective." JEIM Congress 2010.
[3] Venkateswaran, S.; Navistar, Inc: "Blending Automotive and Commercial Vehicle Network Technologies." JEIM Congress 2010.
[4] Stevens, S.; Vector CANtech, Inc.: "Evolution of Vehicle Embedded Software – COTs". JEIM Congress 2010.
[5] Craig, J.; Vector CANtech, Inc: "In-Vehicle Network Development, Best Practices". JEIM Congress 2010.

Links:
Homepage Vector: www.vector.com
Product Information Vector's solutions for J1939:
www.vector.com/j1939

Peter Fellmeth
studied at the University of Applied Sciences in Esslingen, Germany, majoring in Computer Engineering and specializing in Automation Technology. He is group leader and product manager at Vector Informatik GmbH. At Vector, he is responsible for the development of products and customer-specific projects related to J1939, ISOBUS, Ethernet and Car2x.

CAN and J1939 under Extreme Duty Conditions

Vehicle Electronics Guarantees Functionality in Cold, Ice and Snow

Anyone who has participated in winter sports at one time or another is familiar with them: The slope groomers that tirelessly prepare the ski slopes and transport goods to mountain stations or injured persons safely down to the valley. They not only embody a special species of all-terrain track vehicle, they also deliver the experience of true high performance. Vehicle electronics play a decisive role in the incredible capabilities of these machines. Without electronics neither functionality nor safety nor any other significant innovations would be conceivable. This technical article offers insights into the vehicle technology, development process and development tools of the latest generation of PistenBully vehicles from the Kässbohrer Company.

In contrast to conventional off-road vehicles, the technical challenge for the PistenBully is to master the numerous extreme situations encountered in cold, snow and nighttime operation. The machine, capable of moving in any direction on inclines up to 45°, covers an area of 96,000 m²/h with its multiflex tiller. This technology is standard equipment for duties up to 4,000 m above mean sea level and extreme outside temperatures; the elevation capability of the polar version even reaches up to 6,000 m.

Greatest Safety under Extreme Conditions

Slope grooming is often scheduled for evening and nighttime hours, while there are no regular winter sports activities. If vehicle drivers are underway alone in a snowstorm or fog at high elevations or in Arctic regions, the reliability and availability of the vehicles can be life preserving factors. Safety as well as comfortable and fatigue-free steering and controls were therefore key aspects of the vehicle concept. Intelligent automatic functions support the driver and reduce the number of control interventions needed, so that the driver can concentrate on what is important.

Impact and puncture resistant windshield glass protects the driver from rock impacts, and a lighting system with a wide array of running lights, searchlights and working lights turn night into day. Automatic integration of a rear camera image in the cockpit display also provides optimal visibility when driving in reverse. To support grooming on steep inclines the vehicles can be optionally equipped with electro-hydraulic cable drums that carry 1,000 m of cable. A special feature is free rotation of the drum, allowing the vehicle to turn 360° as often as desired. Besides models for

Figure 1: Up to 620 PistenBullys are produced in Laupheim every year

use on mountains and snow, there are also PistenBully versions without a loading platform, versions exclusively used to transport personnel, excavating versions and even versions that can swim. Kässbohrer produces about 600 to 620 units per year, and the cost per vehicle lies between 80,000 and 340,000 euros.

Power for Driving, For Ski Lifts and Emergency Electrical Generators

The vehicles are driven by engines from Mercedes-Benz with power ratings between 90 HP and 460 HP. The latest PistenBully 600 has a standard 12.8 Liter engine delivering 295 kW (400 HP) and torques of up to 1,900 Nm. Two independent hydrostatic drive circuits without separate clutches are responsible for tractive power to the right and left sides. The engine controller delivers the required engine torque when starting up from a stop and prevents adverse events such as stalling. Simultaneously, load limit control offers protection against overloading and over revving of the diesel engine.

A single gas pedal is used to accelerate and decelerate (braking), i.e. there is no working brake, only a parking brake. Changes in driving direction are achieved by differential track speeds. Each drive has infinitely variable output speed adjustment and its direction of rotation can be reversed. As a result, the fully electronic steering can support turning in place, pre-selection of driving direction and speed reduction. The "steering aggressiveness" varies with driving speed; the driver can adjust it to his or hers specific needs. This means that the driving speed can be changed by gas pedal or load limit control without affecting the turning radius. Wheel sensors enable straight-line and uniform curve control or asymmetrical steering characteristics for special applications.

Nothing Runs without Vehicle Electronics

Electronics is or at least was often considered a necessary evil by conventional machine building companies. The Kässbohrer Company, whose origins are in the steel building industry, has come full circle in its perspective here. Without electronics any meaningful interplay between complex vehicle components would be inconceivable. Vehicle electron-

Figure 2: Overview of CAN networks in the current PistenBully

ics is ever present, from steering, control of the engine and hydraulic system, conveniences of vehicle navigation and the control of implements, to functions for operational data acquisition, telematics and GPS.
Consistent and thorough networking of the vehicle by CAN (Controller Area Network) was a prerequisite for achieving a decentralized control structure. This made it possible to rationally combine electronic and mechanical components to make one control module. In comparison to earlier PistenBully generations, decentralization has enabled significant reductions in wiring costs. Nearly all communication is routed over the two primary buses CAN1 and CAN2 (of a total of five CAN bus lines). While CAN3 is used for external communications in fleet management, the CAN4 and CAN5 systems are reserved for future functions not currently needed, and they are technically ready to implement them. Additionally, CAN is utilized for software updates, parameter configuration and in measurement systems. Since all functions can currently be implemented with CAN, the use of newer communication systems such as FlexRay, LIN or MOST is not currently under discussion. The electrical system is set up to be fully modular and is uniform across all vehicle variants. Since the basic wiring already considers all current and future options, it is easy to accomplish expansions and retrofits by means of adapter wire harnesses.

Lots of Power Electronics, just a Few Fuses and no Relays

The PistenBully universal PSX control module is responsible for central control of all functions such as performance and power management, engine control, hydraulics of the drive and tilling pumps, oil flow distribution of the working hydraulics, front and rear, as well as monitoring of all sensors and actuators. It is supplemented by the "central electronics" unit, which besides offering numerous diagnostically capable and short-circuit protected inputs/outputs, also houses other functionality such as central locking, RF remote control, lighting control and voltage converters for 12 V. The full load capable unit supplies a summed continuous current of 640 A at 24 V and thereby achieves a switching capacity of up to 15 kW. The "Central electronics" unit has connections to all five CAN buses. In total there is need for just eight "actual" fuses; everything else has been implemented to be short-circuit protected and "self-healing" without relays.

Figure 3: Controls and display components in the PistenBully cockpit

Maneuvering Is Easy and Intuitive

Connected to the internal vehicle bus (CAN1) are the controls and display elements of the cockpit. In addition to the ergonomic semi-circular steering wheel, also includes a multifunction display with touchscreen, round CAN instruments, a Terminal Control Center (TCC) integrated in the arm rest and a joystick with programmable function buttons. The multifunction display gives momentary information at a glance on the most important operating states and on driving speed, cable drum tension, engine data, etc. It offers intuitive control, on the touchscreen or via the TCC. The operating panel mounted to the right of the driver with its film keypad lets the user select numerous Pistenbully functions directly. A joystick is used to control the various movements of the snow blade and of the rear implement carrier as well as used to set the tilling pressure, cable drum tension, etc. The joystick is an in-house development, since none of the commercially available models could satisfy the expectations of Pistenbully developers.

CAN Controls Hydraulic Module

CAN2 serves as a sensor-actuator bus for engine control and valve control and an interface for sensors. The hydraulic valves are driven entirely via CAN, i.e. without supplemental analog or digital I/O signals. On the part of the sensor/actuator bus, so-called multi-modules provide input channels, digital outputs, PWM outputs and bridge outputs that are diagnostically capable, short-circuit protected and self-healing. The sensors connected here are all equipped with 4 to 20 mA current interfaces to automatically compensate for contact resistances when electrical connections corrode.
While Kässbohrer utilizes its internal KGF protocol for the CAN1 functional bus, the J1939 protocol is used on CAN2 for the drive system. The advantage of standardized drive management based on SAE J1939 is that the drive system can be built with components from outside suppliers, independent of the vehicle producer, including a suitable diagnostic system. On the functional side, the proprietary protocol was used intentionally to prevent unauthorized manipulations and simultaneously to protect know-how. That is why it was also decided not to use the CCP (CAN Calibra-

Figure 4: CANoe as joystick simulator for testing hydraulic valves

Figure 5: CANoe in flash mode

tion Protocol) standard for the ECU application. The CAN bus systems can be parameterized and diagnosed from a laptop.

Safe Return to the Valley even if the Bus Fails

It is interesting that X-by-wire systems have already been used in PistenBully production vehicles since the 1970s, while its implementation in general road vehicles was not even a topic of discussion until decades later. Entirely different legal regulations apply to the operation and safety of pure off-road vehicles not subject to highway vehicle registration, since the vehicles are used exclusively on private land. If there is failure of the steering potentiometer, driving direction pushbuttons and/or the redundant gas pedal with a contact less sensor, driving capability is preserved as long as possible. The vehicle, weighing in at between 7 and 10 metric tons and capable of a maximum speed of 23 km/h, can then be driven safely back to the valley with emergency running characteristics at a throttled-down speed of 5 km/h. Even if both CAN buses fail the PistenBully remains maneuverable with control via PWM signals.

For the electronics, besides satisfying requirements for harsh temperature and humidity conditions and mechanical stresses, other special requirements also apply, e.g. with regard to electromagnetic compatibility. This ensures that the high field strengths of radio transmitters on mountain peaks will never impair vehicle functions.

From Simulation to Real PistenBully Electronics

Software development and vehicle calibration covering all control modules are all performed at the Kässbohrer Company. This lets the producer from Laupheim adapt flexibly to new operating situations. Since the complexity of the software and the electronic functions is constantly growing, developers on a project like the PistenBully must rely on powerful tools for software development too. Over the course of the development process it is essential to perform design functions, tests and simulations of subsystems and overall systems. This is where CANoe with the J1939 Option from Vector comes into play.

CANoe's capabilities as a development and simulation tool range from simulation of a single network node, to testing and diagnostics, to representation of complete CAN networks. Beginning with initial studies on the purely virtual model, the virtual nodes can gradually be replaced by real hardware step-by-step over the further course of development. In this case, vehicle functions are executed by a virtual ECU in an OSEK emulation. Among other things, this makes it possible to control and display the states of virtual sensors and actuators. It is also possible to generate associated control panels automatically.

Short Development Times

At Kässbohrer some of the uses of CANoe are to simulate bus loads, function as a measurement tool and parameterize ECUs by the proprietary KGF Protocol. The developers use this protocol to generate diagnostic and setup information in temperature tests, EMC tests and reaction tests of valve controls, for example, and this helps them to keep solutions for production vehicles lean.

The development of dual-channel fan control in the PistenBully was completed in an exceptionally short time without utilizing real hydraulic pumps, valves or motors. In such cases CANoe realistically simulates all necessary CAN nodes, sensor signals and ECU information. In ECU setup CANoe enables access to EEPROM contents via an in-house flash protocol. This is easy to reprogram with the integrated programming language CAPL (Communication Access Programming Language). The EEPROM data stored on the hard drive can be loaded into the controller at any time.

Figure 6: CANoe in measurement data acquisition during vehicle trials

Indispensable: Real-time Capability of Development Tools

An employee explains: "As developers we rely on good tools, and CANoe's reliability is excellent! Real-time capability is especially important to us. We have already had negative experiences with similar software on another project. It took days of troubleshooting to finally find out that the tool simply could not keep pace with our sampling rate requirements, and this led to erroneous results."

In-house user interfaces, panels and other tools that are frequently needed are generally programmed in-house at Kässbohrer using Borland C++. Even in such custom developments CANoe databases always serve as the foundation. Moreover, CANoe facilitates optimal cooperation with suppliers since the behavior of assemblies can be tested in advance. The PistenBully developers only regret that not all system suppliers provide CANoe simulations of their products or make them available early on.

Mobility for Fine Tuning on the Slopes

Another important aspect is tool mobility. Since snow conditions are constantly changing, fine tuning of the drive system is often performed locally in the mountains. When it is important to perfect control loops for the various types of slope preparations and adapt them to local conditions, CANoe operated on a laptop proves to be an efficient mobile diagnostic and measurement laboratory.

Comprehensive Vehicle Diagnostics on a Display Screen

The vehicle can be fully parameterized at any time via a programming PC that is connected to the CAN diagnostic connector. For every PistenBully there is an electronic vehicle record that seamlessly documents software updates, the life of individual components, current software levels, etc. It is possible to restore the delivered state at any time. If problems occur at the customer, On-Board Diagnostics offers fast and convenient fault localization over the cockpit display. All hydraulic functions, sensors and actuators are designed to be electronically diagnosable. All that is needed for in-vehicle troubleshooting is a circuit diagram and the display; no other equipment is needed. Stored in the fault memory are the error history and error frequencies.

Image rights:
Kässbohrer Geländefahrzeug AG

Thomas Böck (Graduate Engineer)
studied general electrical engineering at the technical college in Kempten, Germany. He manages development in the electronics/hydraulics area.

Peter Betz (Graduate Engineer)
studied telecommunication engineering at the technical college in Ulm, Germany. He is responsible for system development in the electronics development area.

Markus Hörmann (Graduate Engineer)
studied telecommunication engineering at the technical college in Kempten, Germany. He manages test equipment construction in the electronics testing area.

Lothar Felbinger (Graduate Engineer)
studied automation engineering at the technical college in Reutlingen. Since then he has been working as Key Account and Business Development Manager at Vector Informatik GmbH for the Open Networking product line.

Figure 7: Easy fault localization for the driver with OBD

Current Trends in Network Development for Heavy-Duty Vehicles

Factors of Success in Electronic Development

ECU networking in heavy-duty vehicles is characterized by the same challenges as in the automobile. Added difficulties are caused by the large numbers of variants with low production volumes and longer product life cycles, requiring a suitable architecture layout. Specially modified development methods are indispensable in handling cost pressure and sending reliable vehicles onto the street.

The number of ECUs, and hence the amount of software, has multiplied since electronification began in the early 1990s. While this primarily related to the engine controller at the beginning, a large number of electronic "helpers" are being implemented today. Examples include ABS, ESP, ACC and other driver assistance systems that make highway traffic safer and driving more pleasant. Analyses [1] assume that their implementation will increase further, and that electronic control modules will account for 90 % of all innovations by the year 2010. A key aspect is that 80 % of these innovations will exclusively involve software or the functions implemented in software. In this context, it is clear that software development methods play a crucial role in the development process for the total vehicle, and they have a significant influence on a vehicle's success or failure on the market.

Compared to automobiles, heavy-duty vehicle manufacturers are confronted by the special challenges of the relatively large number of variants with significantly lower volumes. Although simultaneous use of electronic ECUs over different brands and integration of standardized components can reduce cost pressure, they make the design of electronics and software more complex.

Flexible Solutions Are in Demand

When one considers the variety of strategies used by different heavy-duty vehicle manufacturers, it quickly becomes clear that there is no universal solution. However, from a bird's eye perspective clear trends can be seen, such as the use of standards, code generators and a universal tool chain. The number of ECUs is increasing at a rather moderate rate, while the number of functions implemented purely in software continues to grow rapidly.

Common to solution strategies is the use of a comprehensive and universal tool chain – from requirements to validation. The use of individual, non-coordinated tools proved to be impractical in the past. The configuration processes and work results of individual tools are too different. This makes it difficult to achieve universality of change requirements during development. Thus, a change would need to be made in different tool configurations without any automatic, inter-tool consistency check. This causes organizational friction losses, especially in inter-departmental or inter-site development projects.

Therefore, a database with authoring tools should stand at the center of the tool chain. Both the database and authoring tools need to be specifically adapted to the requirements of the specific vehicle manufacturer. Besides purely technical aspects, the tools also take the individual development process of the companies into account. Variants management, configuration management and even the maintenance of workflows are represented in the tools. If external suppliers need to be integrated in processes, the data exchange formats that are used may be standards or de-facto industry standards such as the CANdb++ data format by Vector Informatik. In some cases, the vehicle manufacturer also prescribes the use of certain tools to its suppliers. They are then tightly coupled to the database and support the supplier especially in such aspects as compatibility to requirements, quality and efficiency. Examples would be code generators for embedded systems or test tools such as the CANoe .J1939 development and test tool from Vector.

System design is becoming increasingly complex due to growing requirements for networking. Individual ECUs are being installed on different platforms and different countries, which increases the number of variants. This requires flexibility in communication structures and in mapping functions to ECUs. This not only affects the available signals, but also the use of communication protocols. In Europe, for example, proprietary communication protocols are often used, which is similar to the situation in the automotive industry there. In the North American market, however, the SAE J1939 protocol dominates for heavy trucks. There are also differences in the area of in-vehicle diagnostics: In Europe, OBD diagnostics is implemented per UDS (ISO15765), and in the USA per SAE J1939-73.

Attaining the Goal by Different Approaches

The approach at MAN Nutzfahrzeuge AG is based on use of an integrated development database known as the "Common Engineering Data Backbone". All vehicle-specific functions are developed from this platform, and all vehicle-specific information is stored there. The eASEE Tool Suite from Vector – with its 8 domains – serves as a universal development database, and it was specially adapted to requirements at MAN in the framework of a configuration process **(Figure 1)**. It serves the needs of functional development and as a description of the communication matrix. Since MAN relies on the SAE J1939 standard as much as possible in communication, eASEE was adapted to the requirements of the J1939 protocol.

A special module that was developed for MAN and adapted to the Data Backbone serves as a bridge between modeling in eASEE and automatic code generation for the ECUs **(Figure 2)**. In code generation, the Munich-based heavy-duty vehicle producer relies on proven CANbedded J1939 standard software components from Vector. CANbedded J1939 gets all of the information it needs for configuration and code generation directly from the database, and it can generate the embedded code without manual interventions. This enables immediate transfer of changes made in modeling to the ECU code. This process prevents errors such as incorrect configuration of the code generating tool and guarantees error-free and complete code generation. This process also simplifies verification of the total system, since sections of the software have already been checked. It is possible to reuse the communication data for analysis tools like CANalyzer .J1939 or test tools like CANoe .J1939 from Vector, supporting the development of application layers.

Figure 1: The MAN Common Engineering Data Backbone

The Volvo Truck Corporation chose a strategy for software development that has now become established in the automotive industry too: the use of AUTOSAR and its overlying tools **(Figure 3)**. The benefits of this approach lie in the use of standardized and proven tools. They offer benefits in a development used intensively across brands that is distributed among many business sites. Common understanding of the underlying software structures and architecture is quickly achieved. It is easier to integrate suppliers, and it is not absolutely necessary to specify tools. This reduces dependencies on individual tool producers and suppliers.

Problematic in this approach is the use of communication methods that are either incompatible with AUTOSAR properties or can only be used with it in a proprietary way. The use of J1939, in particular, should be mentioned in this context. While AUTOSAR essentially assumes a network of known nodes – and therefore a communication matrix that is known at the time of integration – this is decidedly not the case for J1939 with its plug-and-play concept. Volvo Trucks confronted this challenge with a two-prong approach. The first step was to identify which parts of J1939 were used in Volvo vehicles and integrate them in the existing Vector AUTOSAR tool chain. Secondly, Volvo – together with Vector and other European heavy-duty vehicle manufacturers – adopted portions of the J1939 protocol in AUTOSAR. This strategy lets Volvo exploit the advantages of AUTOSAR directly and universally. On the other hand, AUTOSAR integration of J1939 makes it possible to achieve fundamental independence in tool selection. Volvo chose Vector as its supplier of tools and embedded software components, since Vector already offers solutions in all areas, and it was possible to adapt them to Volvo-specific requirements very flexibly.

Image rights:
Vector Informatik GmbH

Literature references:
[1] J. Svensson, "The Use of AUTOSAR in Volvo Group", presentation at Vector J1939 User Day; slides may be downloaded at: www.vector-informatik.de/j1939ud [most of them are German]

Peter Fellmeth
is a team leader and product manager at Vector Informatik GmbH, where he is responsible for the development of products and customer-specific projects related to J1939, ISOBUS, Ethernet and DeviceNet.

Figure 2: Code for the ECUs is generated based on the description of the electronic structure in eASEE's functional data management.

Figure 3: When standardized data formats are chosen, standard products can be used to describe and create the ECU-specific software.

VECTOR

The reliable, efficient way to design and test networks

Solutions for Commercial Vehicles

Development processes for open networks in commercial vehicles can be implemented in the most cost-effective approach with minimal time using specialized assistance:

> Develop your network design in a systematic approach using reliable tools
> Successfully implement software components for both proprietary and standardized J1939 and AUTOSAR protocols
> Benefit from efficient configuration and extensive diagnostic, testing and analysis solutions

Information and downloads: **www.vector.com/j1939**

The seamless interaction of Vector tools and Vector's comprehensive services will improve the efficiency of your entire development process, from design to analysis.

Vector Informatik GmbH | Germany · Austria · Brazil · China · France · India · Italy · Japan · Korea · Sweden · UK · USA | www.vector.com

Automatic Interoperability Tests for ISO11783 Systems

Universal Test Solution Assures Compatibility

The required "unlimited compatibility" of components on the ISOBUS cannot be attained by performing conformity tests at the end of device development alone. Rather, sound and continual tests over the entire development phase are necessary. Efficient use of such tests can only be achieved by using tools with domain knowledge that cover a large number of tasks ranging from simulation to analysis tests as well as conformity tests. Developers of implements and tractors need a tool that covers conformity testing, cycles through the tests independently, yet offer the freedom to only test certain sections and can be extended to test the application.

In the agricultural machinery field, the information age has long taken hold, and system thinking has replaced insular solutions. As a result, a uniform data interface for interconnecting the tractor, implements and on-board computer has become indispensable in agricultural equipment. In this context, the internationally coordinated bus system ISOBUS was developed and introduced for the first time at the 2001 Agritechnica fair. ISOBUS standardizes data communication between tractor, implements and farm management computers and enables system-wide data exchange. The ISO11783 series of standards consists of 14 sub-standards; they each address different aspects of the technology, ranging from System Description (Part 1), Physical Layer (Part 2), Data Link Layer (Part 3), Network Layer (Part 4) and Virtual Terminal (Part 6), Diagnostics (Part 12) and File Server (Part 13).

"One person's pleasure is another's pain" is common folk wisdom. The situation is similar with the requirement for unlimited compatibility (system and manufacturer independent) of ISO11783-compatible products [1]. For the customer this is not only very easy to handle, it also opens up the possibility of purchasing flexibility and independence from the manufacturer. That in itself is a large motivational factor in procuring such machinery. For manufacturers, however, this promise represents a great challenge in terms of development, operation and maintenance of the machines. CANoe .ISO11783 from Vector offers a universal development and testing solution here. Option ISO11783 for the CANoe tool provides the necessary domain knowledge and supports conformity to the ISO11783 standard **(Figure 1)**. Experience over the past two years in the ISOBUS field has shown that despite a sharply rising number of devices certified by conformity tests [2], different components, such as the Task Controller and implement, do not always harmonize in their interaction without problems. There is the potential for surprises in operation of an implement when the Virtual Terminal is used too. As well as for service technicians, in such a heterogeneous environment as the ISOBUS it is difficult to definitively localize the cause of a problem and correct it if necessary. Frequently, the technician is confronted with unfamiliar devices or combinations of devices. In view of these problems, and to assure customer satisfaction, the manufacturer's initiative AEF (Agricultural Industry Electronics Foundation) set up a project group tasked with conducting activities to improve the interoperability of ISOBUS devices [3].

Uniform Diagnostic Access for the Worst Case

In the framework of standardization tasks, along with continual efforts to refine and extend test cases of the conformity test, Part 12 of the ISO11783 draft standard [4] was written to create a common diagnostic interface. It is based on SAE J1939-73 diagnostics [5]. The section of Part 12 of the ISO standard that has already been published, what is referred to as basic diagnostics, defines open diagnostic access. It provides basic functionalities and is intended to enable a system overview. This includes unique identification of the ECUs on the bus as well as information on the software version, manufacturer's part number and the conformity test performed. Each ECU can report momentary errors, and when requested by the diagnostic tool it can also report previous errors. This information is intended to enable quick and reliable localization of the error causes. This is especially advantageous if a network consists of components from different manufacturers. For example, the tractor manufacturer's service technician can use an ISO11783-12 compatible diagnostic tool to detect problems related to an implement from a different manufacturer. The technician might not be able to correct the problem, but can clearly identify its cause. If the cause lies in the implement, the tractor manufacturer's service technician – who was summoned by mistake – can call up valuable information such as error codes or part numbers of the affected components to give the implement manufacturer's service technician advance information on the problem. This keeps downtimes to a minimum and leads to a higher level of customer acceptance of ISOBUS-equipped machinery.

Ongoing efforts to extend Part 12 of the ISO11783 draft standard are taking the direction of a standardized description format for diagnostics. This would let each manufacturer describe diagnostic contents individually for each ECU. A prepared diagnostic application could use this description to diagnose the ECU regardless of which company manufactured it. The diagnostic description file might be downloaded from the ECU itself or over the Internet. Manufacturers with their own company-specific diagnostic tool would integrate ISO11783 diagnostics into their existing tool. Manufacturers without their own custom tools could use future standardized diagnostic tools. One practical benefit is that a service technician would have system-wide diagnostic capability with just one tool. This en-

Figure 1: CANoe .ISO11783 can be used to utilize, analyze and simulate the complex communication structures of the ISOBUS standard easily and efficiently. Shown here is simulation of the File Server, two implements and the Virtual Terminal.

ables efficient and reliable localization of the real causes of errors and ideally to correct them right away.

Automated Tests during the Development Phase

Introduction of uniform diagnostic access helps to quickly identify a problem on-site and possibly replace the defective part. If there is some incompatibility, however, the situation is generally very different. Replacing the electronics does not offer any help, since this does not correct the cause of the problem. In such a case, corrected ECU software would be necessary. However, it would take time to produce and test this software. In addition, distribution of the modified software is often costly, since devices must be recalled from the field. Suitable preventive actions can be taken to avoid such compatibility problems. One option is the conformity test mentioned in the introduction. However, a disadvantage here is that the application itself is not part of the test. The focus in conformity testing is to test conformity to the standard. In addition, it is difficult or impossible to use the conformity test during development, since 100 percent compatibility cannot be assumed at the beginning of development. Often just point checks of a certain aspect are desired, e.g. a check of the Transport Protocol. Such tests conducted over the course of development are generally performed very frequently, they have many alternative test sequences, and they must be flexible in their configuration. Therefore, such tests should be designed for automation. If problems occur during the test, extensive analytical capabilities are needed as well.

One Tool for All Cases

CANoe .ISO11783 from Vector is a universal development and test solution that is used to verify conformity to the ISO11783 standard, providing the necessary domain knowledge [6]. The Virtual Terminal, for example, is a fixed component of CANoe **(Figure 1)**. Diagnostic messages can be visualized using the Diagnostic Trouble Code Monitor and Scanner. The integrated Test Feature Set makes it possible to define frequently recurring tests and entire test sequences. The test sequences can be easily defined by XML, for example. **Figure 2** shows a schematic representation of the Test Feature Set. In such an environment, CANoe .ISO11783 may assume the role of the test master and link to or drive other tools over various interfaces such as COM or .NET. It is also possible to integrate CANoe .ISO11783 in an existing test environment via the same interfaces. Because of its extensive simulation and analysis capabilities, its use is not limited to just testing or simulation of individual ECUs. The tool can simulate entire networks (simulation of the remaining bus). For example, operation of an implement could be simulated via a Task Controller or the Virtual Terminal. With the help of Vector test hardware VT System, CANoe .ISO11783 also directly drives real consumers such as actuator motors and an ECU's outputs, or reads in sensors **(Figure 3)**. CANoe .ISO11783 can be used to utilize, analyze and simulate the complex communication structures of the ISOBUS standard easily and efficiently. It is a comprehensive, universal tool for verifying conformity over all product phases: from development to operation and service of the working machinery.

Figure 2: Phases of Test creation and interfaces with the CANoe .ISO11783 Test Feature Set: from creating the test sequence to evaluating the results.

Figure 3: Used in tandem, CANoe .ISO11783 and the Vector VT System form a "Midsize HIL".

Image rights:
Vector Informatik GmbH

Literature and links:

[1] VDMA Professional Society for Agricultural Engineering: ISOBUS speaks all languages. Just speak along with it.2005, pg. 2
[2] VDMA Professional Society for Agricultural Engineering: ISOBUS-conformant devices per the ISO 11783 standard, www.isobus.net/isobus_E
[3] www.aef-online.org/
[4] Society of Automotive Engineers: J1939
[5] International Organisation for Standardization: ISO/FDIS 11783-12, ISO11898
[6] www.vector.com/isobus

Peter Fellmeth
studied Computer Engineering at the University of Applied Sciences at Esslingen, Germany, specializing in Automation Technology. He is a team leader and product manager at Vector Informatik GmbH, where he is responsible for development of products and customer-specific projects related to ISOBUS, SAE J1939, Ethernet and DeviceNet.

Forging New Pathways in Testing ISOBUS Task Controllers

Simulations Replace Inflexible and Time-intensive Test Methods

The inter-system and inter-OEM compatibility of ISOBUS-conformant devices lets farmers interconnect tractors and implements from different manufacturers in any desired combinations. As easy as this may seem from the user's perspective, the level of effort required in the device development side is high, especially in the testing phase. A look at John Deere shows that conventional industry methods for testing electronic components are now often running into their limits. An incomparably faster and more efficient means for attaining the desired goal is automatic test sequences with a simulated implement environment.

The tractor is a truly multi-talented field machine due to its ability to interact with many different implements. As long as the tractor remains a machine that offers pure pulling and power take-off functions, and no other interaction occurs with the implement, operation of the two devices is relatively uncomplicated, but it is not very efficient due the lack of an additional interface. Nonetheless, modern agriculture requires intelligent, automated solutions, which support such functions as variable spread quantities for seeds and documentation of the work performed on the field.

Yet, as complexity grows and more intelligent functions continue to make in-roads into agricultural technology, this increases the effort needed to keep operation and handling as simple for the farmer as it was before. Long, drawn out start-up processes are counter-productive, not least in terms of customer acceptance of modern agricultural technology. It must be possible to rapidly and smoothly connect any of a wide variety of implements to the tractor and have the tractor's electronics 'understand' it immediately.

ISOBUS Compatibility and Conformity are Top Priority

The ISOBUS application plays a key role here. ISOBUS was created so that tractors, implements and operator terminals could communicate with one another and exchange data. The technical details are defined in the ISO 11783 series of standards, and they cover all topics from the ISO reference model to diagnostics to the file server. To ensure smooth interoperability of devices from different manufacturers, extensive tests are indispensible for both tractor and implement producers. Along with the obligatory conformity tests, development departments must play through numerous other test scenarios. In addition, manufacturers regularly organize "Plug Fests" where they verify that their product functions are field-ready in their interactions with the products from other manufacturers.

One of the forerunners of modern agricultural technology is John Deere, a company with a long tradition. Its products range from tractors to field sprayers and balers to seeding, harvesting and chaffcutter machines. Not only does it develop agricultural machines, but also construction machines, forestry machines and public utility equipment as well as machines for lawn, property and golf course maintenance. In addition to its German subsidiaries in Zweibrücken, Mannheim and Bruchsal, the American agricultural machine specialist opened another business site in Kaiserslautern in early 2010. Employees in the new European Technology and Innovation Center (ETIC) work on future technologies and bring associated products to production maturity together with development departments at other sites. Precision farming, the integration of intelligent technologies in machines and agricultural electronics, represents a focal point of work at Kaiserslautern.

From Farm Computer to Automatic Section Control in Implements

The concept of Precision Farming illustrates current trends in agricultural technology and puts them in focus. The goal here is to attain the greatest possible yield and maximum economy by optimal use of all available resources such as machines, seed stock, fertilizers, fuel, time, etc. The farmer takes the parameters of the planned field operations on the farm computer and uploads them to the operator's terminal in the tractor by memory card or USB stick, or in the future via WLAN.

Telematics and satellite navigation also make important contributions in combination with steering and track guidance systems as well as section control. The result is seamless application of seed stock and fertilizers without any areas of faulty application. At the same time, the technology provides for minimal overlaps on wedge-shaped fields and saves raw materials at field borders. Implements with section control are subdivided into multiple sections, which can be activated or deactivated independently of one another. Since all activities are logged, movements of the tractor during which the implement either protrudes beyond field boundaries or overlaps already covered areas result in automatic deactivation of the relevant sections.

The Task Controller as an Interface to Device Control

These and other functions mean that the tractor electronics must have a precise knowledge of the implement's technical data and functions. The ISOBUS operator's terminal, as a part of the tractor electronics, is, in many cases, not just a user control and display system, but a minicomputer on which multiple applications run simultaneously. Such an application is the Task Controller, which is described in ISO 11783 Part 10. Ideally, it simultaneously serves as a documentation and control system with an interface to the Farm Management System via the TaskData.xml file. In the John Deere GreenStar 2630 display, the Task Controller represents an interface between the John Deere documentation system and an ISOBUS implement. The first time it is connected, the Task Controller loads a "Device Description File" from the implement's job computer. This device description file generally contains all information necessary for the Task Controller, such as the implement's working width, type of mount to the tractor and number of switchable sections with associated element numbers, if it is an implement with section control. The implement can be operated via the tractor's operator's terminal **(Figure 1)**.

The Task Controller must seamlessly master the entire bandwidth of possible implement device configurations. Only then is it assured that the ISOBUS operator's terminal

Figure 1: John Deere ISOBUS operator's terminal with the Kverneland fertilizer spreader operating interface

and its applications will work properly together with any conceivable ISOBUS implement on the market. However, every work machine operates differently than another and uses a different combination of Task Controller functions. For test purposes, producers therefore exchange special hardware boxes, in which the electronic functionality of their implement product is represented. To the chagrin of test engineers, aside from the ECU hardware and software contained in the boxes, they very seldom included all of the components needed to conduct a comprehensive functional test of the device logic.

Seeking a More Efficient Test Method

In Kaiserslautern as well, employees were using test boxes and real devices of various implement producers to test the functionality and compatibility of their Task Controller. Considering the large number of field implements and outside companies, this is a very tedious and time-consuming process. Theoretically, it would be necessary to test every ISOBUS machine, whether designed for seeding or planting, every fertilizer, every plant chemical sprayer and all other machines with every Task Controller version. In addition, the test boxes are not standardized with regard to their layout or handling. Each company follows a different operating philosophy, and some boxes are pure simulations, while others largely match the real electronics. Before test personnel can actually perform their work, they must first study many different user manuals to gain familiarity with numerous virtual control elements and functions.

This approach, which is indeed typical in the industry but is highly inflexible and unsatisfactory, motivated John Deere to seek out a more efficient test method. Test engineers found the solution in CANoe .ISO11783, a development, test and simulation tool from Vector that was precisely tailored to ISOBUS requirements. CANoe .ISO11783 assures ISOBUS conformity in developments: from the initial phases of product development to the test phase and maintenance. The complex ISOBUS communication structures can be analyzed, visualized and prepared in a wide variety of ways. Functions such as the "Virtual Terminal" and "Interactive Task Controller" simplify working with ISO 11783 for the developer. For example, the CANoe Terminal – unlike a real terminal – can be used to simulate different display types, resolutions or black/white settings. The "Interactive Task Controller" lets users load a device description from any real ISOBUS machine, or it can be used to verify simulators before they are used for testing.

Greater Test Coverage in a Shorter Time

To preserve their independence from implement producers in testing the Task Controller, test engineers at John Deere especially make use of the tool's simulation capabilities **(Figure 2)**. Not only can CANoe simulate individual ECUs, it can also simulate entire networks. Developments can only be tested reasonably and realistically if the later environment is either entirely present or is generated by rest-of-bus simulation. In the case of the Task Controller, in a wide variety of implement variants can be simulated. For example, John Deere can act entirely independent of external manufacturers, and it no longer needs to rely on the physical hardware boxes. A valuable tool for defining automated and recurring tests is CANoe's integrated "Test Feature Set". The system can act as either the Test Master or be inserted into existing test environments. Interfaces such as COM or .NET are available for control and communication with other tools.

Figure 2: Simulation of tractor and fertilizer spreader in CANoe .ISO11783.

The flexibility of the simulations offers considerable relief to John Deere, as is illustrated by the example of section control: simulations make it possible to vary the type and sizes of working machines with little added effort, e.g. to check whether the Task Controller could handle 16 instead of 8 sections. Implements can also be defined whose sections are not strictly adjacent, but instead are offset in back of one another. Since CANoe .ISO11783 represents the standard comprehensively and completely, the agricultural specialist attained a higher level of test coverage in a shorter amount of time. This was especially true in application situations that are either unsupported or just partially supported by the hardware boxes. Such situations include tests intelligent control of driving speed, checking for correct handshakes or simulation of errors, e.g. when an implement does not signal its readiness for section control.

At John Deere's Technology and Innovation Center, CANoe .ISO11783 not only serves to simulate externally produced work machines; it is also used for the company's in-house development of ECUs. In testing, either the real tractor hardware can be used, or it too can be simulated. Since there are sometimes multiple versions of a Task Controller, each needing to be tested, users can quickly toggle between different variants to run. CANoe offers another advantage in distributed development tasks at different company sites. The simulation configurations can quickly and conveniently be exchanged between different departments or even sent to colleagues in the USA via the company's intranet or by e-mail.

Covering Future Requirements

It is no longer reasonable to expect that the complexity of the ISOBUS and the variety of implements available on the market today can continue to be mastered with old, passed-down development and test methods. Taking their place are development, test and simulation tools such as CANoe .ISO11783, which provide for the greatest possible compatibility to the standard in all product phases. The tool's multibus capability enables troublefree display and interpretation of ISOBUS and J1939 messages in a Trace Window. Since the tool covers the full range of ISOBUS functionality and is always at the latest revision level, John Deere attains better test coverage at lower expense in terms of time and personnel. At the same time, the agricultural machine specialist not only has the ability to test extended Task Controller functions, but to also simulate the counterpart device at any time, for the latest and future developments of the ISOBUS standard.

Interesting in this context is the ISOBUS Multiple Product Implement Simulator **(Figure 3)**. A multiple-product implement might be a corn sowing machine with under-root fertilization. It enables simultaneous sowing and spreading of solid fertilizer. One of the benefits, besides time savings, is reduced soil erosion, because the tractor only drives across the field once instead of multiple times. At the time of testing in early 2011, there was still no implement producer that offered such ISOBUS machines on the market. Therefore, it was only possible to have the Task Controller support such machines in a simulation. The full potential of simulations has hardly been exhausted by the basic in-house application. From the perspective of John Deere employees, it would be desirable if manufacturers would exchange their CANoe simulations instead of the inflexible, expensive and difficult-to-reproduce black boxes. The fear that this would somehow reveal internal know-how is unfounded, since it is easy to share the compiled simulations without the source, thereby preserving internal know-how.

Figure 3: Support of an ISOBUS Multiple Product Implement by the Task Controller via a simulation

Translation of a German publication in Elektronik automotive, 11/2011

Image rights:
John Deere

Links:
Homepage John Deere: www.deere.com
Homepage Vector: www.vector.com
Product Information CANoe .ISO11783: www.vector.com/canoe.iso11783

Alexander Ostermüller, John Deere
is a test engineer at John Deer's European Technology and Innovations Center (ETIC) in Kaiserslautern.

Peter Fellmeth, Vector
is a group leader and product manager at Vector Informatik GmbH. He is responsible for the development of products and customer-specific projects in the ISOBUS, J1939, Ethernet and Car2x areas. He is an active member of various standardization working committees for ISO 11783 (TC23/SC19/WG1) and SAE J1939.

Better Test Quality by Automation

Automated HIL Test System Ensures ISOBUS Functionality of Agricultural Machines

The ISOBUS communication protocol has now essentially made it possible to use agricultural machines from different manufacturers together – for the most part without any problems. However, this accomplishment is associated with enormous development and testing effort by the manufacturer. The agricultural equipment manufacturer AGCO/Fendt used the CANoe .ISO11783 simulation and test tool to assist in integrating high-performance test automation in its existing hardware-in-the-loop test environment. Systematic tests now improve product quality, and in an era of 'precision farming' it lets the company confidently address the trend towards further intelligent vehicle components.

The J1939-based ISO standard 11783 describes CAN-based communication in open networks for use in mobile machines in the agricultural industry. ISO 11783 is a multi-master network based on CAN, whose protocol is harmonized with J1939. The ISOBUS concept describes a minimum degree of functionality for software and hardware that ECUs or networks must provide. The ISOBUS protocol ensures that a suitable add-on implement such as a field chopper or manure spreader can be operated with any ISOBUS-conformant tractor and that its full functionality will always be available on the tractor's operating console. The individual parts of the standard, address such topics as Network Management, Tractor ECU, Universal Terminal (previously: Virtual Terminal), Task Controller, Diagnostic Services and File Server.

Advantages of Standardized Agricultural Technology

The advantages that ISOBUS provides to users and manufacturers alike can be illustrated based on the example of the Universal Terminal (UT). After an implement has been attached, the UT reads a library with graphic control elements (Object Pool) from the implement's controller. These objects represent the various functionalities available to the user and enable convenient operation of the implement.

Goal of ISOBUS Functionality Only Attainable with Extensive Tests

Until everything functions as smoothly as in the example, numerous test runs and correction cycles must be performed by the testing and development departments. On the one hand, the ISOBUS standard leaves some room for interpretation in certain aspects, but projects of this complexity need time in the prototype phase before they attain the required maturity level. There is no way to avoid conducting intensive and extensive conformity tests [1]. That is also why the various manufacturers meet regularly at so-called "Plug-Fests" where they test individual devices for their compatibility with one another.

Test Setup of the Entire Tractor Network with an HIL System

The AGCO Corporation is one of the world's largest manufacturers and distributors of tractors and agricultural machines. In 1997, the US-based agricultural machinery corporation acquired the tractor manufacturer Xaver Fendt GmbH & Co. with headquarters in Markt-oberdorf, Germany, where it conducted its development production and sales of tractors. Along with tractors, the company's lineup includes field choppers, combine harvesters and baling presses products which are handled by the company's business sites in Bavaria, Sachsen-Anhalt and Italy.

To overcome the challenge of increasing testing effort, AGCO/Fendt conducts extensive tests on real tractor prototypes. For some time now, the company has also been working with the development and consulting company Gigatronik. AGCO/Fendt develops nearly all of its ECU software in-house, but it outsources some testing-related jobs to specialists in electronics and information technology. Its central tasks are to test Universal Terminals, Tractor Controllers and to validate ISOBUS conformity.

The key components of the test setup used at AGCO/Fendt are a breadboard fixture of the tractor network, a hardware-in-the-loop system (HIL system), a real-time server (RT target) with extensive I/O interfaces and a PC with the CANoe .ISO11783 development, testing and simulation tool from the company Vector, which is precisely tailored to ISOBUS requirements. CANoe .ISO11783 provides ISOBUS functionality for developments from the design phase to the testing phase and then maintenance. The complex ISOBUS communication structures can be analyzed, visualized and conditioned in many different ways. The room for interpretation of the standard mentioned above can be handled in a greatly simplified way, because the tool essentially embodies the standard as interpreted knowledge.

The tractor breadboard fixture incorporates all electrical and electronic tractor components such as ECUs, bus systems, wiring, lamps, switches and controls. They are mounted in a space-saving way on a frame with industrial-format racks, which is suitable for the laboratory. The fixture matches a production tractor fully; all it lacks are the diesel engine, transmission, body and add-on parts **(Figure 1)**.

The breadboard fixture is connected to the RT server's I/O routing block via a multicore cable **(Figure 2)**. The server simulates various ECUs and the missing environment of the engine and sensor signals in real time via MATLAB/Simulink; it supplies such signals as speeds, rotational speeds and temperatures. User-initiated events such as button presses or movements of a joystick must be performed manually. A look at the I/O level illustrates the magnitude and complexity of the overall HIL system. It consists of several hundred voltage and current interfaces, countless digital inputs and outputs, and outputs for driving relays. In addition, there are frequency outputs, all sorts of CAN buses, a UDP interface and multiple voltage outputs with supply voltages for the ECUs of the breadboard fixture.

Figure 1: The breadboard fixture of the tractor contains all electronic components of the real tractor model. Automated tests can be executed to real-time conditions with CANoe .ISO11783 and the RT target.

Manual Testing as Time-Consuming Emergency Solution

Another tool available for tests is a PC with CANoe .ISO11783 simulation and testing software. The system is primarily used for remaining bus simulation [2] and is capable of simulating individual network nodes up to entire sub-networks. In CANoe .ISO11783, entire implements can be simulated, and the simulated implements are used to put the tractor's ISOBUS capabilities to the test. Even sub-areas, such as a UT, can be simulated and displayed with all of their functions on a PC screen using CANoe. Testing of ISOBUS applications takes a lot of time and effort, because conformity must be verified for all parts of the ISOBUS standard used in the tractor implement. This also involves internal tractor communication. Moreover, the ISOBUS standard itself is available in various implementation levels. Therefore, it is not sufficient to simply test to the current level, rather the test must also assure backward compatibility to a number of previous implementation levels of implements. Finally, numerous individual tests (test cases) must be run for many different variants.

Despite the availability of the described HIL system, employees at Fendt and Gigatronik still needed to perform some necessary tests by conventional manual methods. Specifically, this sometimes meant – depending on the test case – that an employee sat in the tractor and executed specific tractor and implement operating functions repeatedly, for hours at a time and over several days. During such testing, the test system logged all user actions, system reactions and the bus traffic. The logging record was then used to precisely analyze the contents of messages, terminal reactions, timing values, cycle times, etc.

Test Automation is at Top of the Wish List

However, the test method practiced in this way was still unsatisfactory for Gigatronik and Fendt. Highly qualified employees were overworked by simple yet time-intensive user actions. And despite the large amount of time required, the tests still exhibited a somewhat rudimentary character considering the many parameters. Therefore, managers began to seek out and evaluate options for systematic and in-depth test automation to replace the undesirable manual interaction. A solution with robots that could simulate the human movements with mechanically actuated control arms would not only be extremely difficult to implement; it would also be inflexible. So this solution was rejected early on.

An electronic approach seemed more appropriate and appealing, especially since a high-performance test tool already existed in the form of CANoe. The 'Test Feature Set' integrated in the software produced a comprehensive tool chain covering all phases: from defining test requirements to executing tests and finally generating evaluation reports. CANoe can act as a test master, or it can be inserted in test environments. Interfaces such as COM or .NET are available for drive control and communication with other tools. Nodes referred to as test nodes can be created for the tests; they participate very normally in real bus events, similar to simulated network nodes. Test engineers define their behavior and functionality in test modules, using either the C-like script language CAPL or the XML standard.

Figure 2: The original HIL system is composed of the real-time target, tractor breadboard fixture and UDP interface for communication with a remaining bus system. The armrest with integrated user controls is implemented as a real component, for example, and must therefore be operated manually.

Figure 3: Integration of the CANoe interface in the existing overall system: The test module controls the existing test environment. The armrest, with its integrated controls, is implemented as a virtual component in the test system and can therefore be operated either manually or automated by the test program.

CANoe Bridges Gap to Real-Time Target

Before test automation could be successfully implemented, it was necessary to answer the question of how the CANoe tests and simulations would be linked to the real-time target in the HIL system. That is because a fast communication channel is indispensable between the two systems. The RT target has a UDP interface with a data link protocol created by Fendt for sending information to the RT target or obtaining information from it. Therefore, Gigatronik developed a compatible counterpart to the CANoe side with its resources – in the framework of a student's senior thesis. The test system can now control the RT server via UDP and optionally transfer to it the same signal vector that was previously made by manual interaction with a real component, e.g. an armrest with integrated controls **(Figure 3)**. In principle, any real, manually operated component can be automated in this way. In the example, it was an intelligent armrest; in another case it might be a steering wheel.

Now that these preparatory tasks have been completed, Gigatronik and AGCO/Fendt have the capability of comprehensively automating ISOBUS and ECU tests and quickly modifying them. In particular, the capabilities of CANoe .ISO11783 now contribute towards much more efficient test execution. For example, the UT provided by CANoe makes it possible to control an ISOBUS implement just as well as with the real UT on the tractor. While an employee would have to execute user control functions for hours on end at the real tractor armrest previously, now a simulated armrest with the same integrated controls automatically executes the same actions **(Figure 4)**. The behavior of the simulated component in the individual test cases is defined precisely in CAPL test modules. The test user interface also makes it possible to start the (virtual) engine, and sequences can be programmed for forward driving and backward compatibility. Especially important is that, in contrast to manual operation, CANoe.ISO11783 can execute tests with absolute reproducibility. In addition, errors may be interspersed in a controlled way. During the test runs, the test system logs all relevant messages and signals, and afterwards it automatically generates a test report.

Decidedly Greater Test Quality by Automation

The automated test runs open up entirely new aspects of quality in ECU testing. It is easy to run tests overnight or over several days, without employees having to be constantly present. Naturally, this makes it possible to go into much greater depth, test much more, and systematically run through many more test cases. They include tests that would not have even been possible without the test tool, such as multiple repeated checks of bus loads, when the tractor drives off, when it stops, when multiple implements simultaneously lift an object, etc. The developer analyzes how other messages react in these situations and how timing values and cycle times behave. Statistical evaluations can also be performed, e.g. by starting the test run thousands of times and logging while varying such parameters as the latency times of certain messages or examining which effects might result when starting up the network.

Figure 4: Instead of using a real armrest with integrated user controls, the signals coming from this component are fully implemented by a GUI that can be switched between manual operation and control by the test program.

Finally, one significant part of EDCU tests is to subject the system under test to errors. Here too, the RT target generates all conceivable error situations upon instruction by CANoe.
Bernhard Stöckl, manager for the responsible testing department at AGCO/Fendt, is pleased: "Before, we had no test automation in the ISOBUS area. Previously, we conducted all of our work directly on prototypes which are not always available. We actually sat by the machine and operated the switches. Now, test automation offers tremendous relief in terms of workload. We require much less time on the real prototypes and no longer need to build so many extremely expensive test vehicles. At the same time, we saved a lot of time, which the developers are now investing in further automation efforts."

Outlook

'Test automation' is the magic word for mastering the constant growth in testing effort, simultaneously improving quality and saving on time and costs. AGCO/Fendt and Gigatronik have shown the capabilities offered by test automation with the CANoe simulation and test system and how to achieve efficient test practice quickly and in an uncomplicated way. The flexibility of the test platform makes it possible to overcome even greater challenges related to communication and interfaces in the process of integrating a large HIL system.
Only by taking new approaches to integration tests can product quality be assured into the future. This issue is assuming increasing urgency in view of the further increasing diversity of ISOBUS-capable agricultural machines. Additional requirements will soon appear in the context of precision farming. More and more GPS applications that support work with field machinery will be implemented in both tractors and implements. In Tractor Implement Management (TIM), the implement controls the tractor, e.g. by specifying tractor speed or rpm. These extensions can be accomplished much more efficiently with test automation.

Translation of a German publication in Elektronik automotive, 12/2013

Image rights:
Lead Figure: Fendt
Figure 1 – 4: Gigatronik

Literature:
[1] Ostermüller, A., Fellmeth, P.: Forging New Pathways in Testing ISOBUS Task Controllers – Simulations replace inflexible and time-intensive test methods. Elektronik automotive. Issue 11.2011. pp. 32 – 35.
[2] Albrecht, S., Decker, P.: Quick paths to a comprehensive remaining bus simulation – Create virtual networks without programming expertise. Automobil Elektronik. Issue 03.2012. pp. 58 – 60.

Links:
Website AGCO/Fendt: www.agcocorp.com
Website Gigatronik: www.gigatronik.com
Website Vector: www.vector.com
Product Information CANoe .ISO11783: www.vector.com/canoe.iso11783

Bernhard Stöckl
graduated in Mechanical Engineering from the Technical University of Munich. He is currently employed as Business Unit Manager for Testing of Drive Technology and Electrical/Electronics at AGCO/Fendt. He previously started up the Hardware-in-the-Loop unit at the company as a test engineer.

Hans-Werner Schaal
graduated in Information Technology at the University of Stuttgart, and in Electrical & Computer Engineering at Oregon State University in Oregon, USA. He is Business Development Manager at Vector for open CAN protocols such as ISO 11783 and J1939 and in the IP and Car2x area. Previously, he was employed as a development engineer, project leader and product manager in the test tools area.

Thomas Bückle
graduated in Information Technology at the Polytechnical College of Ulm. He is Business Manager for the area of agricultural technology and construction equipment and Department Head for Embedded Systems Development at GIGATRONIK Technologies. Previously, Thomas Bückle was development engineer and project leader in electronic development at Daimler and EvoBus.

Benjamin Fernandez
has a degree in Mechanical & Electrical Engineering from ITESM, Mexico, a degree in Mechatronics from the Technical University of Hamburg-Harburg and an MBA from NIT Hamburg. He is a development engineer at GIGATRONIK Technologies.

Development of a Cooperative Tractor-Implement Combination

While driver assistance systems such as adaptive cruise control and lane-keeping assistants are increasingly handling longitudinal and lateral control of vehicles in road traffic [1], in the agricultural industry the trend towards driverless systems is making progress. This trend is also being supported at the "Chair for mobile machinery" of the Karlsruhe Institute for Technology (KIT) in conjunction with electronic drawbars for agricultural machinery developed by the companies AGCO and "geo concept" [2].

Currently, there is great interest in automated driving in the automotive industry. This topic is also gaining in importance in the development of agricultural machines. In a research project at the Karlsruhe Institute for Technology, a new system has now been developed, in which a manned agricultural machine drives ahead as a lead vehicle, and another one follows it autonomously and driverless. The goal of the project is to increase area performance and optimize machine utilization in agricultural business without incurring additional personnel costs.

Electronic Drawbars for Tractors

The electronic drawbar makes it possible to operate two tractors simultaneously with just one driver. This involves using two tractors of the same type, equipped with identical implements and highly precise RTK-GNSS receivers (Real-Time Kinematics) and to interlink them via data radio. While one tractor, which is configured as the master vehicle, drives ahead with a human driver, an unmanned vehicle configured as a slave follows with a prescribed longitudinal and lateral offset. The work settings of the master vehicle to implement the local work process are copied by the slave while still stationary. Good operability and a high level of safety are achieved by the use of environmental sensors to detect obstacles **(Figure 1)** and web-based geo-information.

System Functionality

Five driving modes are available to the driver in operating the electronic drawbar.

The "Parallel Driving" mode serves as the starting mode and working mode. Here the slave follows the driving track of the master and performs the same work task. If the slave follows exactly in the driving track of the master, then the "Tracking" mode is useful. The "Ignore" mode makes it possible to quickly decouple the slave to perform independent driving maneuvers with the master. When leaving this mode, the slave joins up with the master on its current path. The "Turn-Over driving" mode is available for a turning maneuver at the headland. If an obstacle is reported along the planned path, the slave seeks out a collision-free detour path in the "Evasion" mode.

Safety Concept and State Machine

Safety is a key aspect in the development phase of an unmanned vehicle. Based on ISO 25119, typical tools are used to identify risk sources and weaknesses related to the functional safety of a machine, and these risk sources and weaknesses are addressed by suitable measures. A risk analysis yields an Agricultural Performance Level (AgPL) for each scenario studied in the product life cycle, from which requirements can be derived for hardware and software. The electronic drawbar is based on a three-stage safety concept **(Figure 2)**.

> In the "Operational" state, there are no critical error messages or warning messages; the planned path is free of obstacles.
> The "Safety Stop" (SSt) state is understood as an active stop for driving and working drives.
> The highest level of the safety concept is the "Emergency Stop" (ESt) state. Here, the engine is shut off immediately, and the electronic parking brake is engaged.

The safety concept is implemented in the form of a state machine which is active as the central software module in both vehicles which sets the synchronized overall system state. **Figure 3** shows the state machine of the electronic drawbar. At the system start, the "Default" state is active initially.

The role of the tractor can be set as master or slave. After setting roles, an assignment must be made to both vehicles in the "Assignment" state, in which the radio link between the two vehicles is produced. After assignments have been made to the two vehicles, the system transitions to the "Safety Stop". The slave is now in the "Active Stop" state. Now the "Emergency Stop" can be triggered at any time. To initiate driving operation, after enabling by the driver the vehicles are coupled to one another in the "Docking" state, provided that their relative positions and orientations allow this. The system transitions to the "Operation" state, and the slave is able to drive autonomously. It starts in the standard driving mode "Parallel Driving", then any other driving mode can be activated. Risk of collision is detected by environmental sensors consisting of 2D laser scanners and 3D cameras, and detection leads to an immediate "Safety Stop".

Prototype Development at Distributed Sites

Implementation of a complex modular assistance system with external project partners requires a detailed specification of interfaces. Because the development platforms at individual partners are not harmonized, a distributed architecture was selected, in which different software and hardware modules communicate with one another over a proprietary edaugCAN bus and over TCP/IP Ethernet.

Hardware Architecture

While the project partner was responsible for implementation of navigation functions (NAV), access to proprietary and customized Geo data servers (GIS) and access to vehicle control (EXT) via the main controller and user interface for the standard terminal (HMI), KIT implemented the state machine (ZST), and environment detection (UFS) on a rapid prototyping platform.

Figure 1: Prototype setup for testing environmental sensors of the slave tractor

Figure 2: Safety concept of the electronic drawbar

Data Communication

The physical medium used to exchange system and operating data within the vehicles over the proprietary CAN protocol was the tractor's ISOBUS. By choosing 11-bit identifiers, both buses can be operated in parallel without any problems. Each hardware module has its own CAN-ID, under which all relevant signals are transmitted via multiplexing. The bandwidth-intensive environment data is transmitted over TCP/IP-Ethernet. A low number of nodes and network load permit real-time sending and receiving of data. Radio communication between the vehicles is in full-duplex mode. The modems used are linked via RS232 interfaces.

Master and Slave continually exchange data on position and movement states, the system state, settings of working drives, the vehicle state and environmental information **(Figure 4)**. Pragmatic organization of the interfaces enables uncomplicated implementation and validation of subfunctions. Timeouts and checksums were used as safety mechanisms for diagnosis of typical data transmission errors. This guarantees the data integrity and activity of the nodes. If a timeout is exceeded on the CAN bus or in the radio path, this immediately leads to an "Emergency Stop", because it is no longer possible to assure control over the vehicle.

Remaining Bus Simulation to Support Software Development

In the development of ECUs, the remaining bus simulation (RBS) is a powerful tool that is used to support debugging and to validate the software under the most realistic constraints possible. Not only in the automotive industry, but in mobile machinery as well, ECUs, sensors and actuators communicate in different networks simultaneously. Generally, comprehensive software tests are not possible until the ECU's interfaces can be supplied with realistic values. To avoid having to perform time-consuming and complicated software tests in the real system in the development phase, an RBS can be used to virtually simulate the remaining network environment of an ECU. Software tests performed in this way are cost-effective, fast and noncritical in terms of safety.

CANoe

Vector's CANoe software flexibly supports the development of network communication. The structure of a CAN network is fully simulated in CANoe with the help of network nodes, in which the communication behavior of the nodes is stored. In addition, user interfaces, signal generators and other communication interfaces can be created as network nodes. The event-driven or interactive behavior can be simulated in CAPL, a C-like programming language [3] or, for example, with the help of compiled Simulink models. In a database, all messages that occur, including signal structure, are stored and assigned to individual network nodes. CANoe offers multibus capability, i.e. it supports both proprietary and standardized protocols such as CAN, J1939 or ISO11783 as well as FlexRay and Ethernet. For modeling and time-synchronous evaluation, it offers a set of stimulation and tracing functions with many different filter options, graphic signal display and an overview of important communication parameters such as bus load and error messages. Any desired ECUs can be operated virtually or as real ECUs by activating or deactivating the network nodes in CANoe. The bus interface is via an interface from the company Vector. CANoe can also run test scenarios fully automatically and report on the results. These test scenarios may be defined in table form or in XML, CAPL [3], C# or graphically.

Modeling the Electronic Drawbar in CANoe

CANoe is well-suited to this project configuration, especially in early validation of ECUs, because coordinating testing dates and options with the test vehicles is often critical. The approach using RBS enables parallel development paths, which can later be easily merged.

A comprehensive model of the electronic drawbar was developed at KIT that could be used to simulate the individual ECUs as well as the entire communication with the partner vehicle. The ZST, NAV and HMI software modules were modeled as network nodes here. The depth of detail of the individual modules varies within the model. While all messages controlling the system state were simulated in their full functionality, data integrity could be validated sufficiently by assigning constant test parameters in the remaining messages. The use of CANoe in conjunction with the model-based programmable of the remaining messages dSPACE platform proved to be especially helpful. After the exchange of platform-specific I/O blocks, the messages for the developed Simulink models can be immediately compiled for CANoe and integrated in the remaining bus model. Since the serial interfaces of the PC could be accessed in CANoe, the radio path could also be integrated in unmodified form and be tested for data integrity.

Summary

The electronic drawbar has not only proven to be complex with regard to the innovative aspect of object detection and integration of geo-information in a harmonious safety and operating concept. In a continually changing prototype system, the specification and maintenance of data interfaces is a supportive task that must be taken seriously by all parties. When multiple companies are cooperating on a project, a clearly specified modular system and software architecture quickly proves to be an important foundation for quick progress in the project. However, rapid prototyping systems in conjunction with RBS software represent a project-assisting toolbox for efficiently validating subfunctions and entire ECUs so that they are successful in subsequent field testing.

Translation of a German publication in HANSER automotive, September 2014.

Image rights:
Lead Figure: Vector Informatik GmbH
Figure 1 – 4: KIT

Figure 3: Model for defining the operating state

Figure 4: Communications architecture in prototype

Literature:
[1] Winner, H.; Hakuli, St.; Wolf, G. (2012). Handbuch Fahrerassistenzsysteme ("Handbook of Driver Assistance Systems"), 2nd edition, ISBN 978-3-8348-1457-9. Vieweg + Teubner Verlag, Springer Fachmedien Wiesbaden GmbH.
[2] Zhang, X.; Geimer, M.; Noack, P.O.; Grandl, L. (2010). Development of an intelligent master-slave system between agricultural vehicles. pp. 250-255. IEEE. Intelligent Vehicles Symposium. 6/21-6/24/2010. San Diego, CA, USA
[3] Marc Lobmeyer/Roman Marktl: Programming ECU Tests More Efficiently – Tips and Tricks for the Use of CAPL. In: CAN Newsletter (2014), H. 2, pp. 42-43

Links:
Homepage KIT: www.kit.edu/english/
Homepage Vector: www.vector.com
Product Information CANoe: www.vector.com/canoe

Bernhard Jahnke (Graduate Engineer)
is a scientific assistant at the Chair for Mobile Machinery of the Karlsruhe Institute for Technology (KIT).

Hans-Werner Schaal (Graduate Engineer)
is Business Development Manager in the areas of Automotive Ethernet, Car2x and open CAN protocols such as J1939 and ISO11783 at Vector Informatik GmbH.

Prototyping and Testing CANopen Systems

Reaching Goals Rapidly with Minimal Effort

CANopen established itself as a standard for cost-effective and flexible networking of components in numerous application fields ranging from industrial automation to commercial vehicles. Standardized device profiles simplify communication between bus nodes and facilitate smooth interplay between the ECUs, sensors and actuators from different manufacturers. A specialized prototyping and test tool not only performs valuable services in developing complex CANopen projects, but also in providing a fundamental introduction to the subject area.

The bandwidth of tasks in the development of CANopen systems ranges from developing individual ECUs to configuring and starting up total systems. For system producers, the initial use of a system such as CANopen is associated with an incalculable startup effort. In the framework of the V-model, a large share of development tasks involves verification and validation. The risk of detecting errors too late is minimized by comprehensive tests at the earliest possible time **(Figure 1)**.

System Verification with Test Setups

Systematic tests accompanying the development of a total system are indispensible in all development steps. Often it is possible to test, but not until a very late time, when actual system components are available. Then, the system completion date is at risk if problems occur.

In practice, tests are not performed on the real system at the customer's facilities, rather test setups are used for this purpose. Besides special measurement or diagnostic setups, in testing they also include actuators for simulating system environments as realistically as possible. Depending on the project size, such a test setup could incur much cost and effort. Bottlenecks are built into the process during the test phase, since generally only one test setup is available.

A way out of this dilemma might be offered by a software tool that could be used to create a prototype of the future total system in a simple way. Optimally, this tool would also offer extensive test functions.

Creating a Prototype Environment with Tool Support

Above all, prototypes of total systems networked by CAN should support testing and validation activities. In addition, it is important to represent the individual components by real or simulated ECUs. This makes it relatively easy to test the functional capabilities of real ECUs over the course of system development.

Creating a prototype for a complex total system is a complicated endeavor in which it pays to use suitable tools. CANoe .CANopen from Vector offers valuable assistance in creating communication for the prototype system. In just a few configuration steps, it lets the user create a prototype whose communication properties match those of the later real system.

Figure 1: Representation of the V-model

Figure 2: Sample configuration of a simple, simulated CANopen network with central control and I/O node

The behavior of the CANopen ECUs is defined in description files (EDS – Electronic Data Sheet) that are selected or created beforehand. This next step is to define the interrelationships between calibration data that will later be exchanged over the bus, e.g. the "PressureValve" input of the device with address 5 (pressure transducer) is linked to the "Gas pressure" variable of the device with address 10 (control module). This is how all PDO (Process Data Object) interrelationships are defined in the prototype system.

The tool automatically computes the resulting mapping, which can be modified later if necessary. From this configuration information (DCF – Device Configuration File) the user now generates the prototyping environment, i.e. CANoe generates a counterpart for each real ECU with identical communication properties. When CANoe is started, the communications portion of the prototyping environment is already available **(Figure 2)**.

Figure 3: Representation of the CANoe .CANopen environment

Defining Application Behavior

The application behavior of individual ECU nodes cannot be derived directly from the EDS files, since they only map the structure of an object directory. Therefore, formulation of application behavior always involves additional programming effort. The CAPL programming language integrated in CANoe makes it possible to program ECU behavior quickly. As an alternative it is possible to code ECU behavior in a DLL that is linked in the prototyping environment under CANoe. It is also easy to integrate Matlab/Simulink models **(Figure 3)**.

Test Functionality

After the prototype system has been completed, the necessary tests for the total project follow. CANoe supports the user here, both in creating tests and in logging and evaluation. The test functionality required for a CANopen system comprises several stages **(Figure 4)**:

- Protocol level:
  Tests on this level ensure that CANopen-specific communication protocols of the individual devices are implemented within the total system according to specification.
- Communication level:
  What is important here is not the correctness of the protocol, but the correct logical order of (independent) protocol sequences, e.g. in configuring PDOs. Description of the PDO-relevant entries in the object directory must conform to a specific sequence.
- Application level:
  Application tests check the relationships between process variables. The following preconditions must be met to even determine such relationships: First, the process variables must be exchanged via PDOs, and the system must be fully configured. This means, for example, that the state of a valve can be checked as a function of a temperature or a pressure. Clearly, the user must have the relevant know-how to formulate the test.

Creating and Executing Test Procedures with CANoe

Test sequences are formulated under CANoe with the help of the integrated CAPL programming language. Each CAPL test module represents a separate test consisting of individual test cases, which in turn contain a number of test steps. During the test run, CANoe processes the individual test cases sequentially. Suitable test flow control makes it possible to skip or repeat individual tests. Simultaneously, it is possible to monitor conformance to other conditions and restrictions – quasi in background. This might be done, for example, to determine – while waiting for a specific message of interest – whether a specific other message is sent periodically on the bus.

Especially in automated tests, it is important to record the results of individual test steps in detail. Other CAPL functions handle writing of test results to an XML file for later post-processing. Test sequences for CANoe can also be specified under XML. This is advantageous when a large number of test flows need to be generated with a single tool. So CANoe utilizes a number of test patterns specified in XML and processes them accordingly.

Summary

When prototypes are created in CANopen networked systems, this process always involves considerable effort. Yet the process is often essential to prevent findings on functionality from being delayed until a later project phase. Special tools such as CANoe .CANopen offer the user efficient support in creating prototypes. The requirements for technical aspects of communication, in particular, are very easy to cover in this way. In every phase of a project, this gives the system developer the test functionality needed to check and verify components completed up to that point in time.

Mirko Tischer,
Product Manager for CANopen.

Figure 4: Test levels in CANopen

Automatic testing of CANopen devices

Kai Schmidt (Vector Informatik)

The growing complexity of today's system architectures is associated with an increase in the effort that must be invested in test specification, test creation and test execution during the development of such systems and system components. Test specifications should be available in early phases of the development process, e.g. after the system architecture has been created or during component design. This makes it possible to detect errors early and correct them cost-effectively.

The development process for a CANopen system can be described based on the V-model. In the first phase, system requirements are defined, which for the most part contain the definitions of individual "use cases". This information represents the input for the next step, where initial assessments can already be made of the system architecture. Functions are assigned to the individual ECUs, and device descriptions can be created for all devices in the form of EDS files (the format of the EDS files was standardized by CiA and is being further developed by this organization in cooperation with industry). In addition, communication relationships between the ECUs can be configured, as network management and error detection mechanisms. EDS files describe significant parts of the functional scope of a CANopen device. These device descriptions form the foundation for executing the simulation and creating test specifications. Communication-specific tests can be derived directly from the device descriptions. An example of this would be a test

Fig. 1: Development process of a CANopen system

objects in the object dictionary by SDO accesses and records the results. Besides communication-specific tests, application-related tests can also be specified. An example of such a test would be to stimulate the transmission of the digital input of an I/O device. Afterwards, a check is made to verify that the signal value exists at the output. Both tests could be used early in the simulated overall system. As soon as the stability of the overall system has been achieved, development of the individual components can be subcontracted. The EDS files can – with the exception of application-related behavior – be considered as a requirements specification for the supplier. Parallel development of the ECUs at the suppliers is accompanied by the simulated overall system. Application-related tests can also be utilized at the supplier to test the behavior of the device to be developed within the over-

cantly reduce the number of cost-intensive changes desired by the OEMs – which generally occur within the integration phase. Communication-specific tests can be created at the supplier in a similar way as at the OEM.

After completion and acceptance of the components, they are successively integrated into the simulated overall system. The previously created communication and application-related tests can now be applied to the system, consisting of the physical components and the rest-of-bus simulation. As soon as all of the components have been delivered the concluding test of the real overall system follows.

EDS files as the basis for tests

The development process should include the creation of an EDS file appropriate to the device. Unfortunately, practice shows that de-

glect this work step. Faulty or incomplete EDS files are the result; in the worst case there is no EDS file at all for a device. The development process described above shows that it is not just device producers who need to be concerned with creation of EDS files, but system designers too.

The task of the system designer here is to distribute functionalities to the individual components. These could be standardized functionalities such as mechanisms for process data communication, but they might also be manufacturer-specific functionalities. Both of these are mapped via objects in the object dictionary. CiA specifications describe standardized functions. Standardized parameters (process data, configuration and diagnostic data) as well as manufacturer-specific data can be stored in a database format that is also standardized. The necessary objects can be selected from the object pool that is created in this way, and be assembled into an object dictionary.

The device descriptions contain all of the information necessary for simulation of the CANopen device. The overall system, consisting of the individual device descriptions, is parameterized utilizing a suitable configuration tool, and an initial system description is obtained in the form of device configuration files (DCF), whose format has also been standardized by CiA. Based on this configuration, simulation models can be generated and executed in a suitable runtime environment. At an early point in the project, this already enables conclusions about the time behavior of the overall system. If excessive bus loads occur, for example, actions can immediately be initiated to correct the problem, since suppliers have not been involved in the development process yet. Accordingly, the simulated overall system offers a high degree of flexibility. It can be refined iteratively until it satisfies the defined requirements. Changes to the simulated system can be implemented cost-effectively and be checked immediately.

Derivation of test sequences

Besides the simulation, it is also possible to derive initial tests on the protocol and communication levels from the device descriptions. The protocol test includes checking of the SDO protocol, for example. The communication tests do not check for correctness of the protocol, but instead for correct flow of message sequences. For example, it is possible to test whether the configuration sequence for process data objects conforms to the sequence specified in CiA 301.

The following test templates with general application can be defined for a CANopen device:

- SDO download test
- SDO upload test
- Heartbeat producer test
- Heartbeat consumer test
- Transmit PDO test
- EMCY test

Test functions are created for each object contained in the object dictionary here. The test functions are parameterized based on the data contained in the configuration files for the devices. Among other things, test sequences can be generated to check the:

- PDO configuration
- Default values
- Object dictionary
- NMT state machine, and
- SDO protocol

The generated tests may be executed right away in a suitable runtime environment. In the framework of integration work, it is precisely such tests that are used to check the delivered components. In turn, suppliers can generate similar test sequences to assist in development. They can immediately be applied to the prototypes. Essentially, this is a way to generate test sequences of the conformance test (CiA 310) device-specifically. However, the goal of the system should not be to replace the CiA conformance test altogether. The system should accompany the development and give developers a way to test devices in advance of the actual tests. The final certification is only performed by CiA.

Generation template for each version

Generation templates must be created for each test, but they are applied to each device to be tested. A generation template that describes the creation of a test for checking the object dictionary would appear as follows:

```
for all objects
{
  get access type
  if(access == read
only){
    add test function
SDO Upload
    to test sequence
  } // if
  else if (access ==
read write){
    add test function
SDO Upload
    to test sequence

    add test function
SDO Download
    to test sequence
  } // else if
  .
  .
}
```

The generated test sequence created based on this test template contains a number of parameterized (by entry of object index, etc.) write and read routines. They are processed sequentially in test execution.

Iterative development process

Since iterative processes are applied throughout a device's development, the ▶

Fig. 3: Test sequences of a CANopen system

process for generating test sequences must be repeatable as often as needed. Changes to the device design can affect the device descriptions. The test that was originally generated would then likely fail. Nonetheless, it is still necessary to be able to manually extend test sequences after generation, e.g. to incorporate application-specific supplements. These extensions must be read back when the sequence is regenerated.

Attribute	Value
Index	8083$_h$
Name	GPS date
Object code	Variable
Data type	Unsigned32
Category	See CiA 447-2

31	25	24	20	19	15	14	0
r	GPS date day		GPS date month		GPS date year		

Fields	Value	Definition	Unit
GPS date year	0000	Minimum value	Years
	FFFD$_h$	Maximum value	
	FFFE$_h$	Failure	
	FFFF$_h$	Signal not available	
GPS date month	00$_h$	Reserved	Months
	01$_h$	Minimum value (January)	
	0C$_h$	Maximum value (December)	
	0D$_h$	Reserved	
	3E$_h$	Failure	
	3F$_h$	Signal not available	
GPS date day	00$_h$	Reserved	Days
	01$_h$	Minimum value (1st)	
	1F$_h$	Maximum value (31st)	
	20$_h$ to 3D$_h$	Reserved	
	3E$_h$	Failure	
	3F$_h$	Signal not available	
r	11 1111$_b$	Reserved	

Fig. 4: "GPS date" object description

Application-related tests

The application-related behavior of the devices can not be represented in the device descriptions. Furthermore, the tester does not want to have to deal with the CANopen-specific conceptual world and its definitions on the application level. On this level, it is entirely unimportant to know which signal is mapped to which object at which position. More important is information about which signals exist and which devices receive or send them. Among other things, precisely these aspects must be tested. This subject matter can be explained by the example of the CiA 447 application profile (application profile for special-purpose car add-on devices):

The standard defines an object "GPS date". Mapped to this object are the signals "GPS date year", "GPS date month" and "GPS date day".

The CiA 447 profile, besides defining signal allocations in the objects, also defines the transmission type. The standard specifies that the object value "GPS data" is transmitted by SDO protocol. The following information is needed to transmit a signal as part of a test:

- Index + Sub-Index of the object
- Signal length
- Start position of signal within the object

The format of today's EDS files is unable to describe the signal allocations of object values. Accordingly, information such as the signal length and start position of ▶

Fig. 5: Generated test environment

the signal is also unsupported. Even if these requirements could be implemented, it is not possible to automate generation of application-related test sequences, since the behavior of the system is not described.

Generated test environment

Nonetheless, the developer can be supported by generation of an "interaction layer" in test creation. If this extension can be integrated in the simulated overall system, then it is easy to create application-related test cases.

The test system consists of the simulated nodes that are extended to include an "interaction layer". One or more physical devices are tested. The simulated devices are stimulated via generated interface functions. Signal values are mapped to object values and the CAN messages are sent. In the example depicted, the signal value "GPS date month" would be mapped to the relevant position in the object value (startbit 16, length 4 bit).

Parameterization of the test functions assumes that the positions and length of the signals are known. Moreover, the transmission type must be considered. This information is described exclusively in the standard and must be considered in test creation. Use of an "interaction layer" enables signal-oriented test creation. It will be possible to define the function "Send_GPS_month" and generate its implementation based on the CiA 447 specification, if it exists in XML format in the future. Today's format of the specification requires converting the specification to a readable format (XML or Excel).

This conversion task can be assumed by a generator. The generated functions contain a mapping of the signal to the object value and a routine for sending the CAN messages. During test creation, the test engineer need not be concerned about signal positions, indices or transmission types. All the test engineer is interested in are the signal name, sender, receiver and signal value.

kai.schmidt@vector-informatik.de

Connectivity

Practical experiences with smart systems and services

Connectivity today is the mega-trend across industries. Systems and services become smart by being aware of their environment. Connectivity however is challenging as it demands a fully new set of competences, combining embedded systems development with IT and cyber-security. To support the evolution towards secure and robust connected services, Vector has summarized here best practices for developing and delivering connectivity. The article provides hands-on experiences from BMW, Essence, Ford, Vodafone and Vector. Industry case studies emphasize how to master the transformation to connected systems and services.

„Connectivity is the base technology to evolve fully new business models and technologies across industries." This is the summary of Dr. Christof Ebert, CEO of Vector Consulting Services, from his experiences with companies worldwide. Connectivity has been the leading theme in the Vector customer survey „Industry Trends 2017". Connectivity is the prerequisite for digital transformation and enables the real-time observation and control of systems and processes. It is the link between the real world and the digital world. This includes data acquisition, analysis and modeling as well as the necessary information technology and networking. Autonomous driving, predictive maintenance, IoT and Industry 4.0 are current application areas which we will dive into.

Vector literally connected different companies and industries to share their experiences and distill guidance for engineers and decision-makers. BMW, Bosch, Essence, Ford, Vodafone and Vector amongst others shared their strategies on connectivity. Gerhard Mack, COO of Vodafone, set the stage with new business models with innovative services for connectivity and mobility. Dr. Haim Amir, the CEO of Essence group in Israel moved into connectivity experiences in the Internet of things (IoT) and Internet of Services (IoS). Dr. Matthias Traub, Chief Architect of BMW elaborated on innovative electronics and IT system architecture as enabler for networked mobility. Joerg Schaeufele of Vector demonstrated with practical case studies and innovative tools how to design and deploy Service-Oriented Architectures (SOA) for flexible connectivity, such as adaptive AUTOSAR. Frank Kirschke-Biller, responsible for central software and systems engineering at Ford emphasized the growing relevance of systems engineering and security requirements for connectivity. Dr. Ulrich Bodenhausen of Vector provided experiences how on the convergence of IT and electronics for connectivity along the product and service life-cycle.

The new Vector campus with its auditorium proved to be the right place to connect car manufacturers, suppliers, IT companies and industry experts from various industries to share their experiences on connectivity. Excerpts of the event and many practical tips are provided in this article.

Connectivity and digital transformation

"Connectivity drives entirely new business models." Gerhard Mack, COO of Vodafone Germany, emphasizes the relevance of innovating the technology landscape as well as the business processes and distribution models. Vodafone today is the major player in connectivity across industries, and brings vast experiences in the automotive industry. Innovations are primarily fueled by IT. Software-driven solutions have already opened the market to new players with strong IT background teaming up or challenging established car manufacturers. This can be best illustrated by looking to the fast growth of connectivity services **(fig. 1).** Enhanced driver interaction, fluid upgrade processes, predictive maintenance, fleet management, etc. demand IT and software solutions based upon new computing paradigms and infrastructure. Examples include scalable automotive architectures that facilitate seamless connectivity, robust infrastructures for v2x (vehicle to anything) systems with safety-critical demands, or data analytics to predict necessary maintenance and improve overall customer experience.

"IoT and IoS allow value generation by flexibly combining innovative technologies" underlines Dr. Haim Amir, CEO of Essence group in Israel. With industry, home, medical and automotive applications being major drivers, IT will converge with embedded systems such as IoT (Internet of Things) and Industry 4.0. **Fig. 2** illustrates how intelligent component set-ups facilitate IoT services. At the same time embedded industries will evolve towards IT with cloud solutions and dynamic over-the-air upgrades. Different from other fields, critical industries such as automotive comprise practically all quality requirements such as safety, cyber security, usability, performance and adaptability. The underlying software components cover anything from embedded real-time firmware up to complex secured cloud solutions. Failure to meet any of those quality requirements results in expensive callback actions and legal lawsuits. These challenges will soon reach across industries.

„Incrementally implement connectivity services and the underlying digital transformation together with the necessary governance schemes." This warning of Dr. Christof Ebert, Managing Director of Vector Consulting Services, is the wake-up call. Connectivity is much more than a mere technology change. It demands a transformation process in several dimensions. **Fig. 3** shows the necessary paradigm shift in developing connected services. Business models are

Figure 1: Mobility Services Grow Entirely New Business Models.
Image rights: Vodafone Germany GmbH

Figure 2: Connectivity in the Internet of Systems and Internet of Services Generates Value By Combining Different Technologies.
Image rights: Essence Group

Figure 3: Connectivity Demands a Shift of Practices.
Image rights: Vector Consulting Services GmbH

changing thus pushing for new service-oriented delivery models on top of flexible modular platforms. IT and embedded systems must move together thus asking for a new competence mix which impacts all engineers. Not the least there are fast changing eco-systems which must be mastered on different levels, for instance to cope with new supply chain and continuous deployment or DevOps as it is called in the IT industry. Finally it is all about governance, robustness and cyber security – deeply engrained in new cultures to be developed. About 90 percent of all security related incidents are due to human errors according to the experiences of Vector when surveying customers.

Developing connected services is challenging for several reasons, because it connects embedded software with big IT systems, it is developed in a global context in distributed teams, and it has one of the shortest cycle times of all industries. A modern car has 50-120 embedded microcontrollers on board and is connected over various external interfaces to a variety of cloud and infotainment technologies. Onboard software is in the hundred MLOC range, and still exponentially growing. Automotive software product lines and variants are some of the largest and most complex in all industries. It is said that the automobile is rapidly becoming a "computer on wheels." Today an increasing amount of drivers select their car not only based on design or engine, but are increasingly also looking to ecologic footprint and software apps. Dr. Matthias Traub, Chief Architect of BMW remarks that "new architectures are necessary to master this fast-growing complexity inside the car and specifically outside to facilitate future connectivity services with IT backbone, cloud services and the necessary flexible over-the-air (OTA) deployment schemes." **Fig. 4** shows the BMW set-up of new electronics architectures as they have started to be deployed. Autonomous systems demand highly interactive services with multi-sensor fusion, far away from currently deployed functionally isolated control units.

Connectivity introduces new challenges, with respect to information security, robustness and usability. Security and robustness have tremendous impact on business models and potential liability. "The more we share and network, the more we are exposed to attacks of all kinds" explains Frank Kirschke-Biller, responsible for central software and systems engineering at Ford. **Fig. 5** shows the role of systems engineering the life-cycle. He specifically illustrates the need for comprehensive systems engineering based on growing cyber security threats. The exploding need for secure software and protection schemes for our business processes, end-to-end, indicate this impact. Imagine autonomous driving with multi-sensor fusion connected to GPS and vehicle-to-vehicle communication to predict critical situations and foresee appropriate measures in situations where even the driver might not even be aware of what will happen. Complexity and scale demand focus on usability. Today there are already situations where drivers are overwhelmed with the variety of assistance functions and at times rely too much on the embedded IT rather than staying in control. Insufficient usability today is a major source of critical failures caused by humans in healthcare, transportation and production plants.

Figure 4: Powerful Integration Platforms Enable a Seamless Hierarchical Architecture
Image rights: BMW Car Group

Figure 5: Systems Engineering Facilitates a Combined Feature, Function and Serviced Oriented Development
Image rights: Ford AG

Since quality, deadlines and cost are pivotal in this industry, the push for even better processes and project management is continuing at a high pace. Demands are rising towards more agility and flexibility. Users expect the same adaptive behaviors and continuous delivery models as they are used to from their mobile devices. Rapid advances toward autonomous driving and open vehicle communication are straining standardization as they struggle to keep up with issues of legal and ethical responsibility, cyber security, and short-cycle re-certification after OTA software updates. illustrates these advances with the example of the current AUTOSAR adaptive SOA modelling. Joerg Schaeufele of Vector is convinced that "only novel development paradigms such as SOA will cope with the growing need for flexibility. **Fig. 6** shows how SOA is practically modeled and combined with legacy architecures.

Big industry players such as BMW, Bosch, Daimler, Ford, GM, Hyundai, Magna, Toyota and ZF are today considered the market leaders in embedded software technologies as well as drivers in global software engineering methods and collaboration tools. Overall software engineering technology advances are today heavily driven by such automotive manufacturers and suppliers. Examples include

> systems modeling, testing and simulation with models in the loop; the combination of several quality requirements such as safety,
> security, usability and performance to modern software systems;
> service-oriented advanced operating systems with secure communication platforms such as adaptive AUTOSAR (Automotive Open System Architecture);
> artificial intelligence in multisensor fusion and picture recognition for ADAS (advanced driver assistance systems) and autonomous driving;
> distributed end-to-end security for flexible remote software updates directly into the cars firmware;
> connectivity of cloud technologies and IT backbones with billions of devices for environment awareness, location-based services, online apps, remote diagnosis and emergency functions.

Dr. Ulrich Bodenhausen of Vector summarizes his wide experiences in introducing connectivity across the globe: "Connectivity means a fully shift of organizations, cultures and processes." Those who will not reinvent themselves in due time will disappear and replaced by hungry new IT and IoT companies. Due to the fast growth of automotive software with a double-digit CAGR (compound average growth rate) in market penetration and rather solid margins many

Figure 6: Service Oriented Architectures (SOA) Facilitate Flexible, Open and Dynamic Distributed Systems
Image rights: Vector Informatik GmbH

new players such as Apple, Google, and Huawei are getting into this market.

Connectivity is opening the doors for technology innovation, new business models, and collaboration schemes across industries. Multimodal mobility will connect previously separated domains like cars and public transportation. Mobility oriented services such as car sharing create fully new eco-systems and business models far away from the classic buy your own car approach. Connectivity has transformed once isolated systems such as a car or a medical implant into a distributed IT system with cloud access, over-the-air functional upgrades, and high-band-width access to map services, media content, other devices and surrounding infrastructure. Future is arriving while some are still tuning their hamster wheels. Dr. Christof Ebert of Vector thus quotes the technology strategist Hermann Kahn who already observed several decades ago: "Everybody can learn from the past, today it's important to learn from the future."

For more information on connectivity, cyber security and current technology advances plus access to the mentioned presentations and videos, please visit: www.vector.com/consulting

Dr. Christof Ebert
is Managing Director of Vector Consulting Services.

Contact:
Dr. Christof Ebert
Tel. +49-711-80670-1525
E-Mail: christof.ebert@vector.com

Lean Requirements Engineering

Traditional development processes that reached innovation and quality through a cumbersome development process are no longer appropriate. Studies of Vector Consulting Services show that especially in critical software-intensive systems, the cost of rework on the product life cycle can be reduced by 30 to 50 percent by improving the requirements engineering.

Increasingly, companies put their development on a diet. Cost reduction is the dominant goal, but it is also about shorter cycle times, faster implementation of innovations and thus to compete in global competition. Lean and agile means achieving business objectives with an efficient and flexible process. The case study shows how lean requirements engineering (LeanRE) can be used for cost optimization. Our framework was the implementation of Lean Development in medical technology.

The push for innovation in healthcare is accelerating constantly and accordingly the demands placed on medical technology are growing. In order to shorten lengthy development and approval processes, many companies change their development approaches towards iterative and concurrent engineering approaches. However, without systematic requirements engineering the probability is high that a project fails.

Software-intensive critical systems in medical technology are under immense market pressure. While they must be technologically innovative, and exhibit safety without any compromise, the global markets require an ever shorter cycle time with simultaneous efficiency pressure. The answer to this is increased productivity through lean and efficient development processes.

For example, if quality requirements are misunderstood, problems will arise with concerning usability, security and performance. Since clinical workflows are difficult to model due to their high combinatorial complexity and a rather low standardization but also their ad-hoc character, medical requirements engineering must exhibit specific techniques for medical and health care needs.

We want to show the benefits of systematic requirements engineering by means of a case study with development of an imaging platform for medical technology. A distributed IT system was developed that is specifically adapted according to the respective hospital needs and environments. It included more than five thousand requirements and had several hundred developers at different locations worldwide. Lean Development was combined with requirements engineering, to specify for customer-oriented requirements and progressively refine the outcome. The need for a clear feature orientation was met with product line engineering. Our starting point for the improvement was an analysis of the current state of practice.

Based on these observations, the life cycle has been carefully adapted. Unlike typical increment structures, which are derived from the requirements based on dependencies, our focus was a direct reference to the requirements development. The implemented approach 'Feature-Oriented Requirements Engineering', based on the principles of Lean RE, had the following properties:

✚ Feature Model, which is maintained for internal dependencies (e.g. variant management, testing, and documentation) and external relationships (e.g., marketing, product management) and is at the heart of all engineering artifacts. All requirements are modeled first at the previous feature model and evaluated before they are specified in detail and taken into the product.

✚ Value-oriented evaluation of all features in order to preserve the economic perspective of Lean Development from the requirement elicitation to the feature prioritization to delivery. The marginal value of new and updated functions must be transparent at all times.

✚ Graphical modeling of workflows to support the user and usage of every requirement, and to quickly uncover dependencies as well as errors. The user benefit must be tangible on the »

Figures: Vector Consulting

The marginal utility of new and changed functionalities needs to be comprehensible at any time.

basis of workflows in the language of the user, because only on such basis a market-oriented product development and positioning is possible.

✚ Architecture-based modeling provides a clear link between the external view (Why? business benefits; What? requirements, features) and the internal perspective (How? architecture, modules, code, test cases) and thus achieves consistency without overheads. The existing architectural model has been adapted to provide the link at any time from the feature model, and in order to model and maintain variabilities.

✚ Incremental Requirements Engineering with a direct link to project management in order to achieve earned-value implementation.

Practical Experiences

Our experiences with Lean RE in the field of medical technology can be transferred well to other application domains. Let us briefly highlight the major experiences in implementing Lean Requirements Engineering in concrete projects:

1. Separation of problem and solution space. The development of a solution affects the view and understanding of the underlying problem. With the clear separation of the problem and solution space, the subsequent change requests can be effectively reduced. We achieved a better handling for early impact analysis and estimation of the necessary changes to existing architecture and systems.

2. Increased understanding of customer requirements. Users often have an implicit understanding of the requirements. However, they are often unable to articulate precisely what they need. They want to achieve a benefit which still needs to be described and translated to specifications. These requirements should be refined as early as possible. We use a glossary that guarantees the connection to the user language, as well as rapid prototyping to create concepts for user interfaces.

3. Storyboards for workflows with high share of user interfaces. The Storyboard is a container for related requirements, user interface visualization and test cases. It allows representing the case of application but also failure modes, and is refined iteratively with time boxing. For security requirements we specifically apply abuse cases, misuse cases and confuse cases.

4. Structured requirements for planning and budgeting. At the beginning of a project, the additional effort to provide better structure must be invested. According to our experiences, project managers should estimate about 8-10 percent of the project budget for requirements and related architecture analysis. This will improve understanding of dependencies between features and thus reduces errors from not understanding complex behaviors and previous implementations. According to our experience, the appropriate requirements structure will be achieved with three to four iterations. This achieves the necessary stability for the further evolution of the solution. The requirements should be best organized hierarchically along certain functional, that is clinical, domains.

5. Development of a suitable model for traceability. Ad hoc links between requirements and results of the specification to the test cases and documentation without a clear strategy will always reduce the quality and thus foster rework. Therefore, traceability must be established with a clear focus and content orientation. It must define the necessary artifacts that need to be linked, specify the granularity and describe mechanisms for methodology and its implementation in underlying ALM/PLM tools. For medical-technical applications, a well-maintained and intuitive traceability is crucial for market approval such as by FDA regulations. It is essential to maintain the dependency relationships. We recommend to determine the responsibilities for continuous maintenance of traceability relationships in advance and to foresee the necessary effort. This also facilitates an early impact analysis and earned-value based progress tracking and cost control on the basis of implemented and tested requirements.

6. Disciplined use of standards and reviews. Only with clear internal guidelines for requirements engineering, for the work product structure and for procedures and responsibilities, there is some guarantee that the results are consistent across all products for solution development and

Contact

Vector Consulting Services GmbH
D-70499 Stuttgart
Phone + 49(0)711 80670 1525

www.vector.com/consulting

can be economically maintained, as is required in clinical practice today. We recommend the use of industry standards, which are adapted to the needs of the project. Document templates allow compliance with documentation standards. In addition, standards facilitate the necessary routine for reviews, which are often neglected under project pressure – with the result of complicated and time-consuming rework.

Conclusions

The introduction and systematic implementation of lean requirements engineering needs effort both in development as well as at the sources of requirements, i.e. product management, product marketing and sales. Often this effort is seen as too high and too time-consuming, so that the requirements continue to triple ad hoc in the project where they are implemented in piecemeal fashion and with lots of rework – until once more a project misses its goals or must be aborted. From the many failed change projects which we have seen we know the dilemma: Improvements in methodology, processes, training and tools are not addressed because the initial effort to initiate these improvements is considered to be too high. No surprise that 60% of all change projects fail. It is not the lack of technology, but the lack of professional change management competence.

Therefore, it is necessary to capture the benefits of Lean Requirements Engineering quantitatively and to prove that the expenditure for the Requirements Engineering create a return on the investments. A profound cost-benefit analysis, which we conducted in the underlying medical case study showed that the feature-oriented requirements engineering approach can save about 15 to 20 percent of an annual R&D budget. The benefits are distributed as follows across project cost: Product definition (25 percent), project planning (23 percent), architecture (7 percent) and improved verification and reviews (45 percent).

The benefits of lean requirements engineering can be related to several factors. Some, such as better delivery accuracy or less rework, create directly tangible benefits. Others, such as customer satisfaction, are more opportunistic and become tangible in the form of good, lasting customer relationships with further projects. Overall, our experience has shown that a doubling of the effort for Requirements Engineering towards 10 per cent of project budget in the areas of methodology, processes, training and tool support will deliver project benefits of more than 20 percent. That is a return on investment of more than 4, even by only considering the directly measurable benefits. Reason enough for you to evolve your own requirements engineering towards ‚Lean'.

Dr. Christof Ebert
is Managing Director of Vector Consulting Services, where he supports companies worldwide in product development, IT and change processes.

Arnold Rudorfer
is program manager for the development of a system platform at Siemens Healthcare Diagnostics.

VECTOR

Functional Safety and Cybersecurity
Consistent and effective implementation

...with experts of Vector Consulting Services

Benefit from our solutions for Safety und Cybersecurity:

> Vector SafetyCheck and SecurityCheck for a fast evaluation of your product developement, competences, processes and technologies
> Proficient experts as interim Safety and Security Manager
> Preparation of safety analyses and concepts
> Efficient implementation of ISO 26262, aligned with ASPICE
> Implementing Cybersecurity consistently in technical product development
> Methods and tool support for security-oriented tests and hardening
> Development, specification and test of safety and security requirements
> Independent audits and assessments of your suppliers

www.vector.com/consulting

Vector Consulting Services is the preferred partner for optimizing technical product development.

Vector Consulting Services GmbH | www.vector.com/consulting

Improving Engineering Efficiency with PLM/ALM

Rising cost pressure is forcing manufacturers and their suppliers to jointly and consistently master product development. Our industry case study shows how a leading automotive OEM over time has achieved effective interaction of engineering processes, tools and people on the basis of product and application life-cycle management (PLM / ALM).
Its scope is first an introduction to PLM/ALM on the basis of a model driven engineering (MDE) for one or several products or product families. Second, PLM and ALM need tool support to the degree necessary to ease handling and drive reuse and consistency. Third, introducing MDE needs profound change management.
Starting from establishing the relevant engineering processes, we show how they can be effectively automated for best possible usage across the enterprise and even for suppliers. We practically describe how such a profound change process is successfully managed together with impacted engineers and how the concepts can be transferred to other companies. Concrete results for efficiency improvement, shorter lead time and better quality in product development combined with better global engineering underline the business value.

1. Improving Efficiency with Better Engineering Processes

Companies and entire industries are changing very fast at this time. Software and IT are the main drivers of innovation. Software-intensive systems as used in automobiles, aircraft, medical, transportation, utilities and industrial automation deliver today 50–70 percent of the value of these solutions, and this will further grow. The products and solutions must meet increasing quality requirements, but must be developed cost-efficiently, should be easily adaptable to new environments and must effectively exploit the advantages of modern technologies. At the same time new competitors are pushing new solutions on the market, and the technology landscape is increasingly cluttered.

This economic climate enforces two trends in engineering across industries, namely fast innovation and also cost efficiency. Companies invest in growth through innovation by developing new products and solutions. At the same time they are aware of the volatile market situation and thus challenge their development teams worldwide to be as lean and efficient as possible. A senior R&D manager of a leading automotive tier 1 supplier summarized it as follows: "Cars sell in quantities we have never seen. Markets around the world are keen to get latest technology. But we have learned our lessons from the recent recession and made engineering and production flexible to immediately react if sales drop."

There are numerous levers for engineering efficiency improvement. Many companies operate with distributed teams leading to fragmented processes and tool chains with heterogeneous interfaces, redundant and inconsistent data management and insufficient transparency which results have been achieved and what needs still to be done. The underlying root cause is the lack of a shared conceptual model of the product. In consequence, activities such as project management, pre-development and product engineering are rarely integrated well due to the diversity of stakeholders with individual knowledge about projects, products and processes. As a result, engineering results such as specifications, documentation and test cases are inconsistent, items like signals and parameters are arbitrarily labeled, changes create lots of extra work to make sure that nothing is overlooked, and reuse is hardly possible due to the many heterogeneous contents. This pattern is amplified when collaboration across supplier networks comes into the picture, as it is today normal in systems development.

A brief example shows the significance. A supplier is introducing MDE based on some methodology, processes and a modern tool environment that enables seamless collaboration across development centers and with partners and customers. In advance, cost-effectiveness was evident because the system was going to provide faster access to data and less defects and budget overruns due to improved change and configuration control. After the introduction phase however, a MDE tools environment was available, but did not deliver useful models. Engineers were still drawing their previous style pictures, without much modeling methodology. What happened? The tool was designed to support development and was integrated into the company-wide PDM system. Electronic developers but also product managers and project leaders could not work with it and created parallel systems for their documents, which they exchanged between one another using traditional methods. The solution would have been simple if it had been made clear, before introducing the new tool, which processes had to be supported and how these processes had to be first improved and then automated.

An objective-driven tool introduction, which achieves measureable improvements and is accepted by engineers, requires good preparation and implementation. Normally, multiple departments or entire divisions are affected by a change of engineering and tools processes. Before debating tools solutions, the processes have to be under control and the envisaged workflows and work organization must be understood and optimized with the impacted engineers. An efficient electronics development builds upon streamlined processes that are understood and practiced by all engineers, because workflows are well orchestrated and optimally support engineering tasks. Success with ALM and PLM implies that processes and tools are simultaneously improved.

2. Product Life-Cycle Management

Product Life-Cycle Management (PLM) and its equivalent in software, namely Application life-cycle management (ALM), is the overall business process that governs a product or service from its inception to the end of its life in order to achieve the best possible value for the business of the enterprise and its customers and partners. PLM/ALM combines processes, people and tools for the effective engineering of products – from their inception until end of service. It involves tacit knowledge of experts and explicit knowledge, codified in procedures, process and tools. PLM/ALM stretches from know-how to know-what and know-why.

There is a huge misconception that PLM/ALM is a product not a process. By definition, it is the process by which organizations manage the creation, deployment, and operation of software over its full lifecycle. In practice, PLM/ALM has been associated with tooling suites aimed at managing the tasks of this lifecycle, but vendors have rarely delivered on the promise of integrating the management of the full application lifecycle. PLM/ALM is only cost-effective if there are the right engineering processes and if they are automated and instrumented with appropriate tools [1,2]. PLM/ALM integration has been a problem in the industry for a long time. On the one hand, customers generally do not replace their favorite point solutions, such as Jazz and TeamCenter, requirements management with PTC Integrity, development tools like Matlab/Simulink or Rhapsody, and replace them with a single suite from one vendor. Even where the PLM/ALM part is replaced with a single suite, there are often other tools to integrate with, such as defect tracking, change management, document management, etc.

To work efficiently, engineers need to handle a multitude of processes and different forms of knowledge to be shared with colleagues across business processes and even beyond the borders of the enterprise [1]. PLM/ALM helps to integrate those along the entire life-cycle of a release or product or beyond to an entire portfolio as is illustrated in **Figure 1**. Many companies have realized in this fierce climate that their traditional rather organically grown tools landscape with isolated unconnected processes not only won't scale up but also limit their engineering productivity due to manual data exchange, too much rework, inconsistencies and insufficient reuse across products and platforms **(Figure 1, left side)**. A federation of processes and supporting tools with clear responsibilities improves efficiency by more consistency, quality, reuse and not the least employee motivation **(Figure 1, right side)**.

PLM/ALM needs both process and tools support. **Figure 2** shows this relationship based on a study at London Business School [1,2]. The horizontal axis portrays the degree of tools support in an engineering environment, while the vertical axis shows to which degree the processes are optimized. All four combinations of high versus low have an associated impact on engineering productivity. Obviously the upper right quadrant shows best performance impact, namely 20%. Introducing the right process first and emphasizing its understanding in engineering has a higher value (upper left field), than pushing an engineering tool without profound process understanding (lower right field). Note that the values of course depend on the specific scenario and environment. What matters here is the ratio between the four fields.

Figure 1: PLM integrates processes, people and tools for effective engineering

Tuning processes, improving project management, and establishing visibility on new product introduction – techniques well described by common improvement frameworks, such as CMMI [4] – will not yield sustainable benefits if not adequately supported by tools. The prospect of new, high-margin products, combined with the delayed impacts of resource allocation decisions, seduce product managers into starting more projects than their development resources can handle.

The scope of a PLM/ALM system is on the one hand the creation and management of engineering data in one common engineering data backbone and on the other hand the management of processes. PLM/ALM as a concept and solution applies to software engineering as well as to systems or hardware products. It applies to different types and sizes of companies, because it is not prescribing a solution suitable only for big companies but rather a clear focus on processes along the life-cycle. We use it for complex solutions with multiple hardware and software components as well as simple software services.

Today such environments need to satisfy a variety of needs for integrated systems engineering, namely

> Integrated business logic and one comprehensive data model for the entire E/E development process from requirements, through system, SW- and HW-design to test.
> Architecture Design and Management
> Collaboration environment for requirements and test Engineering, modeling of functions, components, networks and communication
> Variant management and product line engineering for effective reuse
> Support for quality requirements such as functional safety from requirements to concept, models, realization to validation, proof and acceptance

Figure 2: Tools without processes are nothing; processes without tools are not good enough

There are different ALM/PLM environments available today on the market. We will focus in this case study on Vector's PREEvision, a leading automotive PLM solution, due to its use by the company which we portray. Like the other state of the practice PLM/ALM environments, it provides a complete tool suite of integrated data and process management modules [3] **(Figure 3)**.

These state of the practice PLM/ALM solutions provide a repository and editor of different models **(Figure 3, left hand side)** and ways to link models to reality **(Figure 3, right hand side)**. This allows both a process-oriented way of working (rigid) and a repository-based way of working (flexible, all engineers share a bunch of models of the product). Compared to the more traditional code generation oriented MDE no process consisting of planned model transformations is defined or required.

Based on a rich and extendable data model for features representing the logical and the physical system architecture and the software architecture, PLM/ALM systems provide highly integrated use cases. For instance in configuration and change control, issues are connected to system data objects, the related realization date is fixed in the release planning module, the implementation time and effort are planned in the project management module, the change of the related software parts are managed in the source code management module and finally the test are planned and executed in the test data module.

Having standardized PLM/ALM solutions around a standard product life-cycle reduces the learning curve to allow focusing on real technical challenges instead of organization overhead.

3. Industry Case Study

Often the expected benefits from modeling and PLM/ALM tools are often not visible. They are perceived as complex and employees are frustrated and continue working with their current work practices. New processes and interfaces create additional frictions and delays. Our consulting projects show that the root cause is often the same: Implementation of efficient processes with adequate tools support with sustainable results requires profound change management – which is rarely taken into account. This implies pro-active preparation way before a tool decision is made and good leadership, coaching and support through pilots and roll-out.

We will show a case study from a leading automotive OEM how to successfully introduce PLM/ALM and modeling.

Initial Situation: The main drawbacks of the initial situation were non-connected tool chains and a document driven subsystem and component engineering process with a lot of manually managed interfaces. Therefore the organization had to deal with a time-consuming and expensive reviews/rewriting process, a lot of redundancies and inconsistencies which often were not detected by these reviews, not standardized naming conventions and an ineffective way of developing and acquiring information. Further the information was spread over several data sources and a common version control was not really possible.

Objectives and Solution Approach: The objectives were to increase the engineering efficiency, the quality and consistency of the working products and to shorten the existing lead times in the electronic engineering. The solution approach was to implement a seamless information management of engineering data from function to software and hardware. That is, centralized information (i.e. model) management in a distributed and decentralized engineering process. The main characteristics of such a solution were alignment of processes, single source of information for collaborative teams, reusability of information, using commercially available tools, and continuous development with consistent underlying models.

Change process: To adequately support the introduction and change process we recommend a change methodology that is adapted to specific business goals **(Figure 4)**. PLM/ALM introduction is a long term activity which has to be funded by sufficient competence and budget and competent internal resources. The necessity of a change must be accepted broadly in the organization and the reliable commitment of the top management is a key. The process tool has to fulfill the main requirements of the customer and it must have the potential to grow with the growing needs of the user base. The change partner must be reliable and the chemistry between the main actors should secure the probability to realize a true long term partnership. These are all necessary conditions for success. If one is a false, failure result. But if they are true success is not necessarily guaranteed.

Resistance out of the organization, lack of money, changes in priorities, pressure from the user groups, theoretic discussions – these are not the exceptions, these are normal influences in such a project. The partners have to be able to deal with it and put the best personnel in the position of project leaders and methods/process engineers. Our customer made good experiences with an iterative-incremental development process – short implementation cycles, early validation with a small group of well experienced pilot users and a professional change management guaranteed mature deliveries for productive use.

In the rollout phase normally a small group of convinced pilot users are facing a big group of engineers, which are

Figure 3: PREEvision solution

open for the change in principle, but they are in the conflict to invest on the one hand more upfront time for learning the new environment and to have challenging objectives for their normal work products on the other hand .Furthermore, migration to a new way of working often includes "clean up" of older specifications, which also requires effort. And there is the small group of engineers, who are convinced, that the current, traditional way of working is the optimum and that there is no need for a change at all. We managed this situation with special trainings for key users and initiating "Jump Start Projects". In these projects the client's engineers were supported by well known and well with local specific engineering knowledge experienced consultants, which have been trained in PLM/ALM before. Such "Jump Start Projects" are focused and have a predefined duration (typical two months).

Lessons Learned: Utilizing a consistent product life-cycle and process repository is a necessary condition for reducing cycle time, as they reduce frictions of unclear interfaces and responsibilities as well as cutting rework because of inconsistent assumptions and cutting retrieval time for specific documents and work products. In implementing PLM/ALM and modeling support we found the following lessons learned:

> Concept: First improve the process then the tools based on concrete improvement objectives that are set, measured and used to correct deviations. Ensure consistency of features and products with a strong systems engineering. Specifically in distributed collaborative environments we see huge benefits from a single data backbone for consistent requirements, specs and models across all changes

> Development: Evaluate tools under realistic conditions. Agree specific requirements to the process and tools which are then used to drive changes. Support the interfaces to the various components and processes through traceability, automatic consistency checks and test automation.

> Deployment: Manage the changes as they impact the entire organization. Pilot changes, coach and train engineers, highlight power users that will set the pace. Introduce model-based development intelligently and step by step, focus on critical components, continuity of requirements to code and test cases, and improving processes in parallel.

> Operations: Support users and ensure continuous improvement. Measure the implementation, and try in each project ten to twenty percentage points improvement, at the spots where you want to put accents, for example 20% less cost variations, or 10% less cost in the test.

5. Summary and Conclusions

Inefficiencies are rampant when engineers are distributed globally and many different tools being used. Concrete efficiency and quality improvements with reduced rework and faster throughput have been showed by applying consistent PLM/ALM and modeling support. The efficiency and effectiveness of engineering processes directly influence engineering cycle time. For instance earlier defect detection in requirements or specs means faster and more comprehensive defect correction.

Without consistent model-based methods and adequate PLM/ALM tools support engineering will be in deep problems within a short time-frame. Increasingly complex requirements must be developed efficiently and with high quality throughout from system engineering to components. Working at a higher level of abstraction and automation will improve productivity and quality. Model-based development will play a crucial role in this evolution. The companies we work with share the same goals: Mastering complexity, improve product quality, shorten development time, and plan functions in different variations and versions for better reuse. The biggest challenges they see are their own learning curves and to keep consistency across features and products. Systems engineering still is undervalued and too often decoupled from application development. Roadmaps are maintained only for subsystems, and thus create variants with an overwhelming complexity. The business case is clear: With a degree of 30 % for modeling, error rates and development costs are reduced significantly.

What did we achieve with PLM? Since we consider knowledge management a regular management activity, we followed through like in other improvement projects by looking into performance results from real projects, as well as some process-related aspects, such as knowledge utilization. We found improved quality, reduced cycle time, improved engineering flexibility, reduced overheads improved communication, increasing alignment of processes and tools, and faster ramp-up time and skill management. Initially improved visibility and aligned terminologies and roles already brought huge gains, as they facilitate a borderless solution building inside the company.

What is next in PLM/ALM? Knowledge management must be better linked to business. Aligned business objectives and metrics must guide and monitor the development processes, the product lines and the project teams. Take as an example a mobile phone or game design with lots of embedded software. Being a commodity, business-oriented targets cover return rates or brand loyalty. Defects increase return rate and reduce brand loyalty with devastating business impacts. Looking to projects, products and processes will improve the design away from overly narrow focus on manufacturing aspects towards usability engineering. Knowledge and experience from past projects will be embedded into the underlying design processes. We stress the need for adequate knowledge management as a basis for success in product and solution development – an aspect going well beyond most PLM/ALM approaches of today.

We should not expect that all product and process knowledge and modeling methodology can be codified and made available via PLM/ALM. Personal contact will always be necessary to provide context and analysis. The support system should therefore be extended to facilitate interpersonal communication and evolve towards a global who-is-who not only at the operational product/project management level but also at the tactical and strategic level.

Correctness and completeness of the information is another aspect that needs to be worked on. We are convinced this cannot be achieved by imposing a reporting discipline only. It can only truly be achieved by ensuring that the provider of information is directly benefiting from making the information available. This can be achieved to continuing to evolve the system to ensure information can be captured as early as possible when it is required at the lowest operational level, e.g. by supporting period project reporting in presentation format to avoid information is presented first a set of slides for the project review before it is entered in the system.

Embarking on a state of the practice PLM/ALM solution combined with strong change management triggered by external support had helped to sustainably achieve the anticipated efficiency effects in the different engineering processes across the product life-cycle.

Figure 4: Vector change model for successful PLM introduction

Keywords:
ALM, Application Lifecycle Management, PLM, Product Lifecycle Management, Efficiency, Industry Voice

Literature:

[1] Ebert, C. and J. De Man: Effectively Utilizing Project, Product and Process Knowledge. Information and Software Technology (IST), ISSN: 09505849, Vol 50, No. 6, pp. 579-594, 2008.
[2] Ebert, C.: Value-driven Process Evolution. In: Marko Kuhrmann et al: Managing Software Process Evolution, pp. 61-76. ISBN 978-3-319-31545-4, Springer, 2016.
[3] PREEvision. http://www.vector.com/PREEvision
[4] Chrissis, M.B., M.Konrad and S.Shrum: CMMI. Guidelines for Process Integration and Product Improvement, ed. 3. Addison-Wesley, Reading, USA, 2011.

Christof Ebert
Dr. Christof Ebert is managing director at Vector Consulting Services. Before he had senior management positions for ten years with a global IT leader. A trusted advisor for companies around the world and a member of several of industry boards, he is a professor at the University of Stuttgart and at the Sorbonne in Paris. He authored several books. For many years he has been serving on the editorial boards of prestigious journals such as "IEEE Software" and "Journal of Systems and Software". Follow Christof on Twitter: @ChristofEbert.

Contact:
Dr. Christof Ebert,
Vector Consulting Services
Ingersheimer Strasse 24, D-70499 Stuttgart, Germany.
Ph.: +49-711-80670-1525
Company: www.vector.com/consulting