



►► PressBook



TECHNICAL PAPERS

on Embedded Network Solutions



5th Edition



Status: February 2014 - 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: SATZTEAM Fotosatz & Neue Medien GmbH, Eberdingen, Germany



Management leaders:
Dr. Thomas Beck (left)
Thomas Riegraf (right)

Dear reader,

This is the 5th edition of Vector's collection of technical articles. Once again, we have added interesting articles and user reports that have appeared over the past year. In 2013, the main topics of our technical articles published were on AUTOSAR, Functional Safety / ISO 26262 and Automotive Ethernet.

News in the AUTOSAR field include a case study prepared together with Daimler and Hella, the new Ethernet support in AUTOSAR as well as an AUTOSAR extension for J1939 for use in commercial vehicles.

Vector has a broad base of experience in the area of Functional Safety / ISO 26262 as well. This is reflected in technical articles on the safety-related MICROSAR OS SafeContext operating system and an article on our entire line-up of products and services for Functional Safety, which range from consulting via design of the E/E architecture to implementation in embedded software.

This year, projects are underway for networking ECUs with Automotive Ethernet at many OEMs and their suppliers, and it is likely that the first ECUs will go into production. In a number of articles, you can learn about the benefits of using the CANoe.IP software tool, the VN5610 interface and the AUTOSAR basic software MICROSAR.

We wish you much enjoyable reading.

Stuttgart, February 2014

Dr. Thomas Beck

Contents

Company Profile	Vector – The Company	
Management Interview	Transforming Personal Responsibility into Performance	
Serial Bus Systems	Serial Bus Systems in the Automobile: Introduction	1/0
	Motivation, advantages, tasks and architecture of serial bus systems in the automobile	
Serial Bus Systems – CAN	Serial Bus Systems in the Automobile: CAN	1/6
	Reliable data exchange in the automobile with CAN	
	CAN Gets Even Better	1/12
	Ways to transition from classic CAN to the improved CAN FD	
Serial Bus Systems – LIN	Serial Bus Systems in the Automobile: LIN	1/16
	Simple and cost-effective data exchange in the automobile with LIN	
Serial Bus Systems – FlexRay	Serial Bus Systems in the Automobile: FlexRay	1/22
	FlexRay for data exchange in highly critical safety applications	
	Beyond FlexRay	1/28
	Requirements on a modern development environment	
	Case Study: Hispano-Suiza	1/31
	Wireless Analysis in a Multi-Protocol CAN Environment	1/32
Serial Bus Systems – Automotive Ethernet	Acquiring Bus Data Wirelessly from Multiple Vehicles	1/36
	Top-notch precision despite "air interface"	
	IP and Ethernet in Motor Vehicles	1/42
	Challenges for the development tool, illustrated by today's applications	
	Challenge of Ethernet Use in the Automobile	1/48
	Flexible interfaces and software tools simplify ECU development	
	New Perspectives on Remaining Bus Simulation for Networks with SOME/IP	1/54
	Validating automotive IP network elements	
	AUTOSAR Learns Ethernet	1/58
Serial Bus Systems – ITS-G5	Testing Car2x Applications	1/62
	Requirements for test tools based on example of the road works warning	
	Car2x – From Research to Product Development	1/66
	How automotive OEMs and suppliers are successfully completing production Car2x projects	
Serial Bus Systems – MOST	Serial Bus Systems in the Automobile: MOST	1/72
	MOST for transmission of multimedia data	
Development of Distributed Systems	Tool-supported Data and Process Management at MAN Nutzfahrzeuge AG	2/0
	An integrated development database manages all engineering data in the E/E development process	
	From Pilot Studies to Production Development	2/4
	Efficiency and quality in calibrating transmissions	
	Case Study: Continental	2/9

ECU Testing	Quick Paths to a Comprehensive Remaining Bus Simulation Creative virtual networks without programming expertise	3/0
	Model-Based Development of ECUs Software Simulation with MATLAB/Simulink and CANoe	3/4
	Model-Based Design Simplifies ECU Tests Mastering Variation	3/10
	Comprehensive Communication Tests for ECUs Developed at Volkswagen Group Identical test environment for both OEM and suppliers	3/14
	Test Bench for Complex ECU Networks	3/18
	Hardware Simulation for the Challenging Unimog Tire Pressure Control System Individuality of omnibus features requires a flexible test system	3/24
	ECU Testing with XCP Support A look behind the scenes	3/28
	Case Study: Nippon Seiki Co.	3/31
	Efficient ECU Tests with Fault Memory	3/32
	Flexible Test Systems for Efficient ECU Testing Functional testing with error simulation at the developer's bench	3/36
	Porsche Validates Gateway ECUs Automatically Real-time analysis in driving trials supplements laboratory tests	3/40
	Case Study: Bertrandt	3/45
	Easy Access to Bus Analysis	3/46
	Network Access with Short Reaction Times Automotive interface with integrated real time computer	3/48
	Case Study: Ford	3/51
	Seamless Logging on Test Drives Acquire vehicle data reliably with data loggers	3/52
	Case Study: GETRAG	3/55
	A Bright Idea Logging CAN bus data for flight tests	3/56
Vehicle Diagnostics	Automatic Validation of Diagnostic Services by Use of a Diagnostic Integration and Validation Assistant at Opel Automated testing of diagnostic implementations based on the example of the Opel Insignia	4/0
	The Standard Mix Does it: Diagnostics with AUTOSAR and ODX – Part 1: Diagnostics with AUTOSAR	4/8
	The Standard Mix Does it: Diagnostics with AUTOSAR and ODX – Part 2: ODX in the AUTOSAR Development Process	4/12
	Diagnostic Tools for WWH-OBD Implementation of the new requirements for OEMs and suppliers	4/18
	From Diagnostic Requirements to Communication Standardization is the trend in the development of automotive electronics	4/22
	Diagnostics from a Distance Interactive diagnostics for vehicles worldwide	4/26

Contents

ECU Calibration	Use of XCP on FlexRay at BMW	5/0
	XCP-on-FlexRay at Audi	5/4
	AUTOSAR-compatible XCP software modules for FlexRay ECUs	
	Case Study: HOERBIGER	5/9
	XCP at the Focal Point of Measurement and Calibration Applications	5/10
	Quantum Leap in Microcontroller Measurement Technology	5/16
	Innovative ECU measurement concept for maximum data rates with minimal effects on execution time	
	High-speed Measurements for Electric and Hybrid Vehicles	5/20
	New measurement concepts enable high data rates and frequent sampling times	
	Analyze Large Quantities of Measurement Data Rationally and Flexibly	5/24
	Efficient Analysis of Model Behavior in ECU Function Development	5/28
	Visualize and parameterize Simulink models easily and efficiently	
	Accelerated Turbocharger Development with CANape	5/32
	Efficiently developing control concepts with a cost-effective rapid prototyping solution	
	Riding on the Razor's Edge	5/36
	Optimal parameterization of an engine controller for drag racing	
	Optimizing Driver Assistance Systems	5/40
	Verification of object recognition algorithms by driver assistance systems	
AUTOSAR	AUTOSAR: New Paths to ECU Software – Part 1	6/0
	Iterative collaboration between OEM, TIER1 and software supplier	
	AUTOSAR: New Paths to ECU Software – Part 2	6/6
	AUTOSAR in practice – Life cycle of AUTOSAR ECU software	
	Early Testing of AUTOSAR Components	6/10
	Analyzing individual software components instead of entire ECUs	
	Revealing Runtime Conflicts	6/12
	AUTOSAR operating system directly measures task execution times	
	Implementing a CAN-MOST Gateway with AUTOSAR Basic Software	6/16
	Analyzing AUTOSAR ECU Software with XCP	6/20
	Convenient debugging by extending the AUTOSAR basic software	
	Plug and Play Solution for AUTOSAR Software Components	6/26
	AUTOSAR ECU Development Process Using DaVinci and MICROSAR from Vector	6/32
	Case Study: FAW	6/39
	AUTOSAR Methodology in Practice	6/40
	On-Board Diagnostics meets AUTOSAR	6/44
	Case Study: AUTOSAR Test System	6/45
	AUTOSAR in Heavy-Duty Vehicles	6/46
	Integration of J1939 in AUTOSAR	

Functional Safety	Model-based Functional Safety in E/E System Development	7/0
	Development of Safety-relevant ECU Software	7/4
	Vector and TTTech to Partner	
	Recipe for Safe Software	7/8
	Development of ECU basic software according to ESO/DIS 26262	
	Intrinsic Safety of AUTOSAR Basic Software	7/12
	Integrating software components with different safety relevance in one ECU	
	SilentBSW – Silent AUTOSAR Basic Software for Safety-related ECUs	7/16
	Coexistence of safety-related and non-safety-related software in one ECU by protecting memory areas	
	Practical Implementation of Mixed-ASIL Systems	7/20
	A certified operating system simplifies the development of safety-related software	
	Seamless Implementation of ECU Software based on ISO 26262	7/24
ECU Software	Timing, Memory Protection and Error Detection in OSEK Systems	8/0
	Requirements on a real-time and multitasking operating system	
E-Mobility	ECU Testing for Electric and Hybrid Vehicles	9/0
	Intelligent measuring the dynamic power consumption	
	New Bus Interfaces for Electric/Hybrid Vehicle Development	9/4
	Realtime performance enables flexible use in the EV/HEV development field	
	Electric Mobility Makes Great Strides	9/8
	626.6 kilometers under real conditions on one battery charge	
	Convenient Charging of Electric Vehicles	9/14
	Smart Charging with MICROSAR IP enables flexible charging processes and easy payment	
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	10/24
	Factors of success in electronic development	
Open Networks – ISOBUS	Automatic Interoperability Tests for ISO11783 Systems	10/28
	Universal test solution assures compatibility	
	Forging New Pathways in Testing ISOBUS Task Controllers	10/32
	Simulations replace inflexible and time-intensive test methods	
	Better Test Quality by Automation	10/38
	Automated HIL test system ensures ISOBUS functionality of agricultural machines	
Open Networks – CANopen	Prototyping and Testing CANopen Systems	10/44
	Reaching goals rapidly with minimal effort	
	Automatic Testing of CANopen Devices	10/48
Consulting	Improving Electronic Engineering Efficiency with Automated Processes	11/0
	Functional Safety Industry Best Practices for Introducing and Using ISO 26262	11/6
	Managing Risks in Global Software Engineering	11/14



► ► Vector - the Company

Milestones / Key Data

- 2014** Retirement of the three founders from the executive board, Founding of Vector Italy and Vector South America
- 2013** Founding of Vector Austria, 25th anniversary of Vector
- 2012** Fifth managing director is Thomas Riegraf, founding of office for aerospace customers in Hamburg/Germany
- 2011** Founding of the 2 Vector foundations, Vector has more than 1,000 employees worldwide
- 2010** Founding of development center in Karlsruhe/Germany
- 2009** Founding of Vector Great Britain, Vector India and Vector China
- 2008** Vector has more than 800 employees worldwide, move into the third company building in Stuttgart
- 2007** Founding of Vector Korea
- 2006** Founding of office in Regensburg/Germany
- 2005** Vector has more than 500 employees worldwide
- 2004** Move into the second company building in Stuttgart, founding of customer centers in Brunswick and in Munich/Germany
- 2002** Founding of Vector France and Vector Scandinavia, Vector Informatik is DIN EN ISO 9001:2000 certified
- 2001** Fourth managing director is Dr. Thomas Beck, move into first company building in Stuttgart, founding of Vector Consulting
- 1999** Vector has more than 100 employees worldwide
- 1998** Founding of Vector Japan, Vector Informatik is DIN EN ISO 9001:1994 certified
- 1997** Founding of Vector North America
- 1996** Sale of the first CANoe and CANape license
- 1994** Vector has more than 25 employees
- 1992** Sale of the first CANalyzer license
- 1988** Founding of Vector by Martin Litschel, Eberhard Hinderer and Dr. Helmut Schelling

The Company

Vector provides OEMs and suppliers of automotive and related industries a professional and open platform of tools, software components and services for developing embedded systems.

Customers worldwide place their trust in the solutions and products of the independent and self-contained Vector Group.

Reliable Partner with Quality

For many years, our customers have realized what they have in Vector: a reliable and competent partner for efficient solutions in electronic networking. This reliability is based upon the factors below:

- > Globally integrated processes and standards
- > Practice-based procedures and regular reviews
- > Regular successful ISO 9001 certification for the Vector group
- > Implementation of actual standards like Automotive SPICE and ISO 26262

High Customer Satisfaction

The delivery quality and delivery time as well as consulting competence from sales, support, and consulting are not just maintained on a high level, but are constantly being improved.

The success: our customers judge us on a scale of 1 to 5 (where 1 is the best grade and 5 the worst) with an average grade of 1.6.

Excellent Working Atmosphere

70 % of our highly-qualified employees work in the development environment. Teamwork in modern structures ensures a working atmosphere that in the past years has been judged "very good" according to internal surveys.

On Location Worldwide

- | | |
|-----------------|----------|
| > Germany | > India |
| > Austria | > Italy |
| > Brasil | > Japan |
| > China | > Korea |
| > France | > Sweden |
| > Great Britain | > USA |

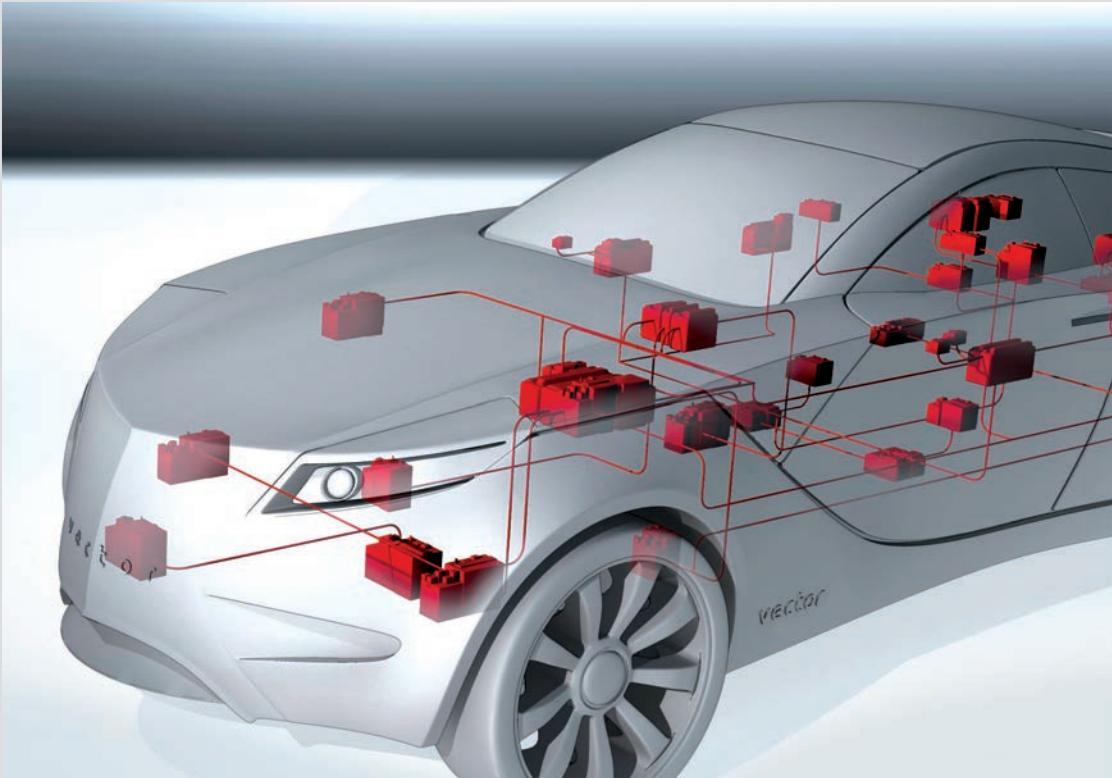


Engineering Services, Consulting & Training

Vector provides you with both technical consultation and adaptation or customization of Vector tools in service projects.

The knowledge of our experienced employees is your advantage in coming up with efficient and customer-specific solutions.





Diagnostics

Tools to describe, implement and test the diagnostic functionalities that are required to run diagnostic services on an ECU. Diagnostic functionalities are set up during the ECU development, stored in a consistent database and used during the ECU's lifecycle.



ECU Testing

Tools and services that allow the test of ECUs in all development phases, check the functionality of prototypes or execute regression and conformity tests.



ECU Software

Embedded software like real time operating systems (RTOS) and communication modules for e. g. CAN, LIN, FlexRay and Ethernet. Software for memory management and for re-programming of ECUs. Basic software for AUTOSAR. Development services for software components.



Development of Distributed Systems

Tools and Services to design and develop a network of ECUs. Tools to simulate, analyze and test the communication of the network.



ECU Calibration

Tools to access the ECU at run-time. This allows acquiring and modifying measurement data and parameters, so the ECU algorithms can be modified and optimized.



Consulting

Vector offers consulting for optimizing your technical product development, the underlying business processes and tools, as well as the successful integration of changes.

Transforming Personal Responsibility into Performance

Exclusive interview with the five managing directors



On the occasion of Vector Informatik's 25th anniversary, the German AUTOMOBIL-ELEKTRONIK magazine spoke with the five managing directors: Dr. Thomas Beck, Eberhard Hinderer, Martin Litschel, Thomas Riegraf and Dr. Helmut Schelling. Interview questions and answers touched on subjects relating to the company, its strategy and jobs in Germany, the technical topics of IP/Ethernet and AUTOSAR and the company's history.

Vector has experienced rapid growth in recent years. What do you attribute this to?

Martin Litschel: The market for electronics in automobiles has experienced strong growth over the past 25 years, which in turn has allowed us to grow as well. But the total package that we offer has to be right, and part of that package is our closeness to the customer.

Eberhard Hinderer: Naturally, our products have to be right too, and we are continually developing and improving them. Over 50% of our employees work in the development area. We must also have a presence in all countries in which automotive electronics are developed, so that we can offer support – including training that is conducted in local languages.

Dr. Helmut Schelling: Partner-based relationships with our customers are also important. Long-term customer satisfaction is something we value highly, and I think that our customers perceive this too.

Where do you see Vector's position in the international market?

Eberhard Hinderer: Vector has several product lines that compete in different sub-markets. The situation is not the same everywhere. In some cases, we play a leadership role with our products, but in other cases we are in a position where we still strive to catch up.

Dr. Helmut Schelling: In the market for bus analysis tools for networking, we are certainly the clear market leader with our

CANoe and CANalyzer products. I am proud of the fact that the majority of embedded basic software in the market comes from us – an independent company. For 18 years now, we have been creating CAN embedded software for ECUs, which has now branched into AUTOSAR. Vector also plays a key role in the AUTOSAR field.

Martin Litschel: We see our primary role as a partner in the development of automotive electronics – and our support spans the entire development process.

Dr. Helmut Schelling: Another strong aspect of Vector is its independence; we do not have any large corporation behind us. Even our largest customers represent less than 10% of our total sales revenue, so we do not depend on any specific customer either. We are not listed on the stock exchange, so we do not need to report quarterly results. This lets us focus very intently on long-term planning.

Vector is now owned by two trusts and is not listed on the stock exchange. Why did you choose this corporate construct?

Eberhard Hinderer: Founding of the trusts has not really changed the original company Vector Informatik GmbH. We four owners only transferred our partnership shares from our private ownership to the two trusts: one private trust and one non-profit trust. This empowered us to safeguard the long-term future of the company, because it is no longer possible to sell the company.

Martin Litschel: We three company founders (Martin Litschel, Dr. Helmut Schelling and Eberhard Hinderer) created the non-profit trust to give back a share of the company's large growth and success back to its employees and to society, because in the end we all worked on our success together. Therefore, 60% of the total company was split off to the non-profit trust.

Dr. Thomas Beck: At the Robert Bosch company, where Mr. Schelling, Mr. Litschel and I worked previously, we saw how well this good model of a trust ownership arrangement can work. We implemented a similar construct that was best suited to our size. This has let us preserve jobs over the long term. The non-profit trust receives a majority of dividends and invests these funds in charitable projects.

Vector has already been recognized as one of the best employers in Germany. What do you attribute this to?

Dr. Thomas Beck: At our company, people probably notice the relative lack of hierarchy, because one-to-one interactions and mutual respect are important to us. Furthermore, practically everyone at our company has a good sense of technology, so we can actually live out our motto "From engineers for engineers".

Dr. Helmut Schelling: On the other hand, we also cultivate a business climate that embodies such events as a company excursion and a holiday celebration as well as profit sharing in good years. We are also making efforts to be very open with information.

Thomas Riegraf: Our working environment is pleasant, and one very important aspect of this is that work tasks are interesting and

challenging. Our colleagues can really immerse themselves in their work and drive things themselves. That is important to us: assuming responsibility and transforming it into performance.

Dr. Helmut Schelling: We place less emphasis on the achievements of individuals than we do on the achievements of teams.

Dr. Thomas Beck: Moreover, we are on a long-term path. At Vector there are no sudden austerity or catch-up programs, because we set our sights for the long term.

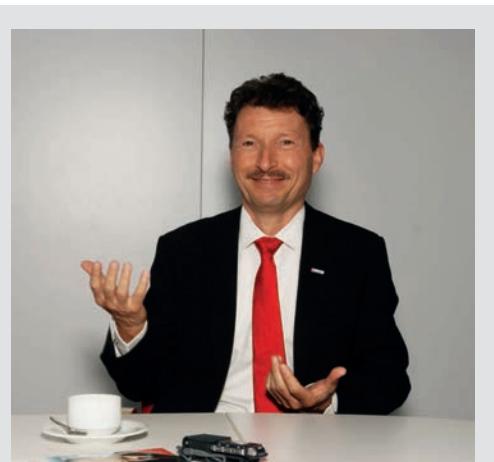
Dr. Helmut Schelling: I believe we have succeeded in preserving the atmosphere of a small company in the organizational structures that we have today.

In our first year of business, we already introduced a policy of addressing new employees informally (German informal "Du" as opposed to the formal "Sie" for "you") – and we still offer this option to new employees today. We in upper management are also accessible to everyone unless we are at a meeting; our doors are nearly always open. Ever since our first year, we have capped off the work week by meeting with employees in a casual atmosphere for a drink together on Fridays. Not everyone comes, of course, but around 60 people are always there, and we as managers are there among them. So, it is not too difficult for a developer to come and talk to us, and we also get feedback and a connection to our employee base.

Eberhard Hinderer: We conduct an anonymous survey of employees every year as well. We take the results of this survey very seriously and make efforts to counteract any deficits we find, because we really want to meet the standard of being a top employer.



Dr. Thomas Beck: "At our company, people probably notice the relative lack of hierarchy, because one-to-one interaction and respect are important to us."



Eberhard Hinderer: "We also conduct an anonymous survey of employees every year, ... because we really want to ... be a top employer."

How do you distribute development activities among the individual countries in which Vector is represented?

Dr. Helmut Schelling: CAN, LIN, MOST, FlexRay and AUTOSAR were all developed in Germany, and from there they circulated around the rest of the world. That is the primary reason why we are at the right location here. We were able to participate in developments and take our products to the rest of the world right from the very start.

Dr. Thomas Beck: If innovations were to occur in another part of the world, our business would be at tremendous risk, because we would then have to intensively shift our development activities to that region – not for reasons of cost, but because of delays and gaps in time and know-how.

Dr. Helmut Schelling: Just as Silicon Valley is the center of the semiconductor and computer industry, Germany is the center for innovations related to the automobile. That is why we are expanding our employee base primarily in Germany, especially here in Stuttgart.

What significance do Ethernet and IP have for Vector and the automotive industry?

Dr. Helmut Schelling: Ethernet and IP play a very large role for Vector. I look forward to every new bus, because then we can once again offer a new option for our tools. We supported the IP protocol in our tools very early on – in diagnostics and communication. In addition, Vector developed hardware with BroadR-Reach as the Physical Layer with related software early on. Vector also has a solution for communication between electric vehicles and charging

stations over IP – and it is the embedded packet most widely used for this application. We always have the same goal: Vector supports its customers with what they need today and what we believe they will need tomorrow.

What solutions can we expect from Vector in the next few years in the Ethernet and IP area?

Dr. Helmut Schelling: We will be implementing 100% of what AUTOSAR has specified in the embedded world; the same applies to the areas of diagnostics and electric car charging.

Martin Litschel: We have already developed hardware that can be used to split Ethernet lines, so that we can tap into what communication partners are exchanging with one another. All those customers who are already working with Ethernet today are implementing this hardware intensively. The primary tools CANoe and CANalyzer fully support the Ethernet protocol. We support Ethernet comprehensively, just as we support other bus systems.

Dr. Thomas Beck: Vector offers a lot of development support tools that offer such functionalities as testing, validation and monitoring. When a new bus arrives, we simply need to integrate this bus protocol in all of our products – ranging from basic software to tools.

Dr. Helmut Schelling: In the IP/Ethernet area, we are responding to customer requirements and are not simply being zealous about promoting our own proprietary solution on the market – this is similar to the way we have handled CAN and ASAM standards. We contribute our know-how towards developing commonly shared standards.



Martin Litschel: "We see our primary role as a partner in the development of automotive electronics – and our support spans the entire development process."



Thomas Riegraf: "Our colleagues can ... drive things themselves. That is important to us: assuming responsibility and transforming it into performance."

Martin Litschel: In our Ethernet solution, we do not experience any problems with the IT departments of our customers, because we have not taken a very esoteric approach – we know the requirements of our customers. Our Ethernet interface connects to a notebook or PC via USB and not via an Ethernet port. It can be operated just like MOST, FlexRay, LIN or CAN.

Dr. Helmut Schelling: We are taking a very proactive approach here. In the embedded environment, for example, we have a dedicated employee who addresses issues related to AVB. AVB stands for Audio Video Bridging via Ethernet, which is a sort of replacement for MOST. So, we are building up our own know-how in this area and are aware of the significance of AVB for our customers.

What are Vector's plans?

Dr. Thomas Beck: Today, we have a lot of new products ready to launch; we will see how they develop in the future. One of the key themes here is test data management. Another rather new theme is architecture support, and here our acquisition of PREEvision represents a big step forward.

For a long time, we have been actively working in the field of configuration databases, which we refer to as engineering backbones. We are involved with a good E/E pilot project at Volvo, but in this case we have also learned how it is very difficult for a large OEM to introduce such a technology into their business. The Volvo project was kicked off in 2008, and Volvo gave a presentation about it at the Automotive Electronics Congress in Ludwigsburg this year. It took five years to achieve full roll-out with 600 users.

What significance does AUTOSAR have for Vector?

Eberhard Hinderer: The largest product area at Vector relates to the theme of AUTOSAR. At least 200 of our developers are working on AUTOSAR software, and this does not even include developers of related tools.

Dr. Thomas Beck: AUTOSAR is like the antelope that was swallowed by the snake. AUTOSAR is the antelope, and its hooves are still dangling outside of the snake. AUTOSAR has not been fully digested by the industry, and it certainly has not been systematically introduced everywhere. At this point, only around one percent of application software in the automobile is based on AUTOSAR, but AUTOSAR will arrive.

Dr. Helmut Schelling: AUTOSAR is often more complicated and expensive than development departments initially envisioned. So, there is still a lot of potential for improving existing AUTOSAR tools. Whether AUTOSAR is implemented will be decided by the tools, or at least they will have a major impact on its implementation.

Thomas Riegraf: It took a full ten years until AUTOSAR was used in production, and we will see the same phenomenon in the IP field, because IP is only being used in very few illustrative applications. What is important to us is that we support developers along the path towards its widespread use and continually improve tools for it.

A look back at the early days

What led to the founding of Vector 25 years ago?

Eberhard Hinderer: The gentlemen Litschel and Schelling worked at the company Robert Bosch back then, while I was employed at Hewlett Packard. I have known Helmut Schelling since we were children, and back then we were already very interested in electronics and computers. At some point, we arrived at the idea of founding our own company – initially with the fundamental idea of offering software development as a service. After half a year of careful deliberations, consulting and planning, we founded "Vector Software GmbH" on April 1, 1988. At that time, our office was a 80 m² floor in an apartment building.

Martin Litschel: When we founded the company, we already had a customer for whom we were writing software for numerically controlled machines. Our job was to take the CAD data of a drawing and use it to automatically generate machine instructions for the control software of NC machines, e.g. milling and turning machines.

Dr. Helmut Schelling: Since these computations required many vector calculations, we chose the name Vector from the many potential names for our company.



Dr. Helmut Schelling: "We will be implementing 100% of what AUTOSAR has specified in the embedded world; the same applies to ... diagnostics and electric car charging."

Why did Vector enter the field of automotive electronics?

Martin Litschel: In 1991, the machine building industry went through a major crisis, so we turned our attention fully to the automotive industry, in which we had already been working on individual projects for some time.

How did you arrive at the decision to offer your own products instead of just doing contract work?

Dr. Helmut Schelling: In principle, we had always planned on offering products, but we first needed a safe harbor, from which we could set out onto the rough seas of products. This worked out surprisingly well, and back in 1992 we entered the market with CANalyzer.

Martin Litschel: Very quickly, we recognized that there were certain problems in debugging the CAN bus. When, back in the 1990s, entire cars had to be recalled to the factory and development engineers needed to debug them, we decided to develop professional tools for this field. Back then, we participated in working groups within Mercedes-Benz so that we could hear, first hand from the engineers, which functionalities they needed.

Dr. Helmut Schelling: The CAN bus was developed back in 1983, and Martin Litschel was involved in this development, but the first tools for CAN did not come from us; they were primarily tools for protocol analysis on the bit level. We were fortunate to have entered the market somewhat later, so we were able to see the problems occurring with use of the CAN bus. These problems did not occur not so much on the bit level as they did on the logical communication level. Back then, CANalyzer arrived at the right time and with the right functionality.

This interview was conducted by Alfred Vollmer, editor of AUTOMOBIL-ELEKTRONIK magazine.

Translation of a German publication in AUTOMOBIL-ELEKTRONIK, 4/2013 (published August 2013)

**>> Contact information of the Vector Group:
www.vector.com/contact**



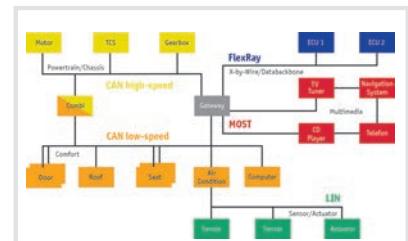
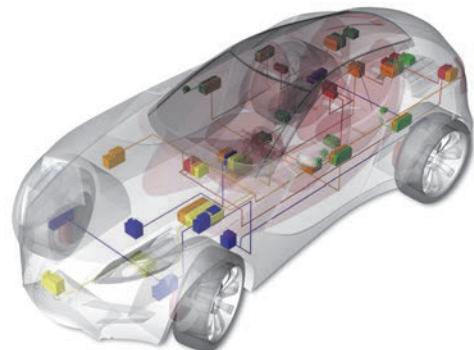
Eberhard Hinderer, Martin Litschel and Dr. Helmut Schelling (from left to right) in the days when Vector was founded.

Serial Bus Systems in the Automobile

Part 1:

Motivation, advantages, tasks and architecture of serial bus systems in the automobile

The share of electronic components in the automobile is growing from year to year. Electronics plays a decisive role, not only in satisfying primary customer wishes for better driving safety and comfort, but at the same time to achieve better fuel economy and reduced exhaust emissions. Another aspect that should not be underestimated is the contribution by numerous serial bus systems in the automobile. Many functions would not even be possible without data exchange between electronic components. This article offers some initial insights into the world of serial bus systems in the automobile.



Motivation and advantages of serial bus systems in the automobile

Recent history of the automobile is characterized by intensive electronification. The driving force for this originates primarily from customer expectations of a modern automobile which are becoming increasingly demanding. Moreover, legislators are continually placing stricter requirements on exhaust emissions. The rising competitive and cost pressures of globalization also produce constant innovative pressure. Automotive OEMs have found electronics to be a way to meet this multiple challenge. Particularly this is reflected in the migration of electronic control units (ECUs) into the automobile which began at the end of the 1970s.

At that time, the first embedded electronic systems still performed their tasks fully autonomously. However, very early it was recognized that by coordinating applications placed in different ECUs, it would be possible to increase vehicle functionality immensely. This was the motivation for integrating communication systems in the automobile.

Ahead of everything else, at that time it was electronic driving dynamics control that dominated advanced development. However the intensive wiring effort utilizing individual dedicated lines only permitted limited data exchange. As a way out of this dilemma, bit-serial data exchange via a single communication channel came into question. This single

communication channel integrates all individual communication channels and is referred to as a bus. Using this bus and associated serial interfaces it is possible to join all ECUs together into a network refer to as a serial bus system (Figure 1). In this context, ECUs are referred to as bus nodes.

Since the introduction of serial bus systems, the complex and often divergent types of wire harnesses in the automobile have become a thing of the past. Bus systems not only simplify project design and installation, but also reduce the weight and space required for wiring. Moreover, the lower

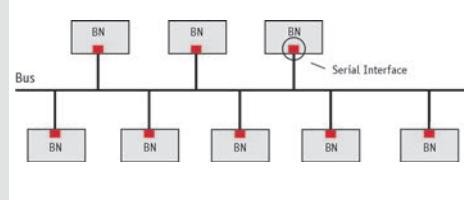


Figure 1:
Bus networking: All electronic control units (Black: Bus nodes) are joined into a system network, the serial bus system, by means of a bus and related serial interfaces.

number of connectors reduces the susceptibility to failure significantly. These many advantages face numerous communication tasks that must be mastered by the serial bus system. The most important communication tasks are discussed in the following.

Communication tasks

A precondition for trouble free serial data exchange is the unique allocation of the data to be sent to the bus nodes. Essentially a distinction is made between sender-selective and receiver-selective allocation (addressing). In case of sender-selective addressing the sender identifies the desired receiver by a unique bus node address. In contrast, in case of receiver-selective addressing the data to be sent are addressed. This means in principle that all data are available for any node to receive (Broadcast). Therefore all bus nodes have the task of filtering out data that are relevant to them. This is accomplished with the help of the address referred to here as identifier.

In order that the receiver acquire the data and address as one unit, the sender packs both of them together as a frame. A typical frame encompasses the address and data with a start and end recognition, which are primarily used to synchronize senders and receivers. A "frame" is also referred to as a "message".

The most pressing tasks of a serial bus system include real-time communication and data integrity. A distributed system can only fulfill its intended purpose if all data reach the destination node in time and without errors. A serial bus system's performance and field of application in the automobile substantially depend on the degree with which it can avoid, reject, detect and correct errors, and can guarantee timely data transport.

Data integrity

Quantitatively data integrity can be described as the residual error probability. This is a statistical measure of data integrity violation. Residual error probability is understood as the product of probability A that the transmitted data are corrupted and probability B that the corrupted data remain undetected. The data integrity of a serial bus system therefore depends first on the extent to which it avoids the corruption of data, and second on the degree to which it can detect corrupted data.

Various interactions related to electrical, capacitive or inductive coupling, as well as electromagnetic fields, come into consideration as potential causes of data corruption in the automobile. Specific sources responsible for corruption might be actuators, fan motors, high-frequency signals generated by the commutation process in DC motors and fast da-

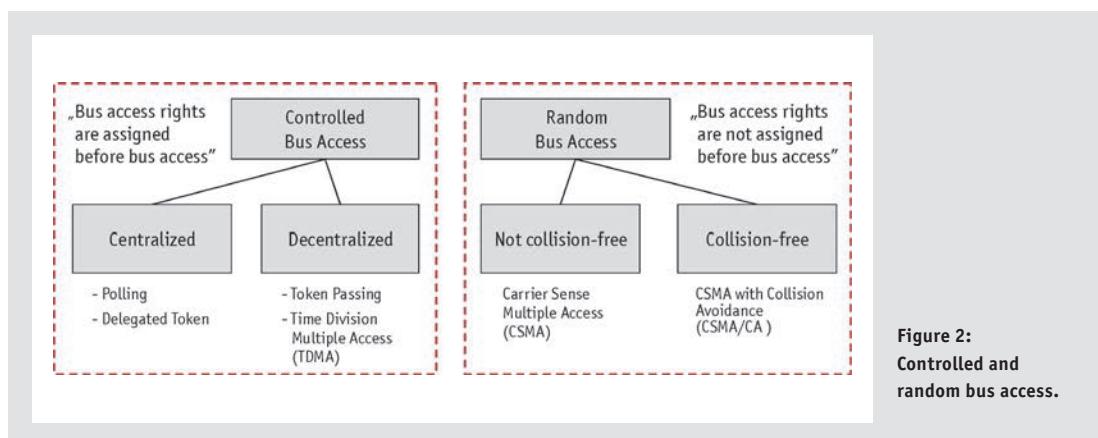


Figure 2:
Controlled and random bus access.

ta transmissions or reflections at the ends of buses. The more successfully these causes can be eliminated, the greater the noise immunity and more reliable the data transmission.

To enhance the noise immunity of a serial bus system, certain important measures are necessary. Besides shielding the transmission medium, as well as all electrical and electronic components, it is important to provide sufficiently large distances between data and power transmission lines and between electrical and electronic components. Furthermore, it is important to limit the data transmission frequency and number of data signal edges and their steepness, to apply the principle of differential signal transmission and finally to terminate bus ends with the characteristic impedance of the transmission medium. Even with optimal physical system design transmission errors cannot be eliminated entirely. Error detection mechanisms are therefore essential. Among the most frequently utilized methods is the checksum method, wherein the sender computes a checksum from the data block to be sent by a defined algorithm. It then sends this checksum at the end of the data block. Using this checksum the receiver is able to verify the received data block.

The more clever the algorithm, the shorter the data block to be protected and the longer the checksum, the better the algorithm's error detection ability. However, due to limited bandwidth and time requirements, a compromise must be reached between error detection ability and the ratio between data block and checksum size (transmission efficiency). Furthermore one must consider that the checksum itself is not immune to disturbances during transmission.

As a rule, after detecting a transmission error, error correction is needful, e.g. by means of an error-correcting checksum. However, unlike simple error detection that would require an explicit longer checksum. For efficiency reasons error-correcting checksums are not implemented in the automobile. The error correction happens by repeating the message: caused either by an error flag set by the bus node detecting the error, or automatically in the case of periodic message transmission.

Real-time capability

A system with real-time capability must be able to guarantee transmission of all data to be exchanged between the various bus nodes within a defined time window. Key factors

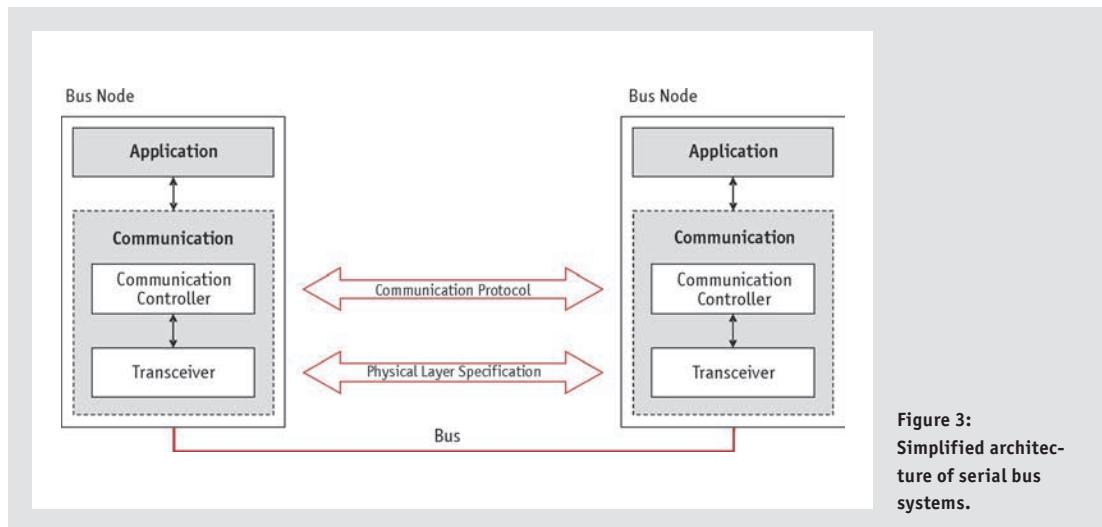


Figure 3:
Simplified architecture of serial bus systems.

here are the number and sizes of messages, the available bandwidth, and especially the type of bus access. In the latter case a fundamental distinction is made between controlled and random bus access (Figure 2).

In serial bus systems with controlled bus access, bus access rights are already clearly defined before the bus access. Such systems offer deterministic message traffic as an important precondition for attaining real-time capable serial bus systems. However, since the entire communication sequence is executed according to a schedule and cannot be influenced, serial bus systems with controlled bus access are characterized by poor dynamic behavior.

This disadvantage does not apply to bus systems with uncontrolled bus access. Each bus node has the right to occupy the bus at any time, e.g. in response to an event that just occurred. This produces very fast bus access; however there is the inherent risk of more or less acute collisions, depending on the event density, message sizes and the available data rate. These are not good conditions for achieving real-time capable data transmission.

Monitoring of the bus by bus nodes wishing to send significantly reduces the risk of collision. It can be prevented entirely by introduction of message priorities. However, these bus access methods based on bus monitoring and message priorities cannot guarantee timeliness. It is possible, that low-priority messages will be delayed unreasonably long.

Architecture of serial bus systems and bus nodes in the automobile

Based on the reference model for data communication specified by ISO (International Standardization Organization), the serial interface of a bus node in the automobile is typically subdivided into two (communication) layers: A lower layer (Physical Layer) and a layer above it (Data Link Layer).

Some of the tasks handled by the Data Link Layer are addressing, framing, bus access, synchronization and error detection and correction. These tasks are defined by a communication protocol. The Physical Layer specification, on the

other hand, covers all aspects of the Physical Layer, from the physical bus interface to physical signal transmission over the bus.

Generally the physical bus interface is implemented with the help of a transceiver. A communication controller covers the Data Link Layer. If all of the bus nodes within the system follow the same communication protocol and the same Physical Layer specification, then the fundamental preconditions for trouble-free data exchange between the bus nodes are satisfied.

In serial communication the sender's application passes to the communication controller the data block to be sent. The communication controller in turn adds the address and checking and synchronization information to the data block, thereby creating a frame. The transceiver now transmits the frame over the bus. In the automobile the physical interconnection structure is generally the line topology, which is very easy to manage due to the passive bus interface. On the receiver side the transceiver accepts the frame and passes it to the communication controller, which evaluates the information transmitted to it and in case of correct data reception routes the data block to the application.

This results in a hierarchical and therefore transparent communication flow. This is guaranteed by completion of the communication tasks assigned to the layers, and by the communication protocol and definition of the Physical Layer (Figure 3).

For some tasks such as bus management (including Sleep and Wake-Up functionality) or diagnostics and configuration of bus nodes, the communication functionality provided by the Data Link Layer is insufficient. By definition higher layers respectively higher communication protocols the communication functionality can be expanded.

CAN, LIN, MOST and FlexRay

Intensified competition is contributing toward more and more safety and convenience functions in the automobile. This not only results in a permanent increase in the number of electronic components in vehicles, but also a substantially greater degree of networking with rapidly escalating data

volumes, since most new automobile functions cannot do without data exchange any longer. To keep the growing complexity of automotive electronics manageable, automotive OEMs create different standards on the system, functional and communications levels. On the system or functional level, "AUTOSAR" (Automotive Open System Architecture) is expected to provide the necessary transparency in the future. Non-proprietary communication standards such as CAN, LIN, MOST and FlexRay provide greater transparency on the communications level.

CAN (Controller Area Network) is used primarily in the powertrain, chassis and convenience areas. LIN (Local Interconnected Network) serves to achieve simple and cost-effective data transmission in the sensor/actuator area. MOST (Media Oriented System Transport) is implemented in infotainment to transmit video and audio signals. Finally, FlexRay enables the most challenging communication in safety-critical distributed applications. Figure 4 shows an example of ECU networking with serial bus systems in a modern automobile. In contrast to CAN, LIN and MOST, however, FlexRay must first become established in the automobile. This fall the first FlexRay production application will hit the streets. The Munich automotive producer BMW is introducing the innovative bus system in an active suspension control system on its new X5 automobile.

CAN was developed in the early 1980s by Robert Bosch GmbH, and in 1994 it became an international standard (ISO 11898). Three of Vector's executive directors played key roles in its development, and in 1988 they founded Vector Informatik GmbH. LIN, MOST and FlexRay emanated from non-proprietary organizations: The LIN Consortium (www.lin-subbus.org), MOST Cooperation (www.mostcooperation.com) and FlexRay Group (www.flexray.com). Although they have not been officially standardized, they can be considered de-facto standards.

Reliable partner for ECU networking and data exchange

The specialists at Vector support automotive OEMs and suppliers in CAN, LIN, FlexRay and MOST networking with a universal tool chain of design and development tools as well as software components and base software for AUTOSAR ECUs. Advising, consulting services and tools for process management supplement the application areas. Its services are complemented by a broad-based training program on Vector tools, software components and serial bus systems.

For entry-level work in automotive ECU networking or data exchange the Stuttgart-based company offers the one-day seminar "Serial bus systems in the automobile". Fundamentals seminars on CAN, LIN, FlexRay and MOST are best suited as introductions to the various development activities related to automotive electronics. Additional information and schedules one can find on the Internet: www.vector-informatik.com

Outlook

Parts 2-5 of this series address the serial bus systems CAN, LIN, FlexRay and MOST in detail.



Author:

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.

Translation of a German publication
in Elektronik automotive, 7/2006



Developing automotive networking is not really difficult,

once you have learned how to do it.

Whether you are new to the field of automotive electronics or are an experienced pro, at the **VectorAcademy** you will certainly find the right workshop for your knowledge level. Your benefit: You just sign up for the learning modules that you really need.

CAN, LIN, FlexRay, MOST, AUTOSAR, Embedded Software... instead of searching for answers in the books, quiz our trainers. This is the most effective way to acquire knowledge and skills. We guide you through practice-oriented exercises. And you have the opportunity to share your experiences with colleagues from your field.

Don't settle for less. And it costs less than you might think. Get a quick overview at our web pages.

► www.vector-academy.com

► Want to try something new?
Visit our free of charge
E-Learning Portal:
www.vector-e-learning.com

Serial Bus Systems in the Automobile

Part 2:

Reliable data exchange in the automobile with CAN

The relentless pace of globalization has brought growing competitive pressure to bear on automotive OEMs and suppliers, which in turn leads to one innovative offensive after another. Electronics plays a decisive role here: Increasingly complex electronic systems provide for a high level of safety and comfort in car driving. The CAN (Controller Area Network) serial bus system makes a crucial contribution here with its specific properties. It assures reliable data exchange even under harsh environmental conditions for example. This technical article is intended to serve as an introduction to CAN technology.

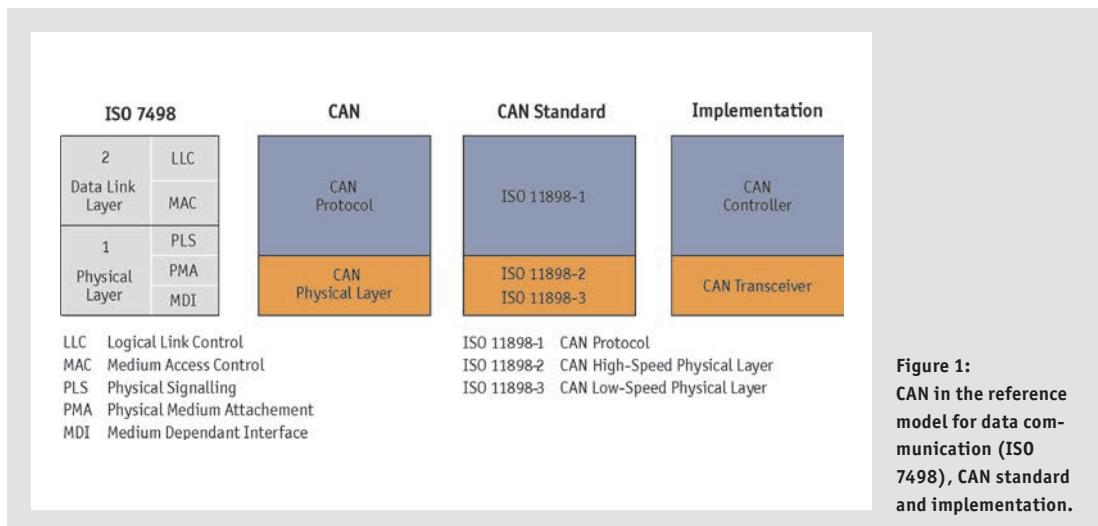
CAN standard, implementation and interface

The CAN technology developed by Bosch [1] has been standardized since 1993 and exists as ISO Standard 11898 which is organized in several parts (Figure 1). The first part contains the CAN protocol and covers the entire data link layer (framing, addressing, bus access, data assurance) and part of the physical layer (physical signaling) of the standardized reference model for data communication (ISO 7498). In the meantime a large number of cost-effective CAN controllers have become available which implement the CAN protocol in hardware.

The second part describes the CAN High-Speed physical layer, and the third part the CAN Low-Speed physical layer. These two parts cover the Physical Layer of ISO 7498 (including

physical bus interface, data rates and voltage levels). The CAN High-Speed physical layer is used primarily in powertrain and chassis applications. It is essentially implemented by the CAN High-Speed transceiver, which supports a maximum data rate of 1 MBit/s. The CAN Low-Speed transceiver with a maximum data rate of 125 KBit/s is generally used for the CAN Low-Speed physical layer that is primarily used in the body/convenience area.

Accordingly the CAN interface (Figure 2) consists of a CAN controller and a CAN transceiver. While the CAN controller handles the CAN protocol, the CAN transceiver assumes the task of physically coupling the CAN controller to the CAN bus operated in differential signal mode. Differential signal transmission enhances noise immunity and requires two



communication lines (CAN-High and CAN-Low line), which are terminated with the characteristic line impedance to avoid reflections at the ends.

Message distribution

Message addresses and message filters are used in a CAN network to organize nodes and messages. Message addresses, commonly referred to as Identifiers (ID) do not identify the CAN target nodes, rather they identify the messages themselves, so that in principle all CAN messages are available to be received by all CAN nodes (message distribution). By means of a filter each CAN node selects those CAN messages from the message stream that are relevant to it (receiver-selective system). The 11 bit wide ID permits specification of up to 2048 CAN messages in a CAN network.

Message distribution offers the following advantages:

- > Cost savings by shared use of sensors,
- > Easy implementation and synchronization of distributed processes, and above all:
- > High flexibility with regard to the configuration.

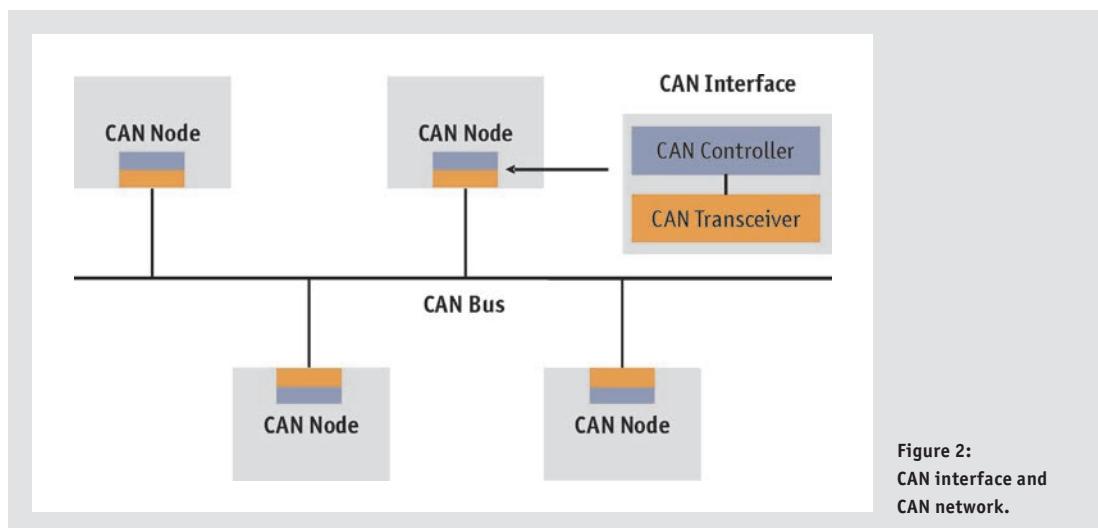
This is because omitting node addresses makes it possible to integrate other bus nodes without having to modify the

hardware or software of existing bus nodes. However this is only true if the added bus node is exclusively a receiver.

Event control

The messages that are transmitted in a CAN network and their sequence do not depend on a time progression, rather they depend on the occurrence of special events. Each CAN node is in principle authorized to access the CAN bus immediately after an event occurs. Given its relatively short message length of max. 130 bits in standard format and its high data transfer rate of up to 1 MBit/s, this method enables quick reactions to asynchronous events. This is an important precondition for real-time capable data transmission in the millisecond range (1 to 10 ms), which is primarily a requirement of powertrain and chassis applications.

Since CAN communication is not based on any time schedule the message traffic is not determined until runtime, and this implies that it carries the inherent risk of collisions. This risk increases with increasing bus load, and it calls the real-time capability of the system into question. To assure real-time data transmission in spite of random bus access, the CSMA/CA bus access method is utilized in the CAN network.



CSMA/CA bus access method

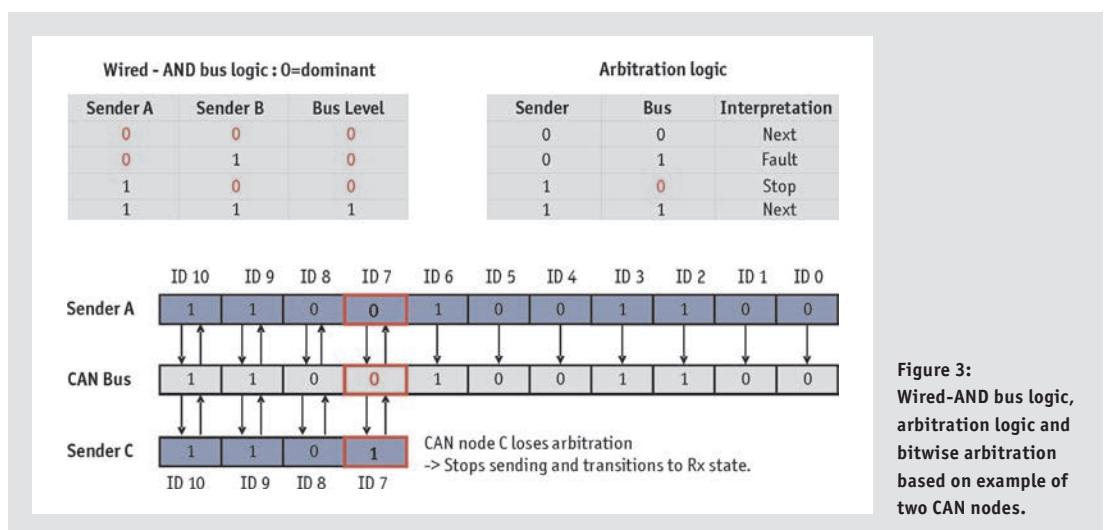
Bus access begins when a CAN node wishing to send first listens to the CAN bus (Carrier Sense – CS). If the CAN bus is available the CAN node may begin to transmit its message immediately. On the other hand, if it detects bus activity it must postpone its send request until the CAN bus is available and the currently running message transmission has been completed; in addition it must wait a duration of three bit times (ITM Intermission). An ongoing message transmission is not interrupted in this method – bus access is nondestructive.

If there are multiple CAN nodes wishing to send, bitwise arbitration (Figure 3) prevents collisions from occurring in spite of simultaneous bus access (Multiple Access – MA). In the framework of bitwise arbitration all CAN nodes wishing to send place the IDs of the CAN messages they wish to send bitwise on the bus, from the highest to the lowest significant bit. The wired-AND bus logic (0=dominant) that forms the basis of the CAN network ensures that there is always an unambiguous bus level. After adding on an ID bit, each CAN node compares the bus level with the level it sent. The arbitration logic decides whether a CAN node may continue to send or whether it must stop sending. At the end of the arbitra-

tion phase the CAN node that gets send authorization is the node transmitting the CAN message with the least significant identifier. Lower priority CAN nodes first switch to the Rx state and access the CAN bus for a renewed send attempt as soon as the bus is free again.

Not only do the bus and arbitration logic prevent collisions (Collision Avoidance – CA), they also provide priority-controlled bus access: The lower the significance of the identifier, the higher the priority of the CAN message, and this results in faster bus access. The CAN message with the smallest identifier (ID=0) will therefore be transmitted without delay.

If the bus load is not too high, this type of random, nondestructive, priority-driven bus access facilitates correct and very fast bus access. On the one hand, it should be noted that delays grow with increasing bus load, above all delays of low priority CAN messages. In the worst case a situation may arise in which CAN messages arrive too late at receivers or are suppressed entirely. On the other hand, the CSMA/CA bus access method produces a reciprocal relationship between network extension and maximum data rate. During bitwise arbitration a recessively sending CAN node must be able to



reliably detect a dominant level. The bit time interval should therefore be sized such that signal propagation times on the CAN bus are fully compensated. A length extension to a network therefore necessitates a longer bit time interval, which in turn defines a maximum usable data rate.

Data transmission

It is primarily Data frames that are responsible for data transmission in the automobile (Figure 4). While in fact Remote frames also exist for requesting data, they are hardly ever used since data transmission in the automobile is not typically request-based, rather it is primarily provided on the information generator's own initiative. The two types of frames have identical layouts; the only difference is that the data field is omitted in the Remote frame.

A basic prerequisite for transmitting Data and Remote frames is synchronism between sender and receiver. Since a clock line has been omitted for reasons of cost and effort, synchronism is achieved by signal edges and a well-defined resynchronization mechanism. Each message transmission begins with transmission of the dominant synchronization bit (SOF – Start of Frame) and this generates the first signal edge (Bus-Idle exhibits a recessive bus level). The receiver ensures synchronization over the entire transmission by evaluating each arriving signal edge and adapting its own bit timing as necessary. The bit stuffing method ensures that a complementary bit (stuff bit) appears at the latest after five homogeneous bits, thereby providing a signal edge.

Following the SOF is the ID, which may be either 11 bits (Standard-ID) or 29 bits (Extended-ID) in length. The standard format dominates in the automotive field. The extended format typically plays a role in conjunction with higher level protocols such as SAE J1939. The ID format being used is indicated by the IDE (Identifier Extension) bit. Another bit switch (RTR bit – Remote Transmission Request) indicates whether the frame is a Data or Remote frame.

The 64 bit wide data field is available for transmitting useful information, in which the exact number of useful bytes is indicated by a DLC (Data Length Code). Following the data field is the so-called CRC sequence (CRC – Cyclic Redundancy Check). The sender generates the CRC sequence based on all bits to be transmitted, a generator polynomial and a well-defined algorithm. Independent of the message filtering the same process occurs at the receiving end with the arriving bits. The two sequences are compared, and the acknowledgement is made after the recessive CRC delimiter in the Acknowledge slot (ACK slot). At the end of a Data frame, after the recessive ACK delimiter, comes the seven bit long and recessive EOF (End of Frame).

Data protection

The probability that corrupted CAN messages will remain undetected is extraordinarily low. It is estimated to be 4.7×10^{-11} [2]. Responsible for this are error detection mechanisms defined in the CAN protocol. On the receiver side, besides the message-filtering-independent CRC that is capable

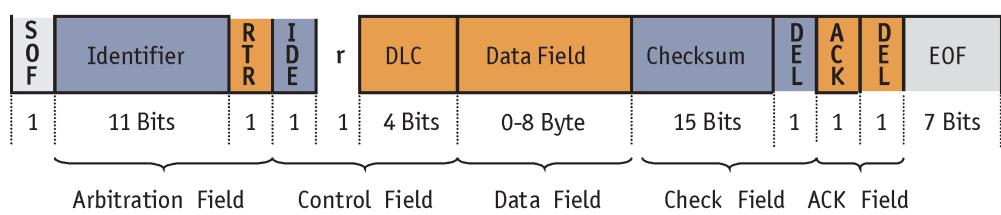
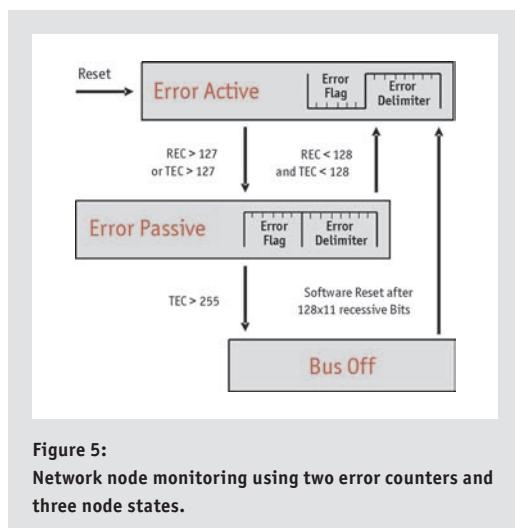


Figure 4: Structure of the Data frame.

of detecting up to five errors within a CAN message, checks are also made of the format (Form Check) and bit stuffing rule (Stuff Check). The sender performs bit monitoring and evaluates the ACK slot.

If one assumes an error rate of 10^{-3} in a CAN network, then given an annual operating time of 1000 hours, a data rate of 500 KBit/s, a mean bus load of 25 percent and a mean message length of 80 bits, statistically speaking a corrupted CAN message will remain undetected by the CAN protocol just once every 4000 years. What is understood as the error rate is the ratio of corrupted CAN messages to the number of all transmitted CAN messages.

As soon as an error detection mechanism signals a transmission error, the CAN node detecting the error terminates message transmission by placing an error flag (six dominant bits) on the CAN bus. The error flag intentionally violates the bit stuffing rule so that network-wide each CAN node perceives what until then was a local error and responds by terminating the message transmission, i.e. by appending an error flag. This method assures network-wide data consistency which is so important in distributed applications.



Error correction consists of repetition of the aborted CAN message by the same sender as soon as the CAN bus is free again (after the error delimiter and ITM). In designing the system it must be considered that the CSMA/CA bus access method does not guarantee immediate repetition. The error recovery time depends on the priority of the message and the bus load.

Node monitoring

Error signaling by error flag gives each CAN node the capability of terminating ongoing message transmissions. Since this also applies to defective CAN nodes, such nodes are capable of bringing the entire CAN communication to a standstill. To prevent this each CAN node has network node monitoring (Figure 5) that can disconnect (Bus off) a node found to be defective based on error counters and rules for controlling the error counters.

Acknowledgment of received CAN messages

In a CAN network each message transmission is acknowledged simultaneously by all receivers in the ACK slot (in-frame acknowledgement), independent of message filtering. A dominant level signifies a positive acknowledgment, and a recessive level signifies a negative acknowledgment. Since the sender places a recessive level in the ACK slot, just one positive acknowledgment is sufficient to confirm a correct message transmission. Because of this node-neutral positive acknowledgement, negatively acknowledging CAN nodes are overwritten and remain unheard. Therefore they send an error flag after the ACK delimiter.

If not a single positive acknowledgment is received, that is the ACK slot is not overwritten by any receiver, the sender detects an ACK error and aborts the ongoing message transmission by sending an error flag.

Outlook

Until just a few years ago CAN was the most sought after bus technology in the automotive industry. The relentless electrification of the vehicle has caused CAN to encounter limits. Vehicle developers are questioning the suitability of the CAN bus especially in bandwidth-intensive, real-time, critical and highly safety-critical motor vehicle applications such as

the “Lane Keeping Assistance” driver assistance system, but also in cost-sensitive convenience applications.

Therefore besides CAN two other bus technologies have become established over the course of time for use in the automobile or they are on an ideal course in that direction. We are talking about LIN and FlexRay here. LIN (Local Interconnected Network) is already being used for cost-effective networking of sensors and actuators in the convenience area. FlexRay is on the verge of being implemented in real-time and safety-critical automotive applications due to its time-triggered communication method, a data rate of up to 20 MBit/sec and the possibility of sending over two communication channels. In its first production application worldwide FlexRay will be implemented in an active suspension control system on the new BMW X5.

Reliable ECU networking and data exchange

The specialists at Vector Informatik [3] support automotive OEMs and suppliers, not only in CAN networking but also in the LIN, FlexRay and MOST bus systems. For customer projects we offer universal tool chains of design and development tools, optionally with software components and base software for AUTOSAR control modules. These products are supplemented by customer support, consulting services and tools for process management in various application areas that enable customized adaptations to specific requirements. These services are rounded out by a comprehensive training program covering Vector tools, software components and serial bus systems.

For an introduction to ECU networking or data exchange in the automobile the Stuttgart-based company offers the one-day seminar “Serial bus systems in the automobile”. Fundamentals seminars on CAN, LIN, FlexRay and MOST convey the basic knowledge needed to quickly gain familiarity with the many different development activities related to automotive electronics [4].

The first part of this series of articles [5] addressed serial bus systems in the automobile in general terms. Upcoming articles three through five will discuss the LIN, FlexRay and MOST serial bus systems. Interested readers will find supplemental and in-depth information on these topics that has al-

ready been published at the Internet site of the Vector Academy [4].

Translation of a German publication in Elektronik automotive, 8/2006

Literature and Internet links:

- [1] www.bosch.com
- [2] Unruh, J., Mathony, H.J., Kaiser, K.H.: Error Detection Analysis of Automotive Communication Protocols, SAE International Congress 1990.
- [3] www.vector-informatik.de
- [4] www.vector-academy.de
- [5] Mayer, E.: Serielle Bussysteme im Automobil – Architektur, Aufgaben und Vorteile [“Serial bus systems in the automobile – Architecture, tasks and advantages”]. Elektronik Automotive 7/2006, pp. 70ff.

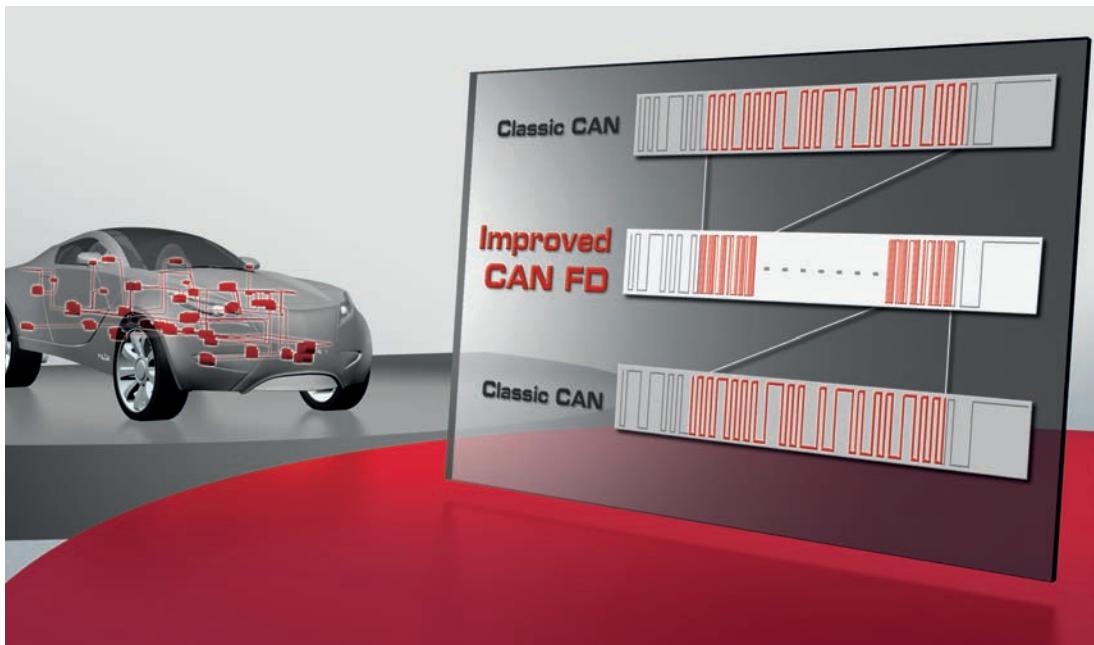


Author:

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.

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 fulfills 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 CAN FD bits to existing CAN events and 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.

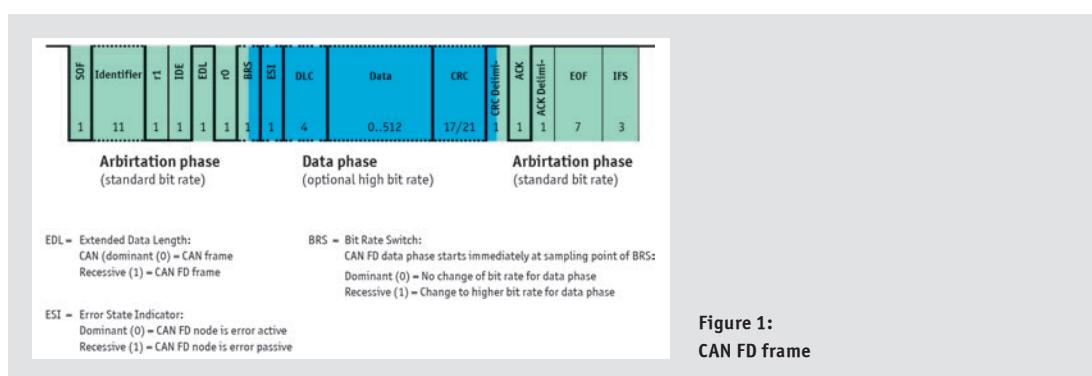


Figure 1:
CAN FD frame

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.

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.

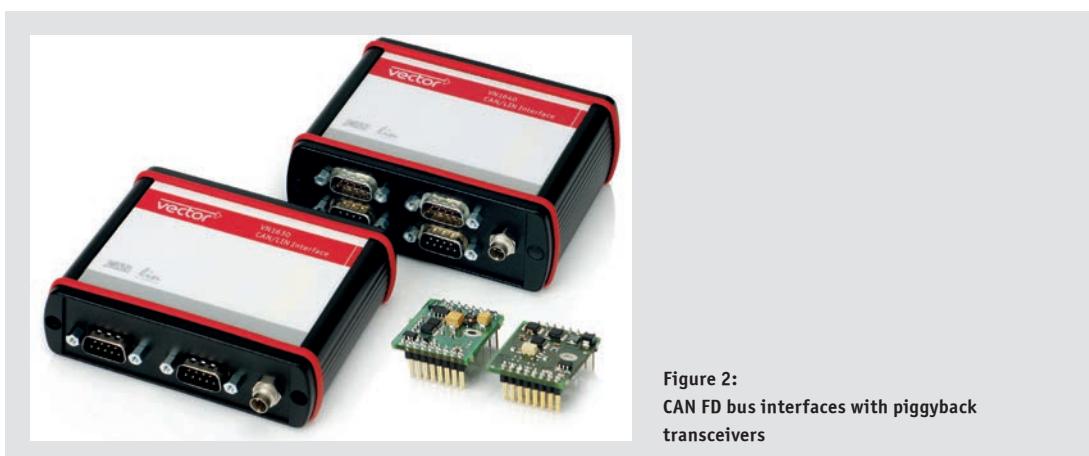


Figure 2:
CAN FD bus interfaces with piggyback transceivers

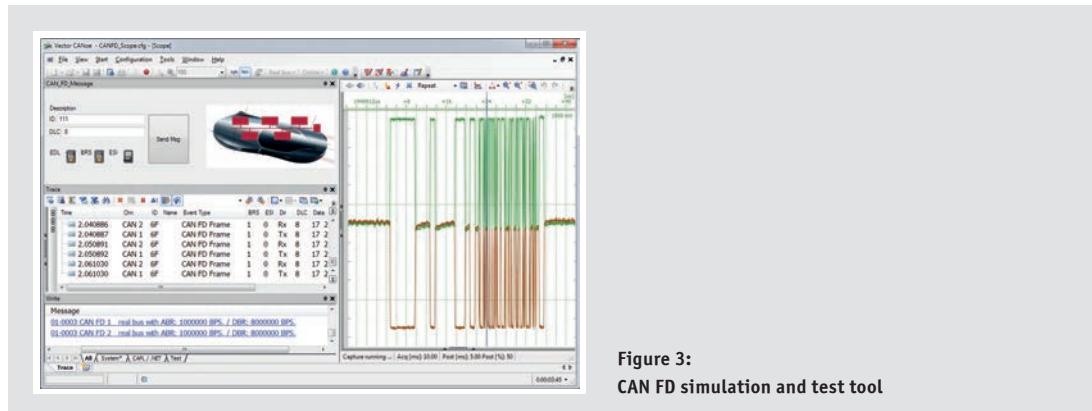


Figure 3:
CAN FD simulation and test tool

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 Hanser automotive, issue 02/2013

All Figures:

Vector Informatik GmbH

Literature:

CAN FD specification V1.0, Robert Bosch GmbH

Link:

Vector CAN Solutions: www.vector.com/can_fd

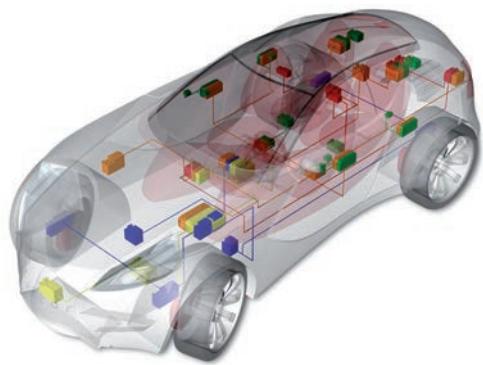


Serial Bus Systems in the Automobile

Part 3:

Simple and cost-effective data exchange in the automobile with LIN

In just a very short time the LIN bus has established itself as the technology of choice for simple and cost-effective data exchange in the automobile. Today, many automotive OEMs are relying on LIN to transmit non-critical signals in the body/convenience area. The following article points out the reasons for the victory of LIN (Local Interconnect Network) in the automobile and explains the underlying technology.



Why another data bus in the automobile?

Growing demands of car users for driving conveniences have led to wide-ranging electronification in this area of vehicle technology. This is reflected in the migration of numerous electronic components into the convenience area. For a long time it was usual practice to interconnect the continuously increasing number of sensors, actuators, stepper motors and DC motors directly to a central control module.

This trend gradually met with criticism: It led to a rapid increase in wiring costs, along with greater space requirements, increasing weight and significantly greater susceptibility to faults. In addition, growing individualization required many different wire harness and connector variants, which in turn made production, installation and maintenance considerably more difficult.

Developers quickly recognized that networking of components over a bus system would be an ideal solution in this automotive application domain too. However, since the CAN bus was not a candidate for use in the cost-sensitive sensor/actuator area, many automotive OEMs and suppliers began to develop their own sensor/actuator bus systems as early as the mid-1990s.

This gradually resulted in the creation of numerous cost-effective and simple yet proprietary sensor/actuator buses. In the year 2000 LIN arrived on the "networking market" as another serial bus system for the sensor/actuator area. This technology has prevailed on a broad front, and LIN can now be found in nearly all vehicles, typically in convenience applications such as climate control, seat adjustment, door controls and mirror adjustment.

LIN Consortium assists in breakthrough

An important reason for the quick establishment of LIN was the founding of the LIN Consortium [1], which prominent automotive OEMs and suppliers as well as semiconductor and tool manufacturers had joined. Its goal was to create a cross-OEM communication

standard for the sensor/actuator area. With definition of a simple and cost-effective Physical Layer based on ISO standard 9141, as well as a simple and lean communication protocol, the LIN Consortium has laid the foundation for success. It set the stage for implementation of simple and cost-effective bus nodes.

The LIN Consortium not only focused on LIN communication itself, but also provided a development methodology (LIN Work Flow), and this furthered acceptance of the bus system in the automotive industry significantly. LIN Work Flow makes it possible to automate the development of a LIN network (LIN Cluster), with resulting time and cost savings. The cornerstones of the development methodology are two data exchange formats used to describe the entire LIN Cluster and individual LIN nodes uniformly (Figure 1).

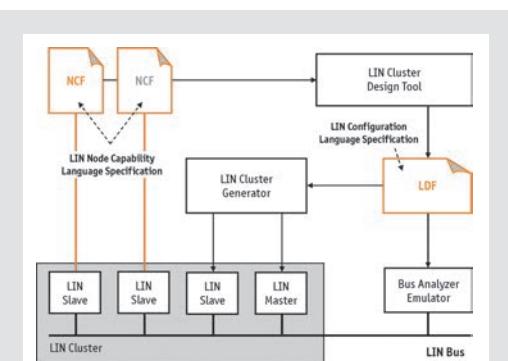


Figure 1:

LIN Work Flow: The standardized and quick path to the LIN Cluster.

Serving to describe an entire LIN Cluster are the uniform syntax (LIN Configuration Language) and the standardized LIN Description File (LDF). Defined in the LDF are all of a LIN Cluster's properties, in particular communication relationships. Generator tools utilize the LDF to generate the software components needed for LIN communication. Additionally, the LDF provides necessary information to analysis, measurement and testing tools and rest-of-bus emulators.

Similarly, the description of individual LIN nodes (LIN Slaves) is given structure by the uniform syntax of the Node Capability Language and standardized Node Capability Files (NCF). The NCF describes the performance characteristics of a LIN Slave (including frame and signal definitions, bit rates, diagnostics) and in the framework of system design it represents the foundation for automated generation of the LDF.

The two date exchange formats and the configuration process defined in the LIN specification let users implement a LIN Slave type (e.g. stepper motor) multiple times in a LIN Cluster or use a LIN Slave in different LIN Clusters (reusability of LIN Slaves).

Making just as important a contribution to the success of LIN is detailed documentation of the specification. LIN specification 2.1 [2], in existence since November 2006, defines the Physical Layer, communication protocol, LIN Work Flow, LIN API as well as diagnostics and configuration of the LIN nodes.

Single-wire communication at rates up to 20 KBit/sec

The goal of creating a low-cost communication protocol for serial data exchange in the non-safety-critical sensor/actuator area primarily affected the design of the Physical Layer. Physical signal transmission in a LIN Cluster does not involve use of the differential signal transmission familiar from CAN, rather a conventional single-

wire line is used. To assure sufficient noise immunity in spite of this method, the supply voltage and ground of the ECU electronics are used as reference voltages for the bus level. A LIN transceiver serves as the physical bus interface. A level at least 40 % below the supply voltage is interpreted by the receiver as a logical "0". Receivers interpret a level at least 60 % above the supply voltage as a logical "1".

The maximum data rate is limited to 20 KBit/sec to keep noise emissions within limits. For line lengths up to 40 meters the maximum recommended node count is 16. This takes into account the node and line capacitances as well as the maximum allowable time constant of the LIN Cluster prescribed in the LIN specification.

In terms of circuit technology a LIN Cluster is equivalent to an Open Collector circuit. A pull-up resistor ensures that the bus level remains nearly at the level of the supply voltage (High level) while the Tx transistors of all LIN nodes are inhibited. The bus level is pulled to nearly ground level (Low level) as soon as one of the Tx transistors is enabled. Accordingly, the Low state is referred to as the dominant level and the High state as the recessive level.

Master-Slave communication

Communication in a LIN Cluster is based on a Master-Slave architecture. A cluster consists of a Master node (LIN Master) and at least one Slave node (LIN Slaves). For cost reasons, explicit communication controllers are not used. Instead LIN communication is implemented by software tasks in every node, the so-called Slave tasks. The LIN Master also has a Master task that is used to coordinate cluster communication (Figure 2).

Coordination is achieved by means of periodic execution of the LIN Schedule that is organized in frame slots (Figure 3). At the begin-

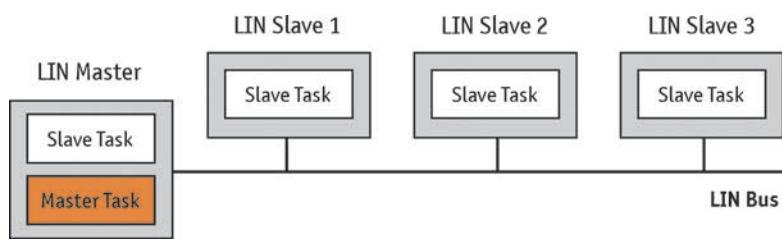


Figure 2:
Master-Slave communication architecture:
All LIN nodes have a Slave Task to participate in communication in a LIN Cluster.
One LIN node also has a Master Task for controlling the LIN communication.

ning of each frame slot the Master Task transmits a frame header with a Frame Identifier (ID), which all LIN Slaves evaluate in their Slave Task. Immediately after the frame header a LIN Slave transmits the frame response associated with the ID. The LIN Frame, consisting of the frame header and frame response, is available to be received by every LIN node (Broadcasting) due to ID-based message addressing.

Data transmission by LIN frames

Due to the lack of a communication controller, serial data transmission in a LIN Cluster is handled over the microcontroller's serial interface (Serial Communication Interface - SCI) and is performed byte-by-byte. The SCI transmits each byte with the LSB (Least Significant Bit) first, and the byte is framed by a start bit and a stop bit (SCI frame). That is, a LIN frame is composed of a combination of a number of SCI frames distributed between the frame header and frame response (Figure 4).

In transmitting the frame header the LIN Master performs two key communication tasks: It synchronizes the LIN Slaves and delegates communication by assigning a sender and one or more receivers to the frame response.

Due to cost sensitivity issues, the LIN Slaves may use on-chip resonators with a frequency tolerance of up to 14 percent. Therefore the LIN Master transmits a Sync Break first to let all LIN Slaves know that transmission of a LIN Frame is beginning. The Sync Break is made up of at least 13 consecutive dominant bits, and it elicits a SCI error from all LIN Slaves. It is terminated by the Sync Break Delimiter (at least one recessive bit). The LIN Master transmits the communication clock pulse with the subsequent Sync Byte (SCI Frame with the value 0x55).

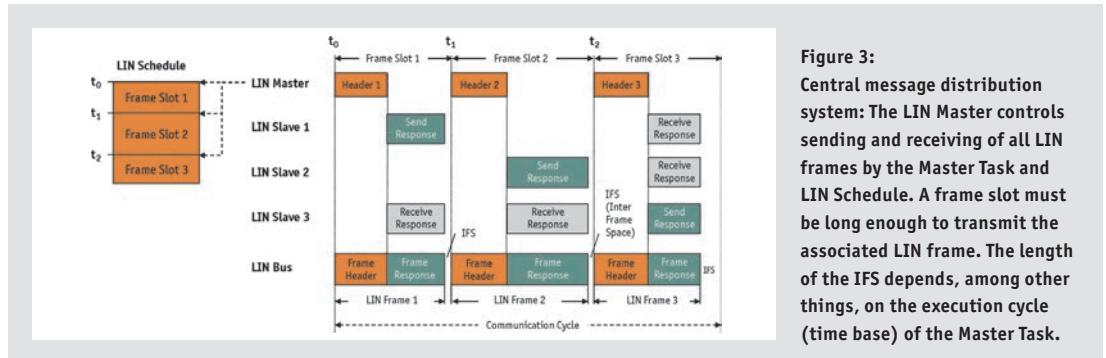
An ID serves to delegate communication; it is six bits in length and is protected by two parity bits with even parity and odd parity. The ID and two parity bits together are referred to as the PID (Protected Identifier). The first 60 IDs are available for useful data communication. The last four identifiers, ID 60 through 63, are reserved (of which ID 60 and ID 61 are used for diagnostic purposes).

The frame response is composed of up to eight data bytes and a checksum for error detection. A distinction is made between the classic and extended checksum. The classic checksum is the inverted modulo-256 sum of all data bytes. There is a transmission error if result of adding the modulo-256 sum to the arriving data bytes does not equal "0xFF". The extended checksum also considers the PID in forming the inverted modulo-256 sum.

Since the LIN Slaves are usually equipped with very simple and low-cost microcontrollers, not only are they allowed to delay transmission of the frame response a little (Response Space), but they may also insert sending pause (Interbyte Spaces) between transmissions of SCI frames. Overall, this may lengthen the frame response by 40 percent. The same applies to the frame header, primarily because there are different methods for generating the Sync Break. It is absolutely essential to consider the 40 percent time reserve in designing the LIN Schedule.

Sporadic, Event Triggered and Diagnostic Frames

The LIN specification contains provisions for making the communication cycle that is rigidly defined by the LIN Schedule more flexible and economical. The two frame types "Sporadic Frame" and "Event Triggered Frame" are provided for this purpose. In introducing these supplemental frame types it has become common practice to refer to the conventional LIN frame as an Unconditional Frame.



A Sporadic Frame is understood as an Unconditional Frame that shares the same frame slot with other Unconditional Frames. Sporadic Frames are transmitted entirely by the LIN Master as necessary, so collisions are impossible. If the LIN Master has no need for any of the frames, the associated frame slot simply remains empty.

The Event Triggered Frame was introduced to communicate sporadic changes or events on the part of the LIN Slaves. It essentially corresponds to an Unconditional Frame, but with the difference that multiple frame responses from different LIN Slaves are allocated to the frame header. The frame response that is used to complete the Event Triggered frame header depends on the needs of the related LIN Slaves. There is a need when there are new data to be transported. The frame response of the Event Triggered Frame is identified by the PID of the associated Unconditional Frame in the first byte.

In contrast to the Sporadic Frame, however, collisions cannot be excluded in the Event Triggered Frame. In case of a collision the LIN Master is responsible for transmission of all Unconditional Frames assigned to the Event Triggered Frame. It does this by activating and processing a "Collision Resolving Schedule".

Both conventional Unconditional Frames and special Diagnostic Frames are suitable for diagnosing the LIN Slaves. Unconditional Frames are used for simple signal-based diagnostics, while Diagnostic Frames are used either for user-defined diagnostics or diagnostics based on a standardized transport protocol [3] and uniform diagnostic services [4] [5].

The LIN specification defines two Diagnostic Frames: The "Master

Request Frame" and "Slave Response Frame". The Master Request Frame (ID=0x60) represents the Diagnostic Request. In this case the LIN Master transmits both the frame header and the frame response. For example, a Master Request Frame is transmitted if there is a Diagnostic Request via CAN. The Slave Response Frame (ID=0x61) corresponds to the Diagnostic Response. In this case the LIN Master transmits the header, and the specific LIN Slave transmits the response.

Management functions

The LIN specification defines Status Management and Network Management. Status Management specifies that LIN Slaves must inform the LIN Master of transmission errors that are detected such as parity or checksum errors. This is done by a "Response Error Signal" in an Unconditional Frame ("Status Frame"); however this frame does not contain any further information on the type of error. The LIN specification does not define error handling rather it leaves this task to the user.

The primary task of LIN Network Management is to regulate the transition of all Slaves in a LIN Cluster from the normal communication state (Operational) to the Sleep state and in the other direction (Figure 5). If the LIN Slaves do not detect any bus activity for four seconds, they switch from the Operational state to the Sleep state. The same condition elicits a Sleep command from the LIN Master, which is really a special Master Request Frame.

Conversely, when a LIN Slave detects a "Wake Up Signal" followed by a valid header it switches from the Sleep state via the Initialization state to the Operational state. The Wake Up Signal consists of a

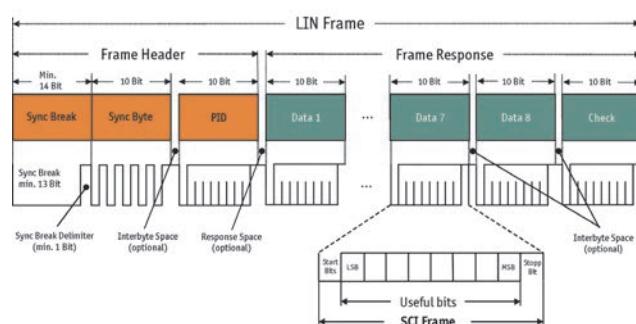


Figure 4:
Each LIN frame is composed of a frame header and a frame response. The frame header is always sent by the LIN Master. The frame response may be sent by one LIN Slave or by the LIN Master.

dominant pulse minimum 250 microseconds and maximum 5 milliseconds in length, and it may be sent by any LIN node. The LIN specification prescribes a maximum Initialization phase of 100 milliseconds, i.e. the LIN Master must begin to execute the LIN Schedule at the latest after this time span. If it remains passive the relevant LIN Slave sends another Wake Up Signal. The number of repetitions and the interval between repetitions, as well as timeouts are defined in the specification.

In case of networking issues: Quick resolution with external expertise

In networking with LIN, CAN, FlexRay and MOST, Vector Informatik [6] supports automotive OEMs and suppliers with a universal tool chain, embedded software components, base software for AUTOSAR control modules and hardware interfaces. Users of the CANoe.LIN tool for example benefit over the entire development process from practice-proven functions for model generation, simulation, functional testing, conformity testing, diagnostics and analysis. Multi-bus design and maintenance of the communication data of a networked system is supported by DaVinci Network Designer for LIN, CAN and FlexRay for example. Development, calibration and diagnostic tools for in-vehicle ECUs complete Vector's extensive product line-up. Besides consultation the Stuttgart-based company also offers a tool environment for the electronic systems development process. These services are rounded out by a comprehensive training program covering Vector tools, software components and serial bus systems.

For an entry-level introduction to serial data exchange in the automobile the Vector Academy [7] offers the one-day training course "Serial bus systems in the automobile". Training courses on CAN,

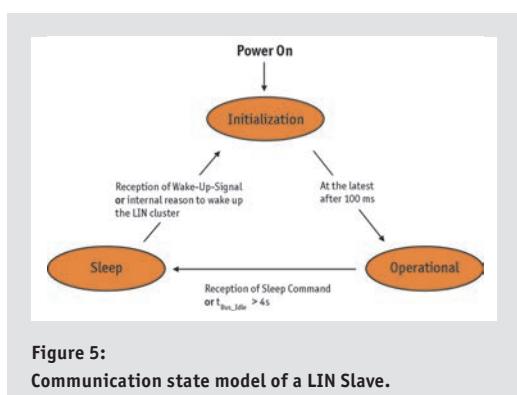
LIN, FlexRay and MOST fundamentals convey the necessary basic knowledge needed to quickly become familiar with the wide variety of development activities related to automotive electronics.

The first two parts of this series of articles addressed the topics of serial data exchange in the automobile and CAN. The next two articles will discuss the serial bus systems FlexRay and MOST.

Translation of a German publication in Elektronik automotive, 1/2007

Literature and links:

- [1] www.lin-subbus.org
- [2] LIN Specification Package Revision 2.1
- [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-informatik.com
- [7] 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.



▶▶▶

SOLUTIONS FOR **LIN**

Count on the worldwide leading solution for LIN!

Vector has the 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 and networks production-ready with calibration, stress and logging tools from Vector
- Rely on our Vector service teams for development, training and support

For successful projects, take advantage of proven tools, scalable modules and over 20 years of networking expertise at Vector!

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

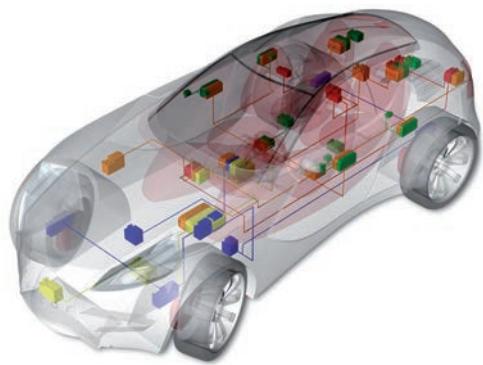


Serial Bus Systems in the Automobile

Part 4:

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 imple-

mentations 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 requirements 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 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).

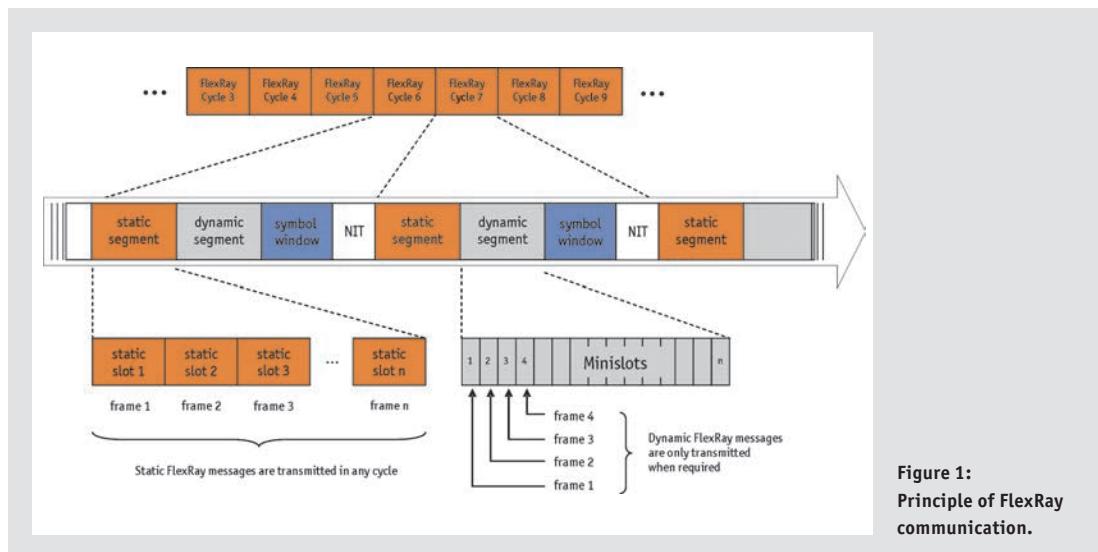


Figure 1:
Principle of FlexRay communication.

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).

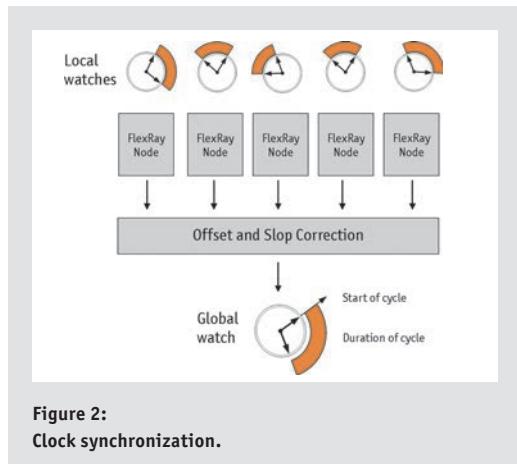


Figure 2:
Clock synchronization.

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 (minislotting) is also based on allocations and synchronous incrementing of the slot counters on the FlexRay nodes.

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

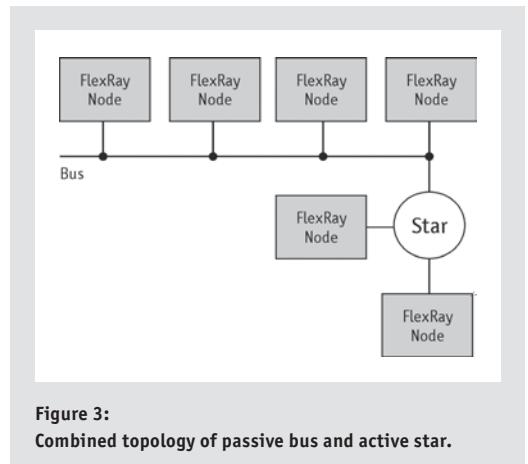


Figure 3:
Combined topology of passive bus and active star.

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.

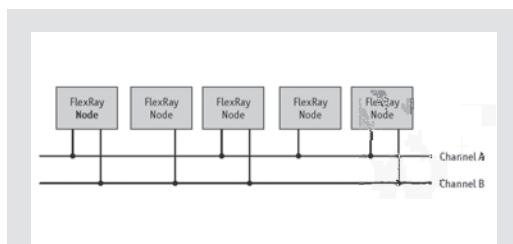


Figure 4:
Passive bus structure with two communication channels minimizes failure risk.

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].

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 multi-bus 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/tcan
- [7] www.vector-informatik.com
- [8] www.flexray.com
- [9] www.flexray-solutions.com
- [10] www.vector-academy.com

Translation of a German publication in
Elektronik automotive2/2007, 4/2012



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.

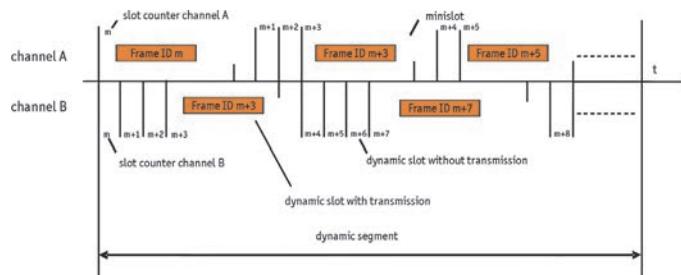


Figure 5:
Example of communication flow in the dynamic time segment.

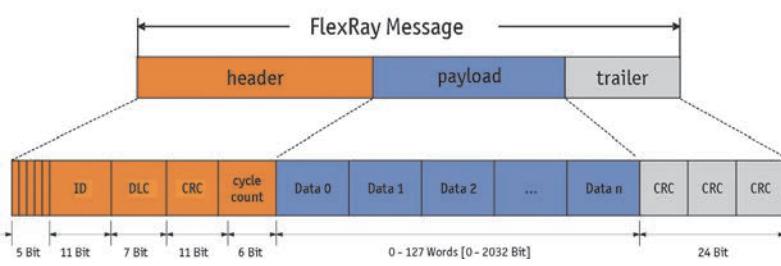


Figure 6:
Structure of the FlexRay message with header, payload and trailer.



>>> **SOLUTIONS FOR
FlexRay**

**Experience with series projects.
Choose proven FlexRay solutions.**

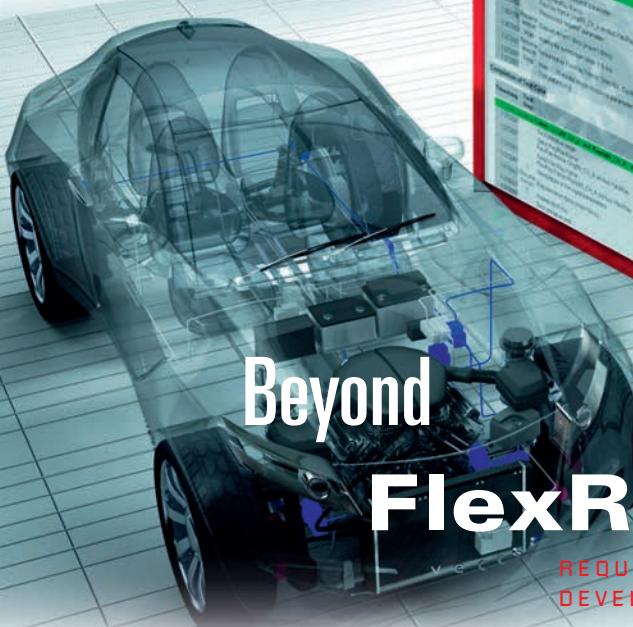
Take advantage of our extensive experience with FlexRay series projects. Use Vector's comprehensive product portfolio for your FlexRay networking:

- 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 FlexRay on the road - with series-proven tools, scalable modules and 20 years of Vector networking know-how.

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





Beyond FlexRay

REQUIREMENTS ON A MODERN DEVELOPMENT ENVIRONMENT

The FlexRay communication bus and the FIBEX database exchange format represent the current state-of-the-art of in-vehicle networking. In this context, the FlexRay bus must fulfil requirements related to remaining bus simulation, diagnostics and higher protocols, testing and AUTOSAR development methodology, over the entire development process.

These requirements are fulfilled by FlexRay development tools quickly and extensively, sometimes in completely new ways. This article discusses the requirements that need to be met for an effective development platform. It addresses special requirements of the FlexRay bus for remaining bus simulations, higher-level protocols, diagnostics, highly specialized tests and AUTOSAR development methodology.

Quick and reliable simulation of ECUs

In the development process of an ECU, it is essential to be able to operate the ECU before it is integrated in the total system. The other ECUs of the FlexRay network must therefore be simulated as communication partners. This is referred to as a remaining bus simulation. The strict time requirements associated with FlexRay are very difficult to maintain; often simulations on the FlexRay bus are non-deterministic in their execution. The resulting slot misses result in oversampling or undersampling of data on the bus. Therefore, it is often urgently recommended that these simulations be executed on a dedicated, jitter-free and deterministic (real-time capable) platform. Available solutions include expensive HiL systems and difficult

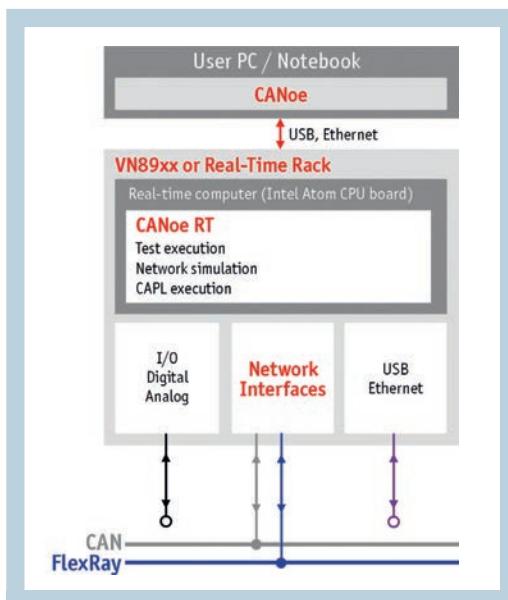


Figure 1: CANoe RT is used to simulate or test ECUs in real time.

to implement rapid prototyping platforms. Nevertheless, there is an effective and cost-saving solution specifically designed for remaining bus simulations in testing: execution of the simulation on the hardware interfaces of the bus simulation or analysis tools being used. For example, in the case of CANoe RT (**figure 1**), available on the high-end interface VN89xx (mid 2010), the user can run a simulation of the total system in real-time, or he uses a FlexRay interface from Vector to execute the simulation for one ECU. These solutions are seamlessly integrated in the simulation and test tool CANoe. This means that developers can always use the same platform and the same code, regardless of whether they are implementing a pure simulation or remaining bus simulation, or whether they are testing real ECUs on the bus.

The tools and platforms offer additional functions for a remaining bus simulation as well. For example, the send behavior of ECUs is automatically modified (Interaction Layer) based on global system states (e.g. ignition on/off). Alive counters are incremented and supplemental checksums are already computed autonomously. This allows developers to focus fully on development or testing of the applications of their ECUs. Naturally, in the case of testing it is also possible to inject errors on the communication level.

Protocols on higher layers

ECUs not only exchange signals such as vehicle speed and oil temperature, they also communicate via protocols on higher layers of the ISO OSI model – such as transport protocols, network management protocols, in flashing or calibrating. During normal operation of the ECU, such protocols are needed to coordinate bus communication. In service phases, the protocols are used to evaluate or update the ECUs. The ECUs of a specific OEM each have their own characteristics, e.g. of a transport protocol including OEM-specific modifications. Modern development tools must not only be able to analyze or represent the basic data on the FlexRay bus based on the FIBEX description but also evaluate the mentioned protocols and transmitting information through them. These functionalities are provided in the form of add-ons or plug-ins (**figure 2**).

A modern modular approach to development tools enables re-use of plug-ins in different tools and seamless integration in the tool itself. Ideally, extensions are integrated as if they were a native component of the tool. Protocol extensions provide special dialogs for setting and querying parameters of the protocols. The functions of plug-ins can be

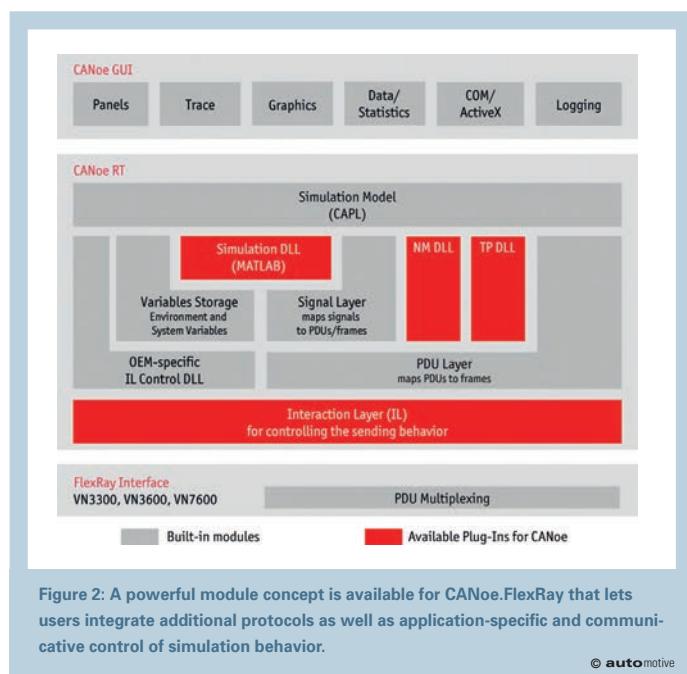


Figure 2: A powerful module concept is available for CANoe.FlexRay that lets users integrate additional protocols as well as application-specific and communicative control of simulation behavior.

© automotive

used in dedicated programs or scripts via internal interfaces. This allows developers to influence and have an access to communications by program control.

Testing by diagnostics

Access to an ECU's diagnostic functionality is very important, especially in early phases of the development process, e.g. for reading out fault memory or for flashing. Since diagnostics normally access an ECU via a CAN bus, a CAN-FlexRay gateway is needed for a FlexRay ECU. However, such a gateway is generally unavailable in early development phases. That is why CANoe's *Diagnostic Feature Set* and the measurement and calibration tool CAN-ape support direct access to the diagnostic data over the FlexRay bus (**figure 3**). Special transport protocols for FlexRay (AUTOSAR, ISO and OEM-specific variants) are supported in this context.

After loading the diagnostic description databases in CAN-decila or ODX format, diagnostic dialogs and functions are available in CANoe and CANape. For example, this allows reading out the fault memory and displaying it including symbolic interpretation of the trouble codes. In addition, the diagnostics console is used to symbolically send and evaluate all diagnostic requests, including their responses and parameters. Diagnostics communication is displayed in its domain-specific notation in traces. A dedicated diagnostics-API is also available, which enables easy access in programming, testing and tool control.

A simpler way to create ECU tests

Frequently used standard tests often run according to the same scheme. Therefore, a test tool should support fre-

quently recurring test patterns. For CANoe, it is very easy to create sequential tests and test steps using the *Test Automation Editor*. Prepared or user-specific libraries are linked for this purpose; they allow the test engineer to simply select and parameterize the test cases. Optionally, complicated or high-effort tests can be specified algorithmically. This makes it possible to execute loops or branching based on the results of prior tests. Static and algorithmic tests may be combined.

Along with stimulation tests, the bus communication can simultaneously be monitored regarding the FlexRay protocol. This includes detection of null frames and erroneous frames as well as monitoring of the frame period, response time behavior etc. Monitoring activities are performed by so-called observational checks, which are already contained in CANoe and only need to be activated and parameterized before use.

For Hardware-in-the-Loop or environment simulations, it is frequently necessary to connect the ECU to its real sensors and actuators. The digital and analogue input and output variables must be provided or read-in. The "VT System" by Vector was developed for this purpose and for open-loop simulations and tests. It provides automotive-capable I/O ports that can be integrated easily in tests. Unlike general digital or analog I/O cards, the system can selectively switch between connection of an original load or original sensor or else a suitable simulation of the input or output. At the same time, large voltage ranges and currents can be handled by suitable signal conditioning. This also makes it possible to simulate faulty behavior of a sensor or actuator or a short circuit.

Testing AUTOSAR-functionality in FlexRay systems

In the AUTOSAR design methodology, communication as a basic service for the application is transparent. In the framework of the overall system, communication is defined in the "AUTOSAR System Description". If an AUTOSAR-ECU needs to be tested simply and quickly, the test tool has to know the communication description for the ECU. It is more user-friendly to directly read in the "AUTOSAR System Description" rather than the FIBEX databases. This will be supported by CANoe.FlexRay.

Key concepts in the AUTOSAR context are: "Software Components" (SWCs), "Runtime Environment" (RTE) and "Virtual Functional Bus" (VFB). Even if the ECU does not physically exist yet, it must still be possible to test its "Software Components". For this purpose, the "DaVinci Component Tester" is able to integrate the real ECUs code of a SWC in the test environment. CANoe provides the RTE, VFB and basic software. The test is executed with the well-known functions of CANoe, including the automatic creation of a test report.

Outlook

In future, the number of AUTOSAR ECUs and the number of included SWCs will grow rapidly. These complex

© Carl Hanser Verlag, München 2009. All rights including reprinting, photographic reproduction and translation reserved by the publishers.

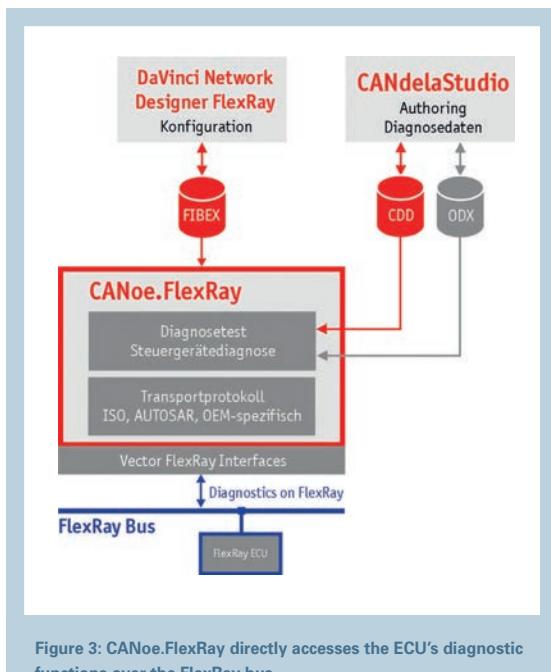


Figure 3: CANoe.FlexRay directly accesses the ECU's diagnostic functions over the FlexRay bus.

© auto:motive

systems require a powerful analyzing capability to test the internal communication between the SWCs. Therefore, the test tool needs access to the communication on the VFB. This is usually done via XCP-on-FlexRay.

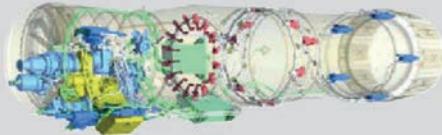
But especially extensive testing needs a higher performance. Therefore, the coupling of the control unit with the test environment CANoe is done via a JTAG or Nexus debug port. Only in this way parameters of the VFB, RTE or basic software (BSW) of a real AUTOSAR ECU can be read-in and calibrated quickly. This will be done by connecting the VFB of the real ECU to the test environment CANoe. Thus, the testing and analyzing at the ECU-internal interfaces of the SWCs can be done quite similarly in future, as today at the external control connections of an ECU. If the requirements described are realized in a sophisticated way in the development tools and in the implementation of FlexRay systems, FlexRay networks can be developed more efficiently than CAN networks.



Dr. rer. nat. Carsten Böke is Senior Software Development Engineer for the *Tools for Networks and Distributed Systems* product line at Vector Informatik GmbH.



Vector Informatik GmbH
www.vector.com



Case Study

Validation of data bus concepts for the “more electric aircraft” engine controller



The Customer

Hispano-Suiza is a Centre of Excellence of the SAFRAN Group in the areas of engine control and monitoring systems, electrical and electronic equipment. Hispano-Suiza is the global market leader in power transmissions for commercial jets. FADEC International LLC, a joint-venture formed by BAE Systems and SAFRAN, is now global market leader in 100+ passenger civil aircraft engine controllers.

The Challenge

Integrated and optimized systems for tomorrow's advanced design “more electric engine”

Requirements for the “more electric engine” controller needed to be analyzed with regard to the data bus properties. For evaluating CAN and FlexRay, a rapid prototyping and analysis tool chain should be implemented to provide information on timing, reliability, and bandwidth usage under varying conditions.

The Solution

Simulation, analysis and testing with one tool

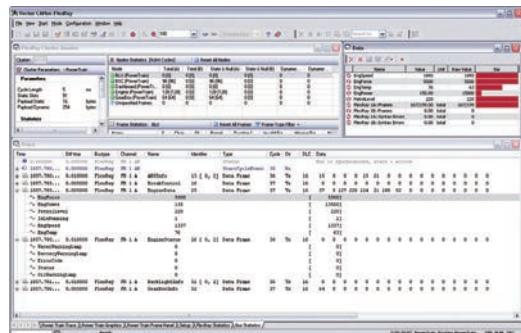
CANoe.FlexRay is used to design a CAN and FlexRay communication system and perform real-time simulations. Analysis with CANoe provides all the timing characteristics of the system. In final evaluation, the same analysis is applied to the real system.

The Advantages

Fast and reliable development and analysis of FlexRay-CAN systems

CANoe.FlexRay supports Hispano-Suiza engineers in developing the “more electric” engine controller:

- ▶ Simulation setup of several alternative communication designs and flexible parameterization facilitates simulation, testing, and analysis of FlexRay systems
- ▶ Functional and integration testing of electronic units
- ▶ Network integration testing
- ▶ CANoe.FlexRay in gateway operation: Simultaneous stimulation and analysis of CAN and FlexRay networks
- ▶ Cycle multiplexing, in-cycle multiplexing, signal groups, and sub-frames: well-organized display in analysis windows, flexible use in simulation



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



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

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.

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

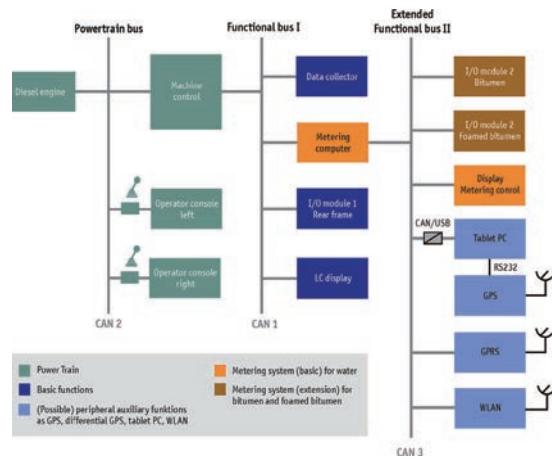


Fig. 2: The electronics concept reflects the modular layout of the machines



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 de-

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 re-

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-syn-

chronous 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

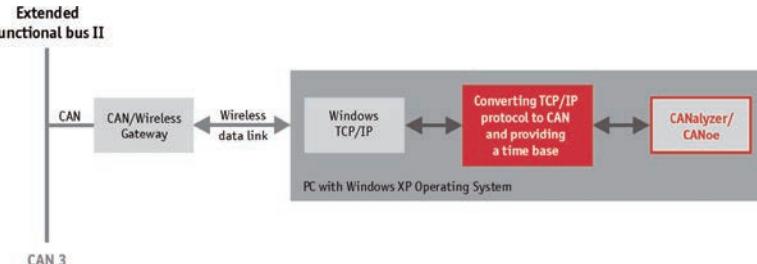


Fig. 4: The extended WLAN tunnels the CAN messages, including time stamp, via a TCP/IP interface, thereby enabling time-conformant representation of the data

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.

requirement in the analysis of multi-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

achieved a 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.

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

The Vector solution for Automotive Ethernet: Tools, basic software and services

Ethernet in Motor Vehicles

The Ethernet technology is already used in motor vehicles for diagnostics over DoIP and the PowerLine communication with electric charging stations. Currently new applications and transport layers are added, e.g. camera-based assistance functions via BroadR-Reach®, etc. Automotive OEMs and suppliers are confronted with many different challenges in this field. Vector can support you with professional tools, basic software and services.

Overview of Advantages

- > Practice-proven products and competent services based on Vector's many years of experience in automotive networking
- > Support of the new technology BroadR-Reach®
- > Universal multibus tool chain makes it easy to integrate new technologies in existing vehicle architectures
- > Practice-oriented solutions through participation in standardization committees such as OPEN Alliance SIG and ISO 15118 as well as intensive cooperation with users

Tools

- > **CANoe.IP** – Cross-network remaining bus simulation and testing of ECUs and entire networks with one tool. The .IP option contains functions for analyzing and stimulating Ethernet, BroadR-Reach®, SOME/IP, AVB and more.
- > **CANalyzer.IP** – Analysis of the whole vehicle communication beyond bus and protocol boundaries. The .IP option contains functions for analyzing and stimulating Ethernet, BroadR-Reach®, SOME/IP, AVB and more.
- > **VT System** – Modular test environment for efficient ECU and functional tests. The VT System, together with CANoe, forms a powerful and flexible testing solution.
- > **CANape und VX1000** – Measurement, calibration and flashing over XCP with high data throughput resulting in minimal runtime effects on the ECU
- > **Indigo** und **CANoe.DiVa** – Diagnostics and automated validation of the diagnostic protocol, also via DoIP.
- > **vFlash** – Very easy to use tool for flashing one or more ECUs, also via DoIP.

AUTOSAR Basic Software

- MICROSAR ETH** – Ethernet-based communication:
- > Fast and parallel diagnostics and re-programming of ECUs according to ISO 13400-2 (DoIP)
- > Service-oriented communication using Service Discovery and SOME/IP
- > PDU- and signal-based communication
- > Measurement and calibration via XCP-on-Ethernet

- MICROSAR V2G** – Vehicle-to-Grid communication:

- > Intelligent charging of electric and hybrid vehicles using Smart Charge Communication (SCC) according to ISO 15118 and DIN 70121
- > Communication via Internet mechanisms and -protocols, e.g. HTTP

- MICROSAR AVB** – Audio/Video Bridging:

- > Audio and video streaming
- > Selection of the most accurate timer
- > Displaying a synchronous system clock
- > Guaranteed Latency

Network Interface

- VN5610** – Network interface for BroadR-Reach®, Ethernet and CAN. It can be used as a media converter and as an interface.

Universal ECU

- VC121** - Universal ECU that can be used for gateways and I/O control functions. It is ideal for quickly developing prototypes and for use in small scale production runs. It supports the Ethernet/BroadR-Reach®, CAN, FlexRay and LIN bus systems.

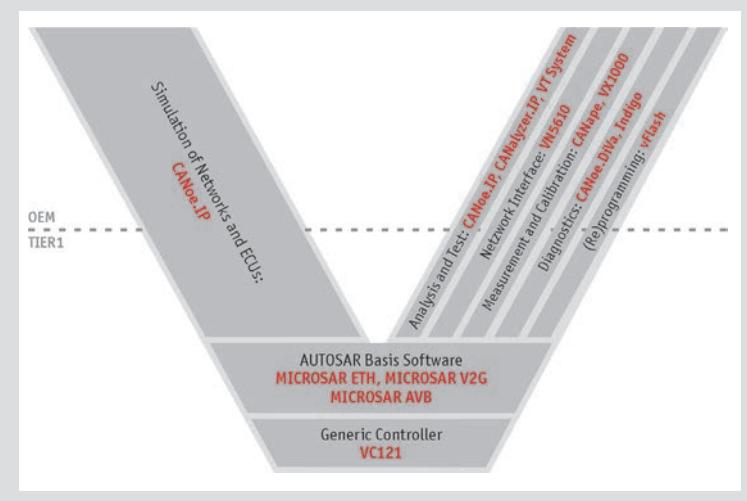
Services

Vector offers you services with know-how:

- > Seminars and Workshops
- > Technical consultation
- > Project support

The knowledge of our experienced employees is your advantage for efficient and customer-specific Automotive Ethernet solutions.

All about the Vector Automotive Ethernet Solution: www.vector.com/ae



The Automotive Ethernet solution from Vector covers the entire development process for networks and ECUs.

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 time-synchronously

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 indispensable 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

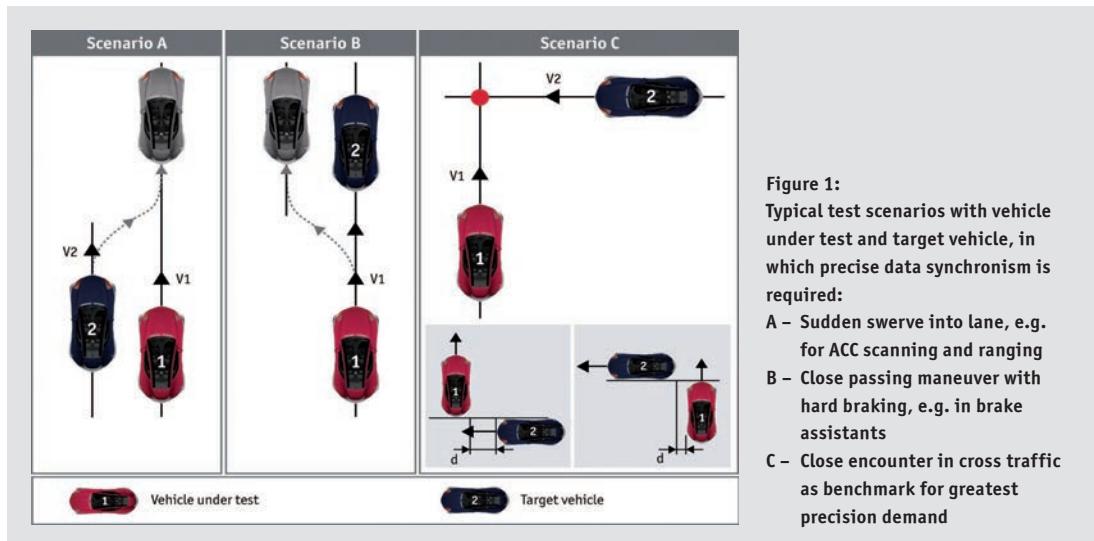
Validation of assistance systems involves an enormous amount of technical effort. Fractions of seconds are involved in deciding the success or failure of actions and reactions of the vehicles, and driving maneuvers must be absolutely reproducible. So, it is not only out of concern for safety that Daimler 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.

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 different 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 .IP 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.IP 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 lets 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.IP 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.IP or CANalyzer.IP is used. For a general explanation and overview of the principles of this method, see [4].

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



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

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 and 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**).

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 lane-changing vehicle gets the information that it must not pull out into the neighboring lane.

Developers will have to become accustomed to the idea of logging 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

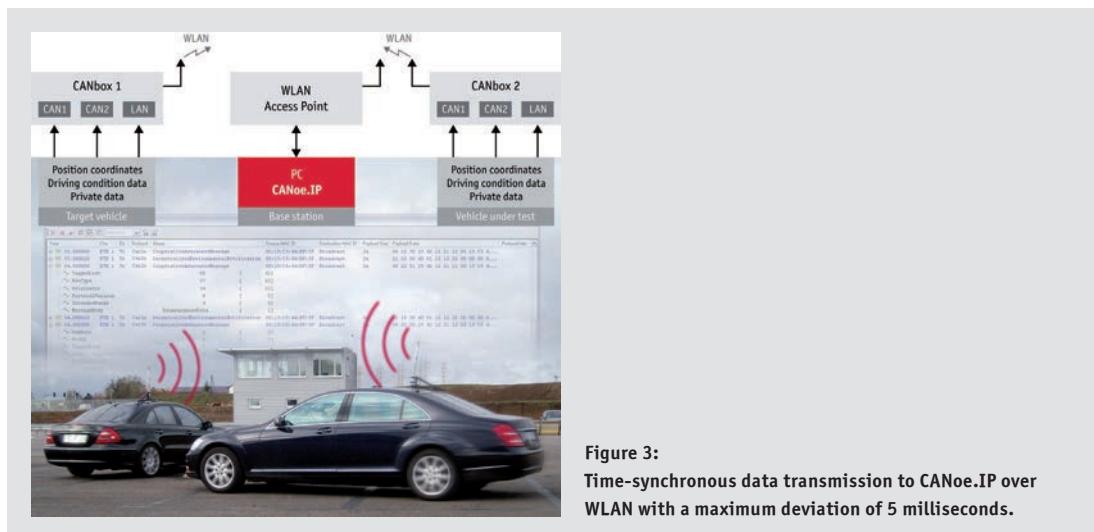


Image credits:

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 Assistentensystemen [“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] Wielbel, 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_IIEEE1588_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.IP: www.vector.com/vi_canoe_ip_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 robot-controlled 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.



Figure 4:

CANoe.IP simulates a traffic scenario based on the example of a received skidding warning.

Solutions for CAN / CAN FD

The Vector Solution for CAN Networks: Tools, Basic Software and Services

Classic CAN and improved CAN FD

- > **CAN (Controller Area Network)** is a message oriented multi-master protocol for quick serial data exchange between electronic control units in automotive engineering and factory automation.
- > **CAN FD (CAN with flexible data rate)** is an enhancement of the CAN protocol developed by Bosch company. The main differences to CAN are the extended payload from 8 up to 64 bytes, and the ability to send the payload with higher data rates. In this way, the requirements for higher bandwidth networks in the automotive industry are fulfilled, while profiting from the experiences in CAN development.

Overview of Advantages

- > Practice-proven CAN products and competent services based on Vector's many years of experience in automotive networking
- > Support of the improved CAN protocol CAN FD
- > Universal multibus tool chain makes it easy to integrate new technologies in existing vehicle architectures

Tools

- > **CANoe** – Remaining bus simulation and ECU or network testing with a single tool for CAN and CAN FD
- > **CANalyzer** – Analyze the entire vehicle communication across bus and protocol boundaries for CAN and CAN FD
- > **VT System** – Modular test environment for efficient ECU and functional tests. CANoe is the related test automation software.
- > Option **SCOPE** – Integrated oscilloscope solution for CANoe and CANalyzer. Supports CAN, CAN FD and LIN bus systems.
- > **CANape** und **VX1000** – Measurement, calibration and flashing over XCP with high data throughput resulting in minimal runtime effects on the ECU.*
- > **Indigo** – Diagnostics quick and easy with DoIP and OBD support.
- > **CANoe.DiVa** – Automated testing of diagnostic protocol implementation and integration in ECUs
- > **vFlash** – Very easy to use tool for flashing one or more ECUs
- > **Network Designer CAN/LIN/FlexRay – Tools** for the design of network architectures

Embedded Software

- > **CANbedded** – Configurable software components for CAN communication. Used by numerous OEMs worldwide.
- > **MICROSAR CAN** – Contains basic software modules for CAN communication that are defined in the AUTOSAR architecture
- > **Flash Bootloader** - Used to reprogram ECUs quickly, efficiently and reliably
- > Support of CAN FD can be implemented in CANbedded and MICROSAR CAN at customer request.

Network Interfaces

Available for CAN FD with reconfigurable FPGA hardware architecture:

- > **VN 1600** – Flexible network interfaces for CAN, CAN FD, LIN, K-Line, J1708 and I/O
- > **VN8900** – Modular network interface with integrated real-time PC for CAN, CAN FD*, LIN, FlexRay and J1708
- > Bus transceiver for CAN, LIN, FlexRay and J1708

Universal ECU

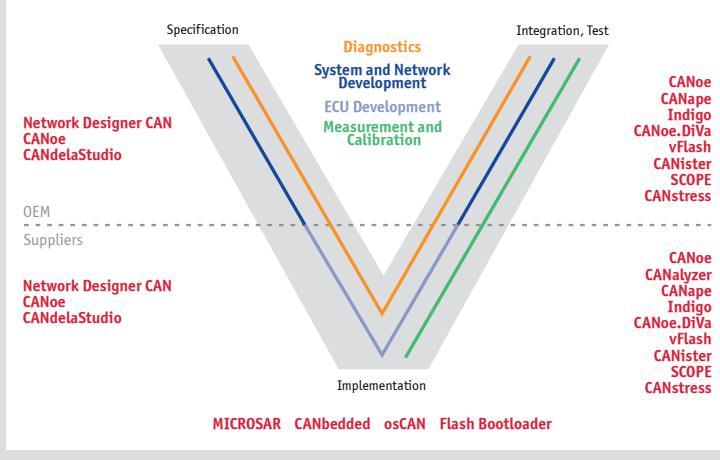
VC121 - Universal ECU that can be used for gateways and I/O control functions. It is perfect for quickly developing prototypes and for use in small scale production runs. It supports the Ethernet/BroadR-Reach®, CAN, FlexRay and LIN bus systems.

Engineering Services

Vector offers you expert advice services:

- > Seminars and workshops
- > Technical consultation
- > Project support

The knowledge of our experienced employees is your advantage for efficient and customer-specific CAN solutions.

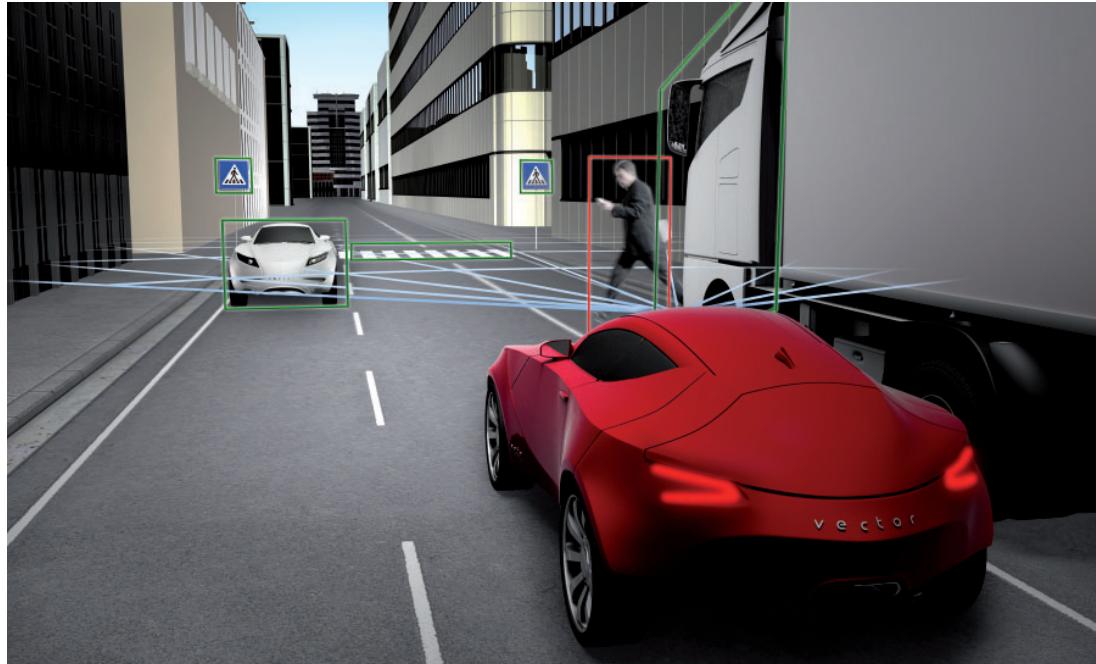


The CAN solution from Vector covers the entire development process for networks and ECUs.

*Please check for CAN FD availability

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

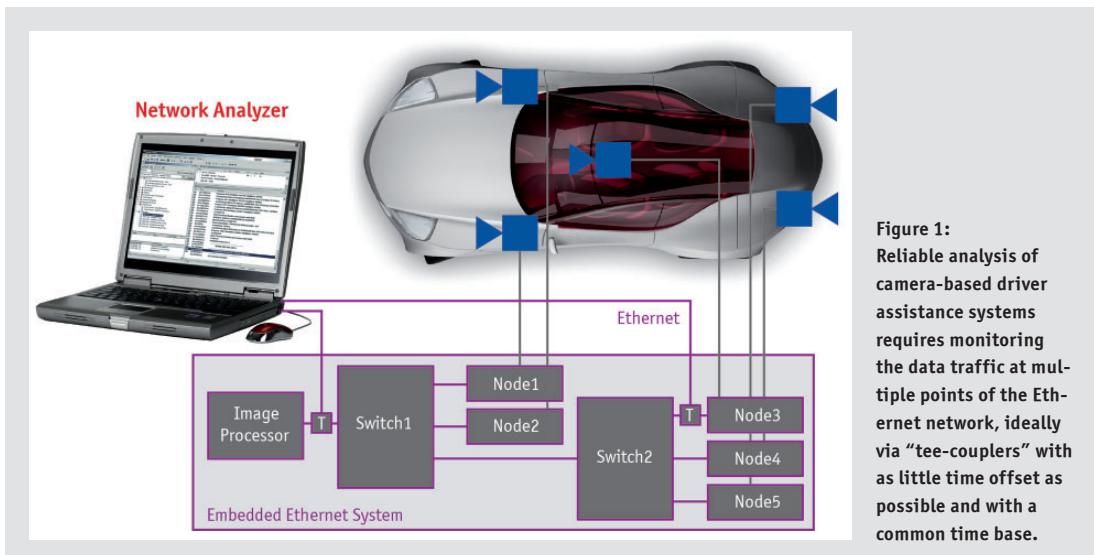
A camera-based driver assistance system at BMW will be the first production implementation in the motor vehicle to utilize IP and Ethernet as the system network in the vehicle [1]. OEMs and

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 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.

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

	Diagnostic Access	Audio/Video Time Sync	Smart Grid	Control Communication	Service Discovery	Address Configuration	Address Resolution, Signalling, etc.
7 Application	DoIP	AVB IEEE 1722 802.1AS	ISO 15118 Part1 (2)	SOME/IP	Bonjour	DHCPv6 DHCP	ICMPv6 ICMP
6 Presentation							NDP ARP
5 Session							
4 Transport	TCP/IP UDP			TCP/IP UDP	UDP		
3 Network	IPv4/ IPv6					IPv4/ IPv6	
2 Data Link	IEEE Ethernet MAC + VLAN (802.1Q)		ISO 15118 Part 3		IEEE Ethernet MAC + VLAN (802.1Q)		
1 Physical	100-BASE-Tx	BroadR-Reach			Automotive Ethernet Physical Layer (e.g. BroadR-Reach)		

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.

(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 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.

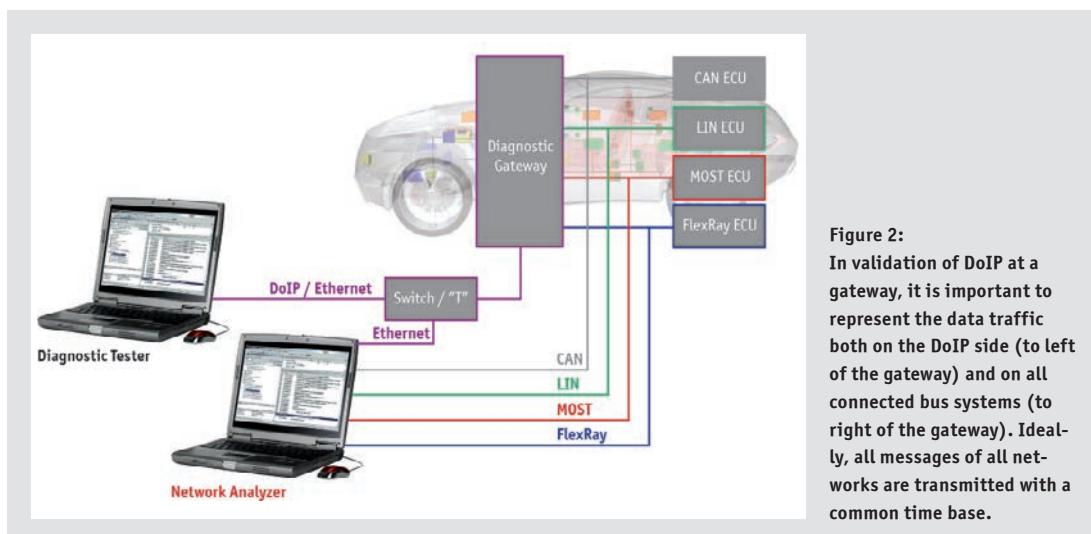
Validation of IP production projects

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 IP, 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.IP can help to reduce development time, allowing valuable resources to be used more effectively on the application side (**Figure 3**). CANoe.IP 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.IP 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).

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., Mattheus, 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.

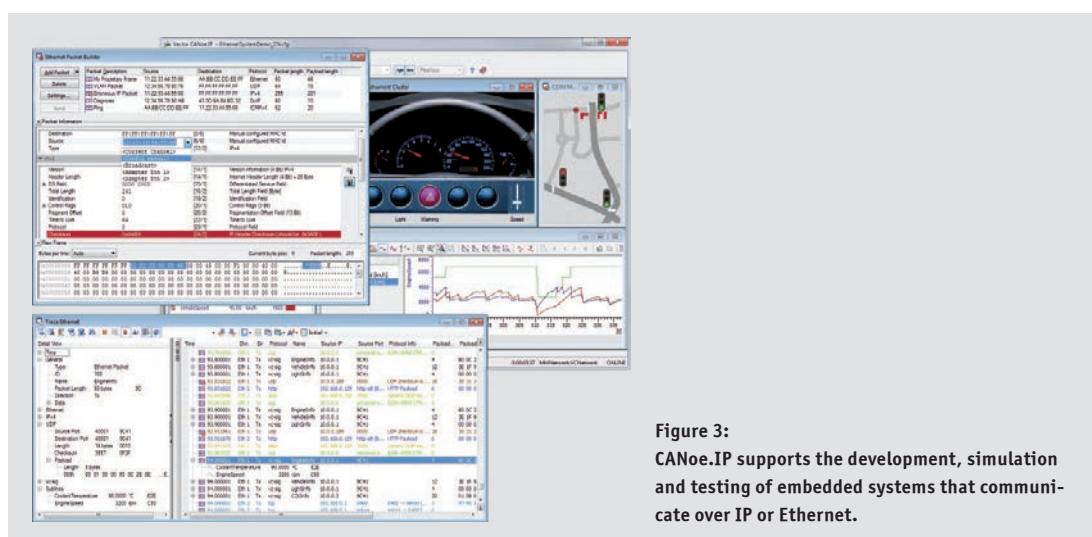


Figure 3:
CANoe.IP supports the development, simulation and testing of embedded systems that communicate over IP or Ethernet.

- [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.

Links:

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

Product information CANoe.IP:
www.vector.com/vi_canoen_ip_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, 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. Previously, he worked in various industries as development engineer, project leader and product manager in the test tools area for several network technologies.

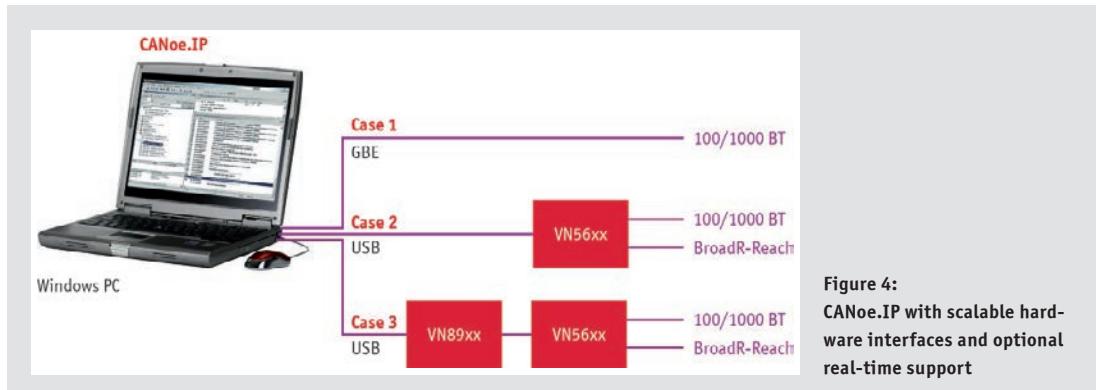


Figure 4:
CANoe.IP with scalable hardware interfaces and optional real-time support

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).

Use Case	Audio/Video Time Sync	Address Configuration	Service Discovery	Service Control	Flash Update	Helper Protocols
7 Application	API			API		
6 Presentation						
5 Session	IEEE 1722 (AVTP)	IEEE 802.1AS (gPTP)	DHCP	SOME/IP	DoIP TFTP	
4 Transport				UDP/TCP		
3 Network	AV-transport	Time Sync		IPv6/ IPv4	ICMPv6, NDP ICMP, ARP	
2 Data Link			Ethernet MAC + VLAN (802.1Q)			
1 Physical			Ethernet Physical Layer (Ethernet, OPEN Alliance BroadR-Reach, Reduced twisted-pair Gigabit Ethernet)			
 = Audio Video Bridging (AVB)						

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].

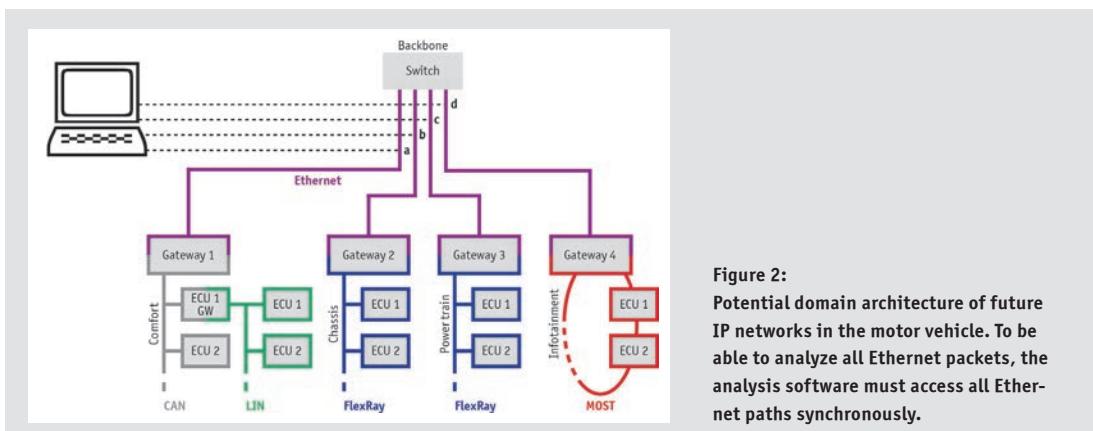


Figure 2:
Potential domain architecture of future IP networks in the motor vehicle. To be able to analyze all Ethernet packets, the analysis software must access all Ethernet paths synchronously.

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 pre-configured 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.IP, 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

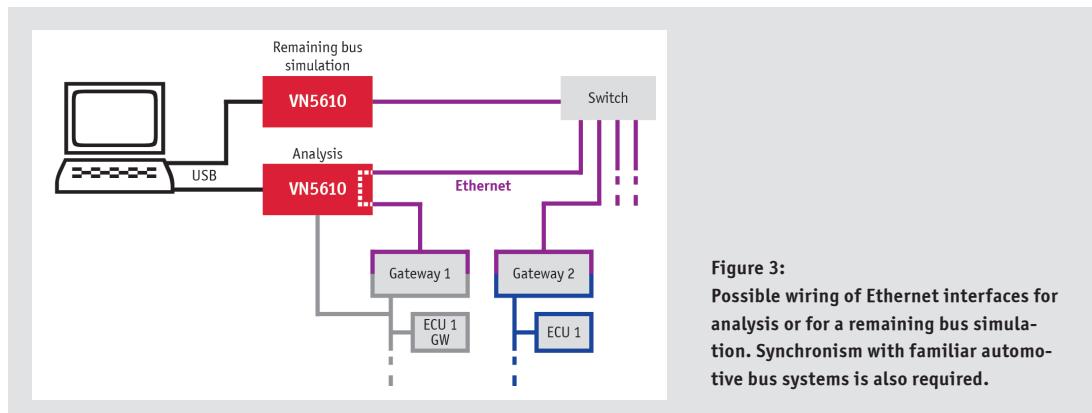


Figure 3:
Possible wiring of Ethernet interfaces for analysis or for a remaining bus simulation. Synchronization with familiar automotive bus systems is also required.

provides a way to test ECU behavior in response to specific error cases such as protocol errors.

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.

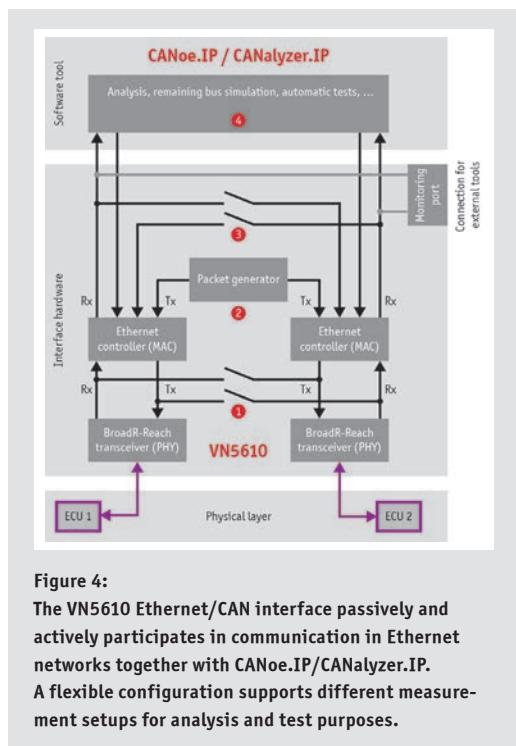
The use of high-performance analysis tools from the field of office communications together with external media converters is often overly simplified. The noted requirements can only be implemented by specialized hardware that is closely intermeshed with the analysis and simulation software. One combination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



Figures:
Vector Informatik

>> **Contact information of the Vector Group:**
www.vector.com/contact

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

Information on IP products: www.vector.com/ip



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,**

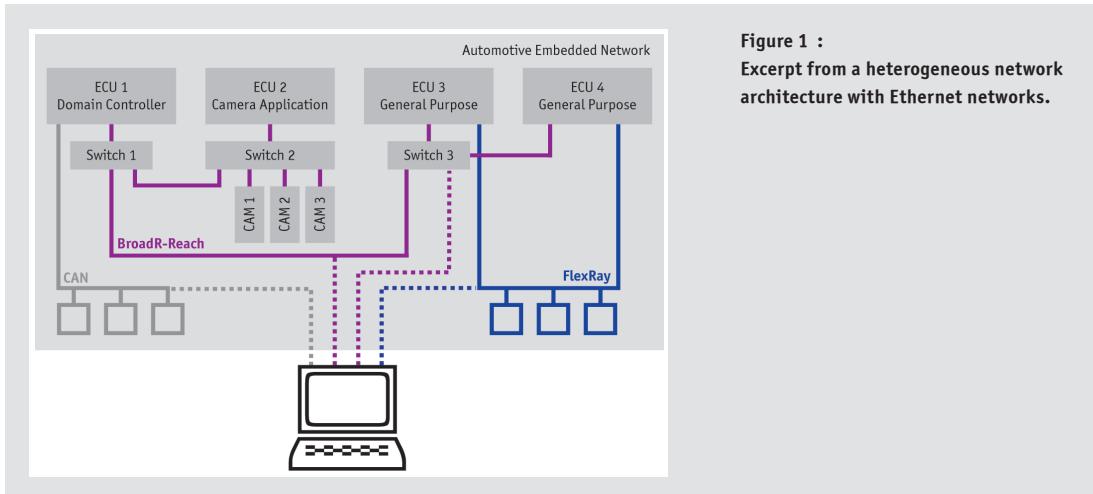


Figure 1 :
Excerpt from a heterogeneous network architecture with Ethernet networks.

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.

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

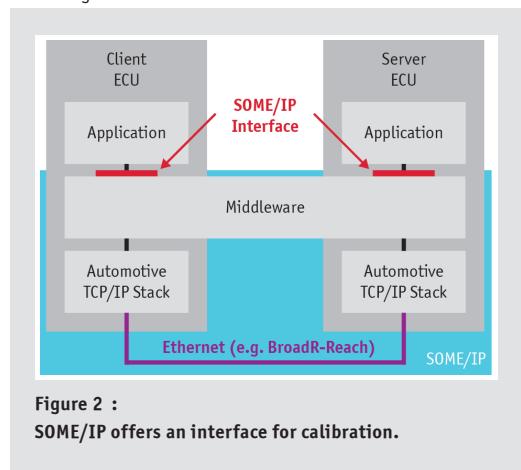


Figure 2 :
SOME/IP offers an interface for calibration.

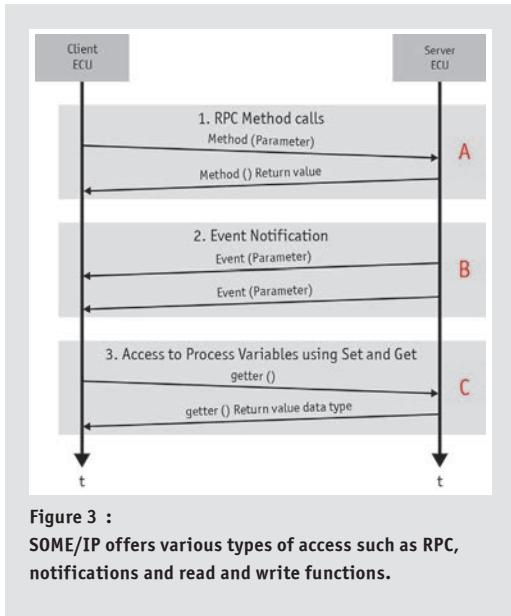


Figure 3 :
SOME/IP offers various types of access such as RPC, notifications and read and write functions.

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 be used for supporting heterogeneous networks be synchronized with all commonly used bus systems and IP networks?

The development tool CANoe.IP 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

All Figures: 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

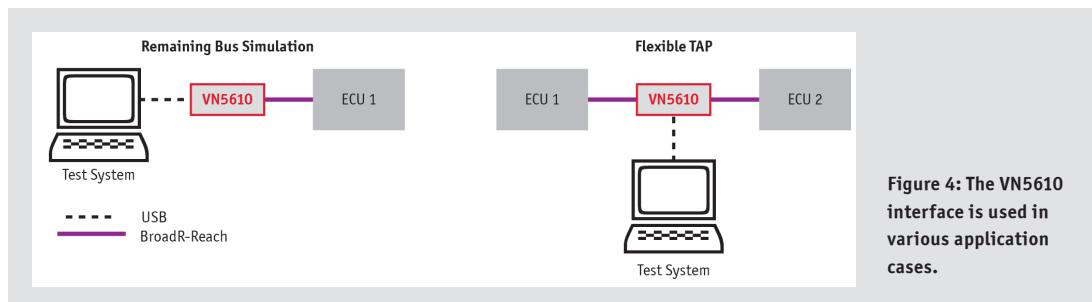


Figure 4: The VN5610 interface is used in various application cases.



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

>> Contact information for all locations of the Vector Group:
www.vector.com/contact

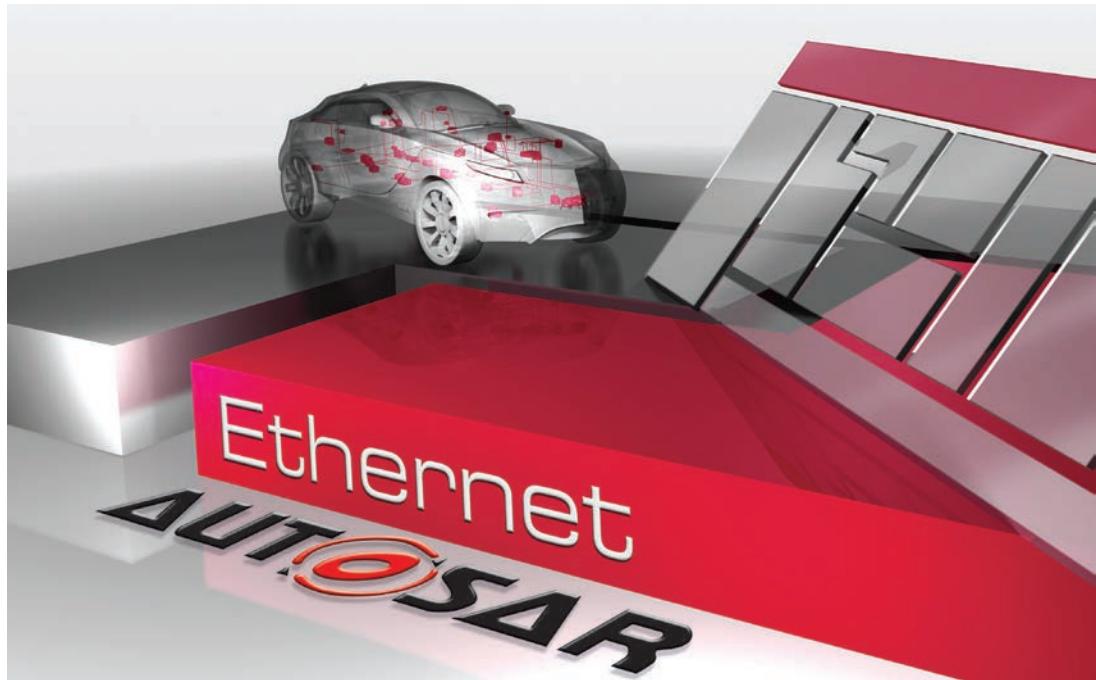


Matthias Schwedt (Dipl.-Ing. (FH))
is FPGA developer for Ethernet, FlexRay and MOST150 network interfaces as well as overall Project Leader for the VN56xx Ethernet network interface product line at Vector Informatik GmbH.



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

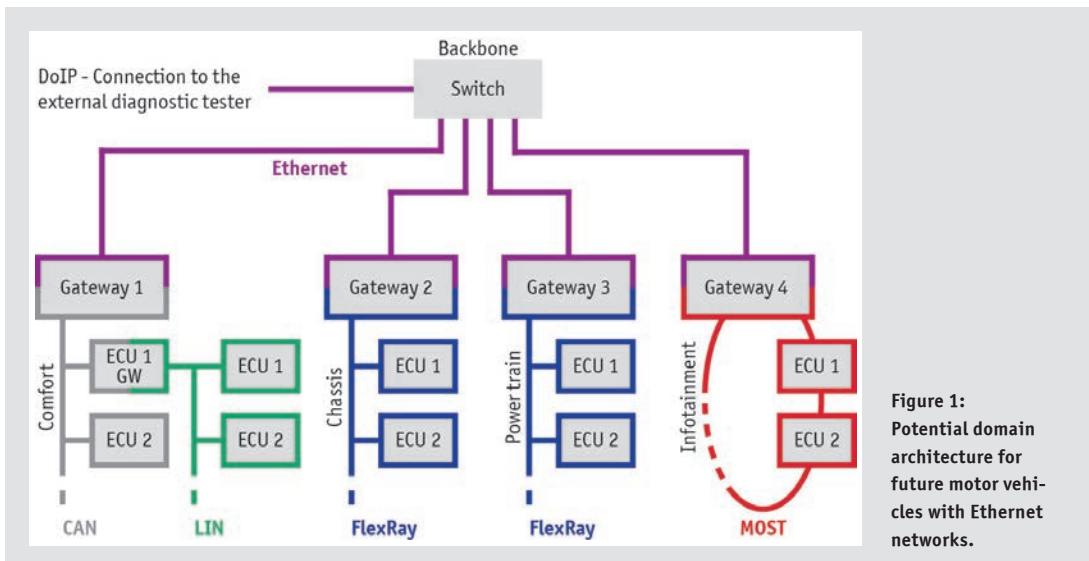
common off-the-shelf TCP/IP stack is recommended here.

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. 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**).

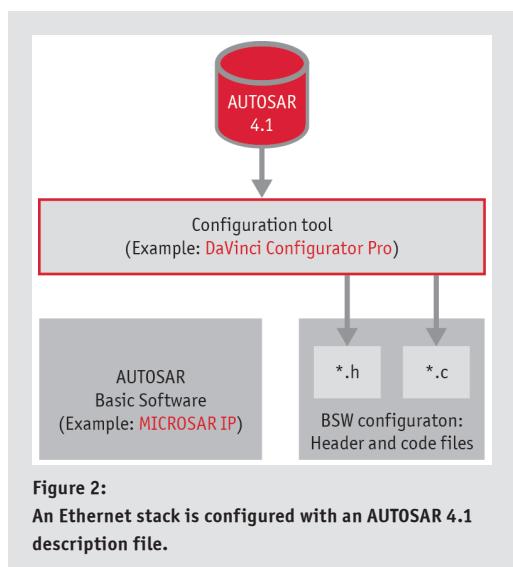


Figure 2:
An Ethernet stack is configured with an AUTOSAR 4.1 description file.

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 necessary 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.

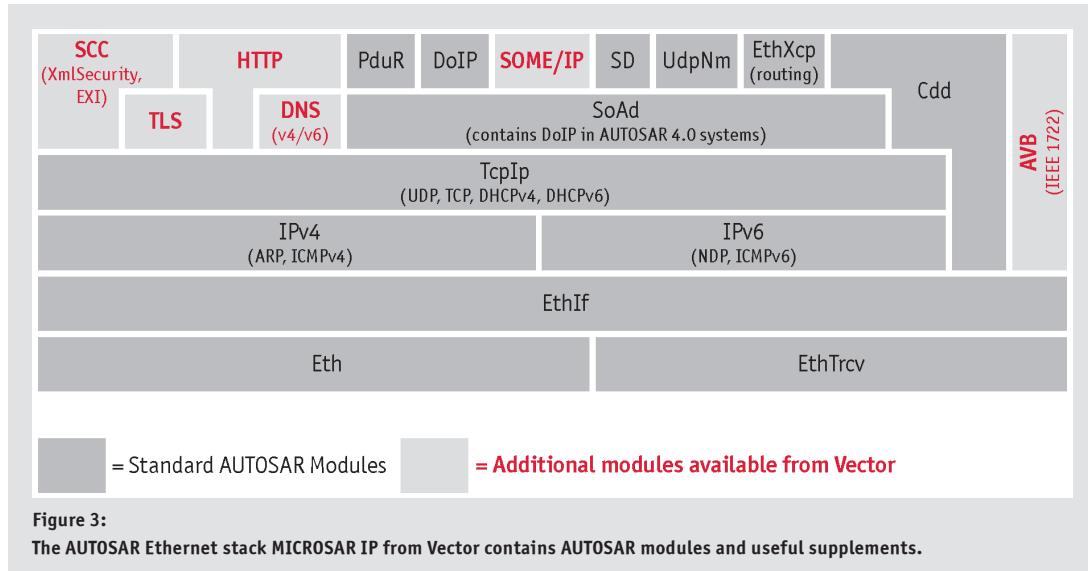
Routing of DoIP over a gateway to a sub-Ethernet network is not standardized in the ISO 13400 specification. Nonetheless, solution approaches have already been worked out with various automotive OEMs.

The Ethernet stack defined in AUTOSAR is available from Vector as ECU software under the product name MICROSAR IP (**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 IP 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

All figures: Vector

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

>> Contact information for all locations of the Vector Group:
www.vector.com

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

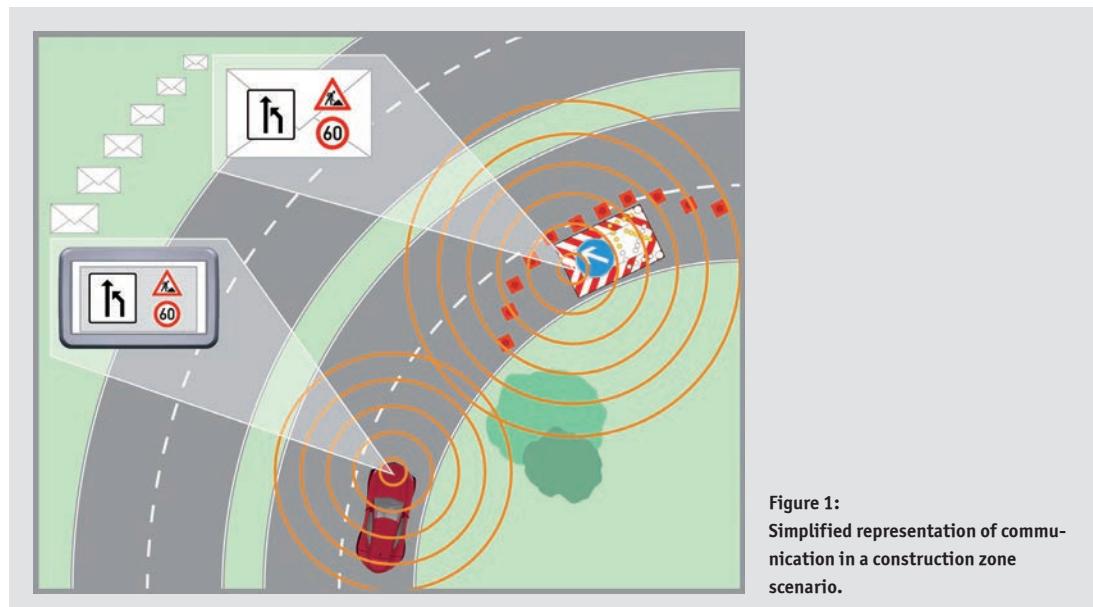


Figure 1:
Simplified representation of communication in a construction zone scenario.

IEEE802.11p (ITS-G5) standard over the 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 interpreted 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.

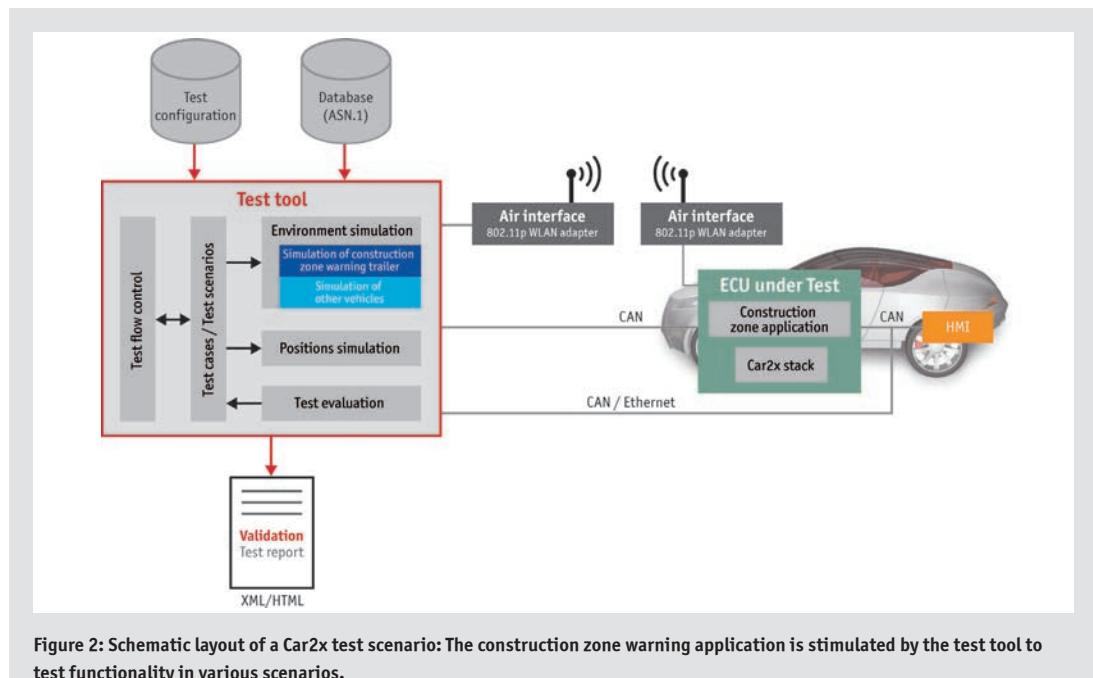


Figure 2: Schematic layout of a Car2x test scenario: The construction zone warning application is stimulated by the test tool to test functionality in various scenarios.

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**

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_canoecar2x_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.



**>> Contact information of the Vector Group:
www.vector.com/contact**

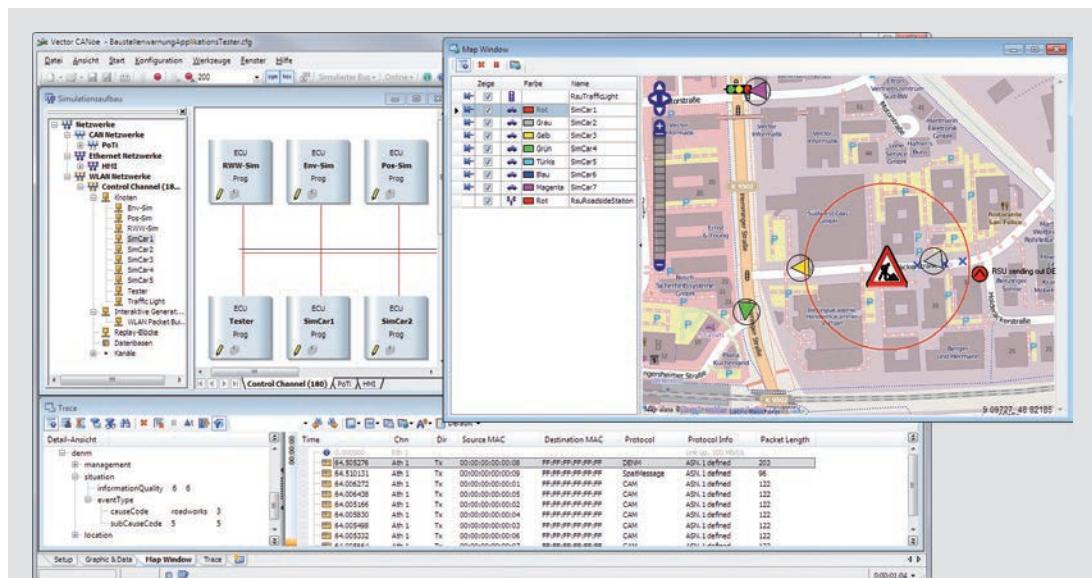
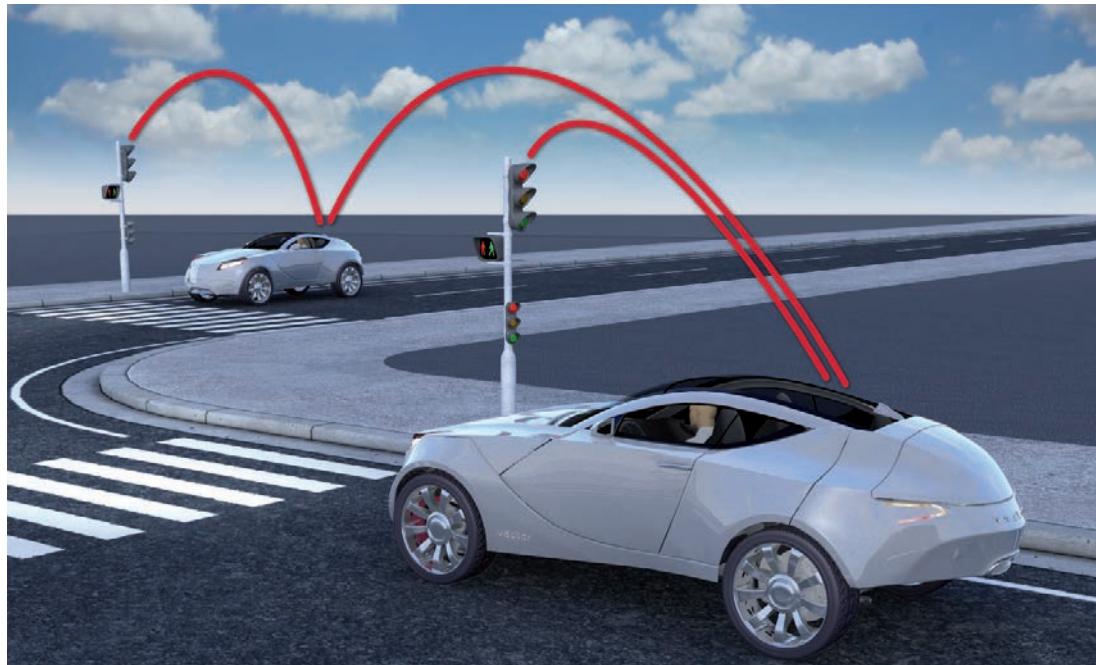


Figure 3: The entire test scenario can be implemented and visualized in CANoe.Car2x. Clear representation of the ITS stations in the map window with a display of the driving direction.

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

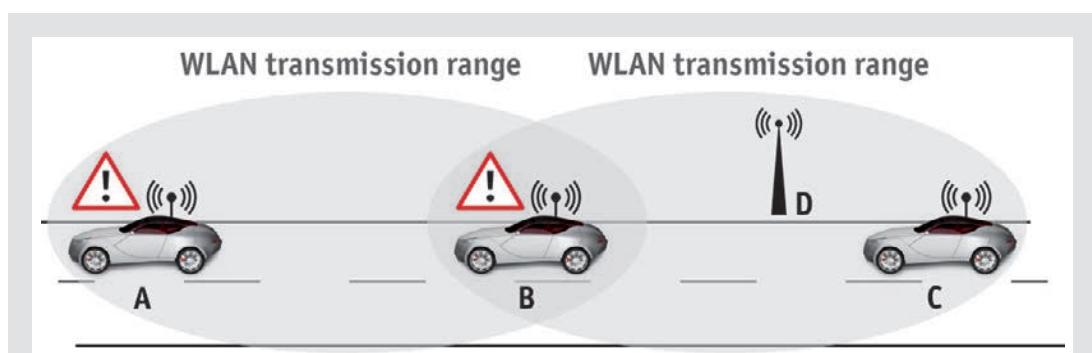


Figure 1:

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.

sending and receiving on the same radio channel. This requires support of the air 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

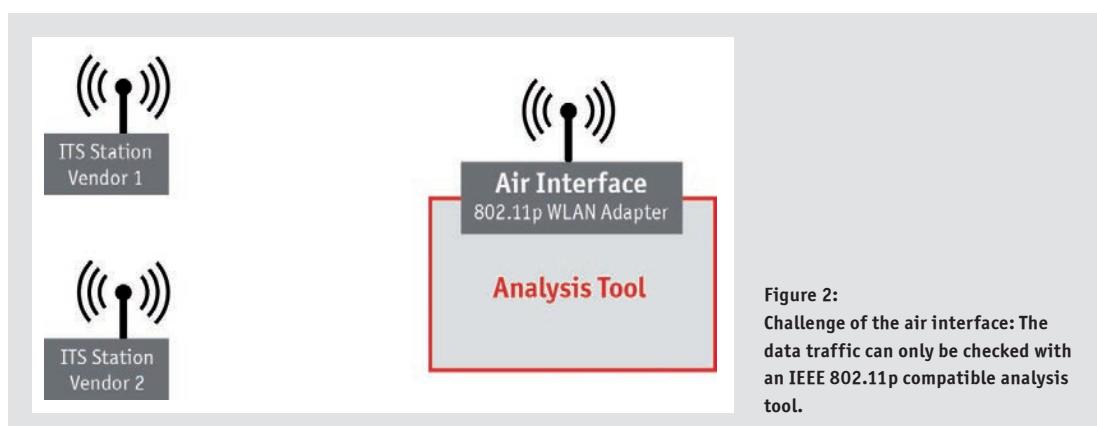


Figure 2:
Challenge of the air interface: The data traffic can only be checked with an IEEE 802.11p compatible analysis tool.

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 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 IP. 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 WLAN 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.

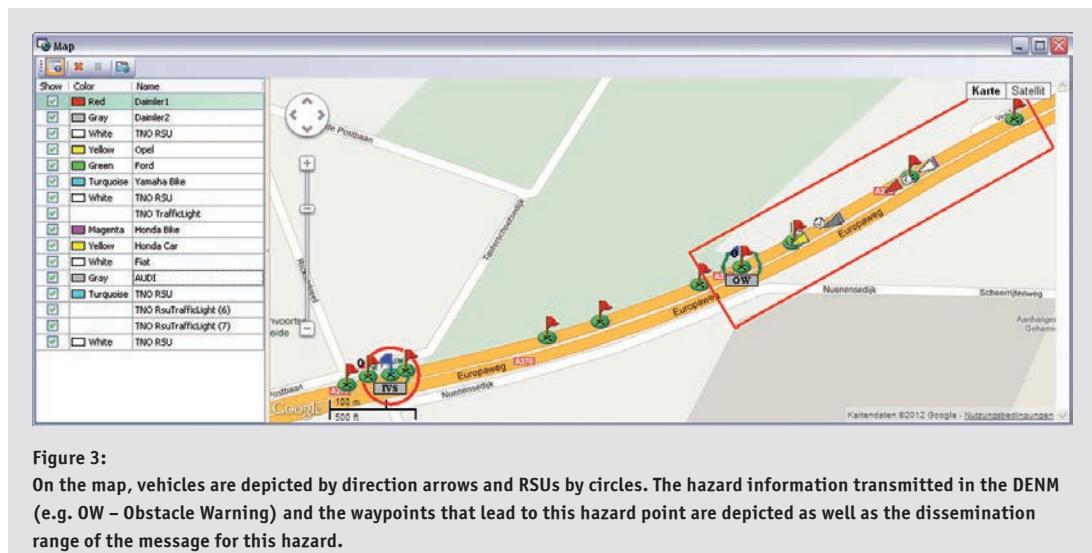


Figure 3:

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.

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.

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=oksfjr
- [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_canoе_car2x_en.html

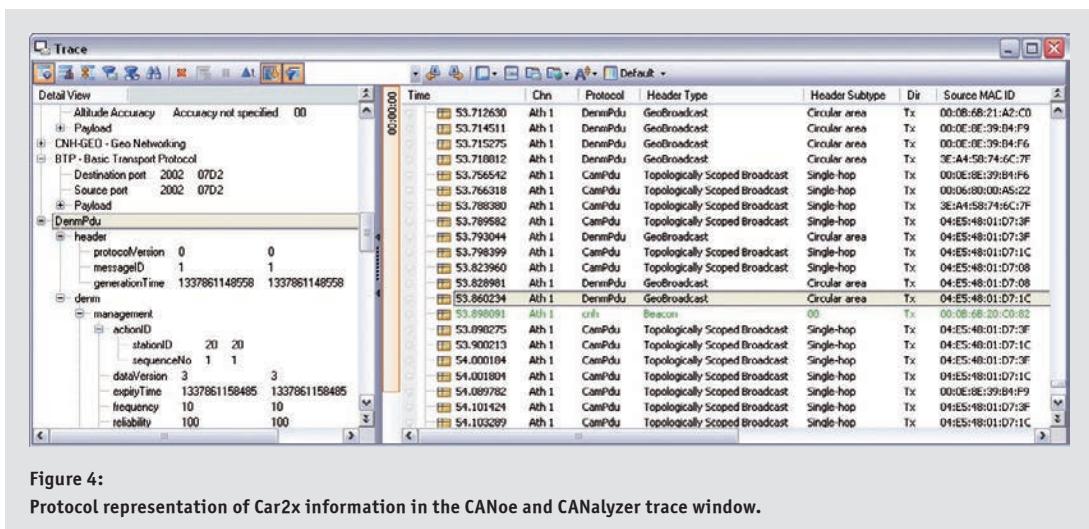


Figure 4:

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

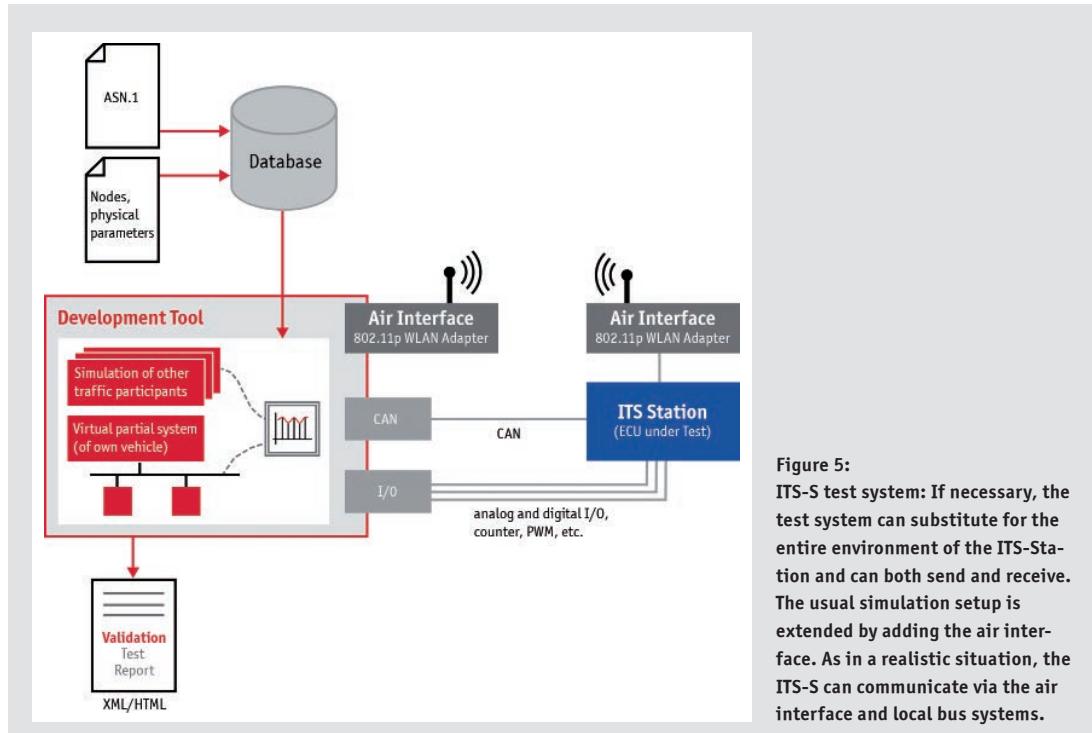


Figure 5:
ITS-S test system: If necessary, the test system can substitute for the entire environment of the ITS-Station and can both send and receive. The usual simulation setup is extended by adding the air interface. As in a realistic situation, the ITS-S can communicate via the air interface and local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



Hans-Werner Schaal
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 at Vector Informatik GmbH for the Open Networking product line and is responsible for IP and Car2x. Previously, he worked in various industries as development engineer, project leader and product manager in the test tools area for several network technologies.



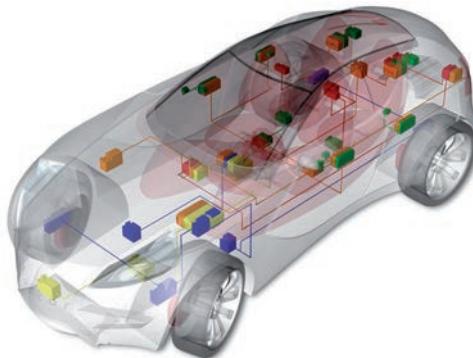
Thomas Löffler
(Graduate Engineer) is Senior Software Development Engineer at Vector Informatik GmbH working in the area of Car2x. His areas of focus are product definition and project management of customer projects. Mr. Löffler also represents Vector on a number of Car2x standardization committees.

Serial Bus Systems in the Automobile

Part 5:

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.



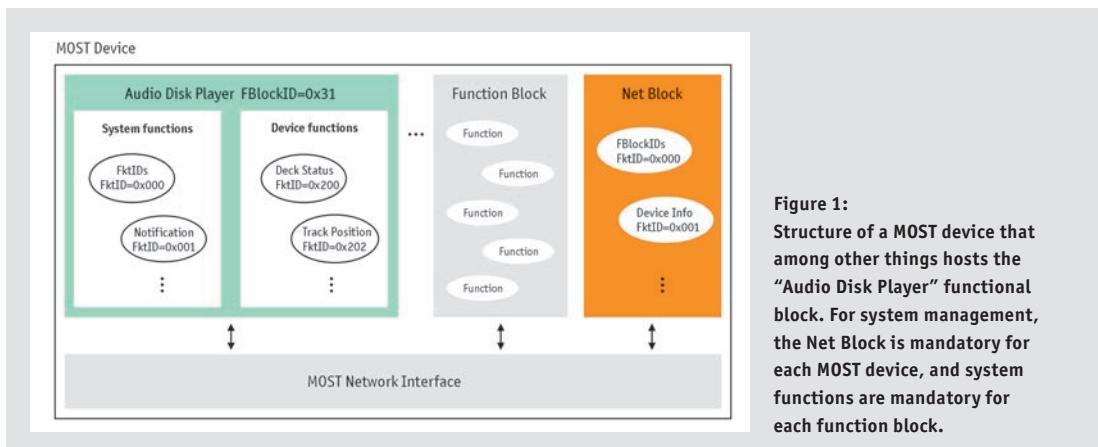
Electronics is responsible for a large number of innovative safety and convenience functions in automotive technology. Experts predict that in just a few years electronics will represent a share of up to 30 percent of vehicle value, and the worldwide market for electronics in cars will grow by approx. 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.

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 “SetGet” 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.

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.

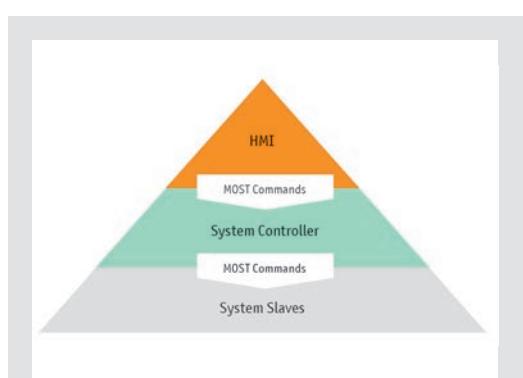


Figure 2:
Hierarchy of a MOST system

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

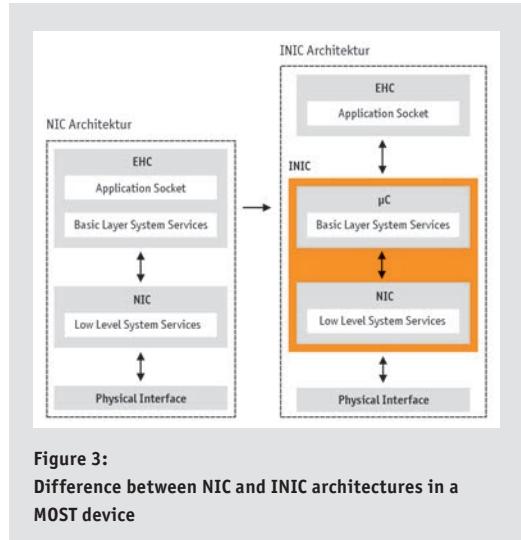


Figure 3:
Difference between NIC and INIC architectures in a
MOST device

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



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.

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).

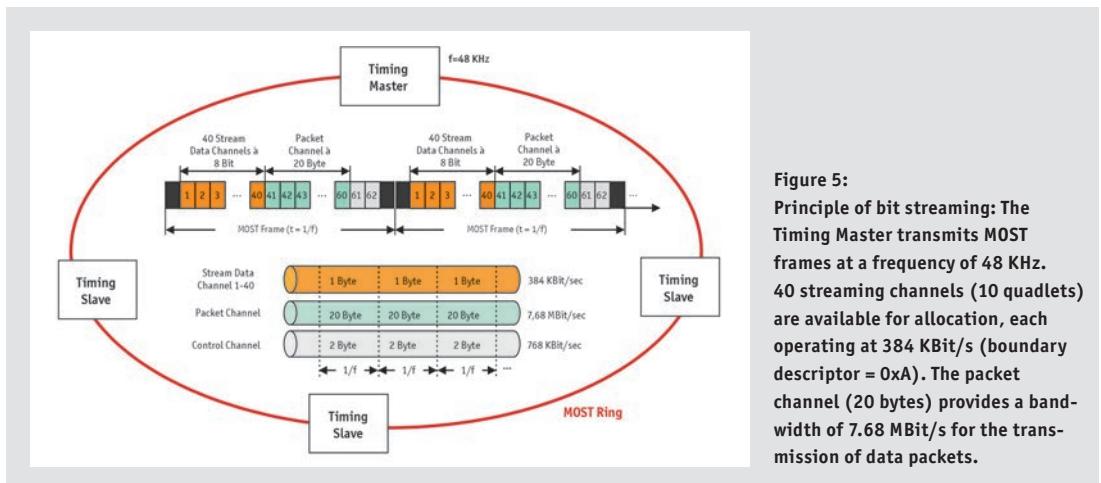


Figure 5:
Principle of bit streaming: The Timing Master transmits MOST frames at a frequency of 48 KHz. 40 streaming channels (10 quadlets) are available for allocation, each operating at 384 KBit/s (boundary descriptor = 0xA). The packet channel (20 bytes) provides a bandwidth of 7.68 MBit/s for the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

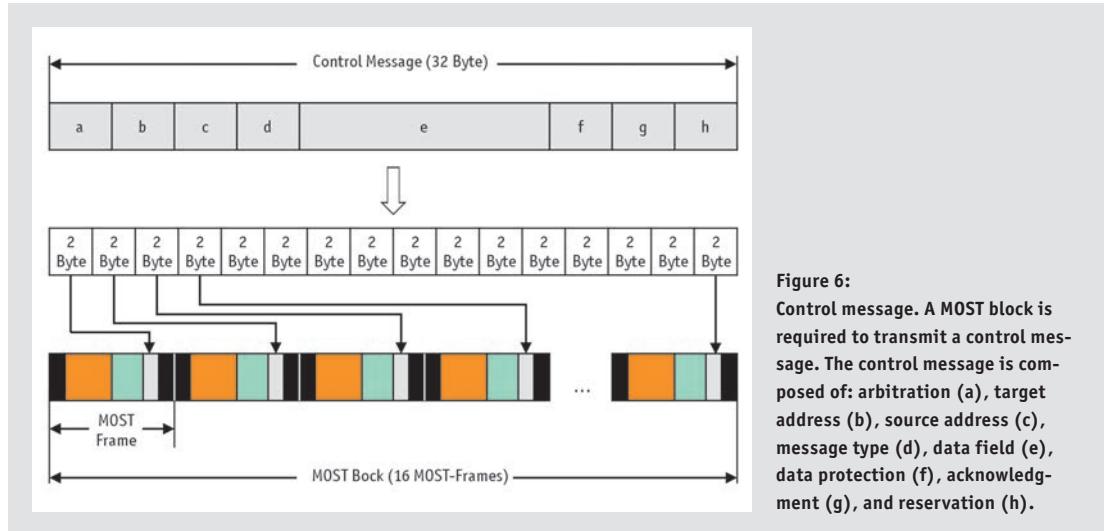


Figure 6:
Control message. A MOST block is required to transmit a control message. The control message is composed of: arbitration (a), target address (b), source address (c), message type (d), data field (e), data protection (f), acknowledgement (g), and reservation (h).

Physical Layer

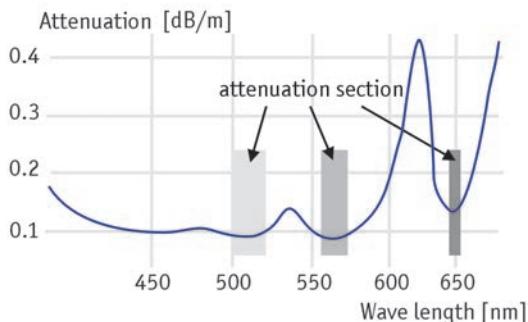
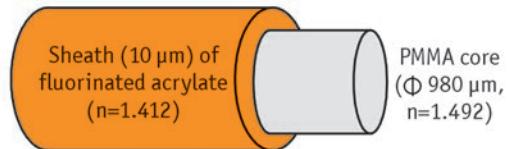
Today, optical conductors of polymer fibers (POF – polymer optical fiber) are the state-of-the-art technology for transmitting audio and video signals in the MOST system (Figure 7). Overall, the technical properties of polymer fibers are far superior to those of electrical transmission media. Especially 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 currently 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 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.

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 α 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 a_0 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

Translation of a German publication in
Elektronik automotive, 9/2007

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.

Tool-supported Data and Process Management at MAN Nutzfahrzeuge AG

At MAN Nutzfahrzeuge AG an integrated approach was applied to managing all engineering data generated in the E/E development process and its subprocesses. The goal is to further improve the efficiency and quality of development, despite the growing complexity of electronic systems. MAN Nutzfahrzeuge AG developed and introduced an integrated development database for this purpose, which is based on the eASEE Tool Suite from the Vector company:

The MAN Common Engineering Data Backbone.



1 Motivation and Goals

The development of vehicle functions, ECUs and ECU networks is becoming more and more complex. It is becoming increasingly important for MAN Nutzfahrzeuge AG to be able to master this complexity into the future. The primary objective is to increase development efficiency while further improving product quality.

2 Basic Ideas

Two factors led to the development of the MAN Common Engineering Data Backbone: First, it was clear to management very early on that quality could not simply be achieved afterwards by an extensive testing process. Rather a universal, well lived-out development process is necessary, which encompasses all areas of development. Second, management at MAN favored an integrated database solution that saves all data of the development process in a meta model (single source) [Figure 1] and thereby enables very efficient data

usage and data universality. The ability to set relationships between data further increases efficiency. This database solution is referred to at MAN as the Common Engineering Data Backbone.

3 Implementation

The company was looking for a technical platform that lets users concentrate on the actual contents: The data structures and functionalities. The system already provides basic mechanisms such as user administration, version management and client-server architectures. That is why the choice was made to go with the eASEE Tool Suite from Vector.

3.1 eASEE as Technology Platform

eASEE is a process tool whose core consists of a hierarchical configuration management system for any desired data. This basic system includes:

- > Functions for versioning and variant formation
- > A user-configurable data model for user data and meta data
- > A workflow engine
- > Multi-site operation, and
- > A differentiated roles and rights concept.

Overlaid on this basic system are modules specifically designed for various process areas. These modules cover the majority of key process functionalities that are needed in the automotive industry [Figure 2]. Programming interfaces are provided to allow for individual extensions. Besides being used at MAN, eASEE is also used at Bosch, General Motors, Daimler, ZF, Volvo, Porsche, VW, Audi and Getrag.

3.2 The MAN Common Engineering Data Backbone

Today, the MAN Common Engineering Data Backbone consists of eight domains. Its foundation is an Oracle database upon which the eASEE Tool Suite is placed [Figure 3].

The process model distinguishes between the actual development process ("Do it") [Figure 1] and the management process ("Control it") [Figure 4]. Analogously, in the MAN Common Engineering Data

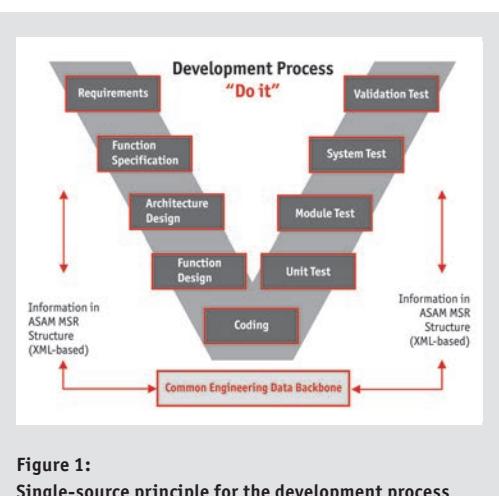


Figure 1:
Single-source principle for the development process

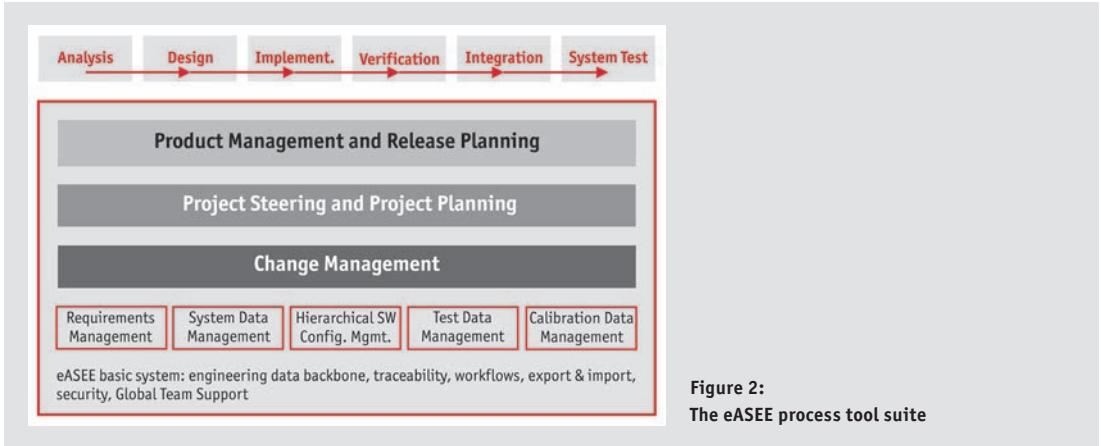


Figure 2:
The eASEE process tool suite

Backbone there are “Do it” data domains (FDM, TDM, CDM, etc.), and a “Control it” project management domain (PPM) [Figure 3]. The two areas are interlinked across all domains. Project schedules (Gantt Charts), for example, are automatically updated based on the states of the elements in the data domains. Thus, the project leader no longer needs to perform maintenance work on the states of work packets.

Function Data Management – FDM

The individual data domains are oriented directly toward the process. Function Data Management (FDM), for example, represents the left side of the V-Model. This domain contains a complete meta model used to describe all of the data of an electronic structure, including:

- > Vehicle configurations
- > ECUs
- > Hardware (connectors/pins)

- > Signals (e.g. CAN)
- > Vehicle functions
- > Functions
- > Software architectures.

In addition, a classic requirements management system is represented.

Based on these data structures, the FDM offers many functionalities for the daily tasks of the engineer. These include:

- > Various interfaces to expert tools, e.g. Matlab/Simulink/TargetLink as a development environment and XMetal as an XML-based authoring environment for detailed specifications [Figure 5]
- > The option of a virtual vehicle design/check,
- > Signal path analysis
- > Ability to generate DBC files to describe bus systems for analysis and test tools such as Vector’s CANalyzer and CANoe.

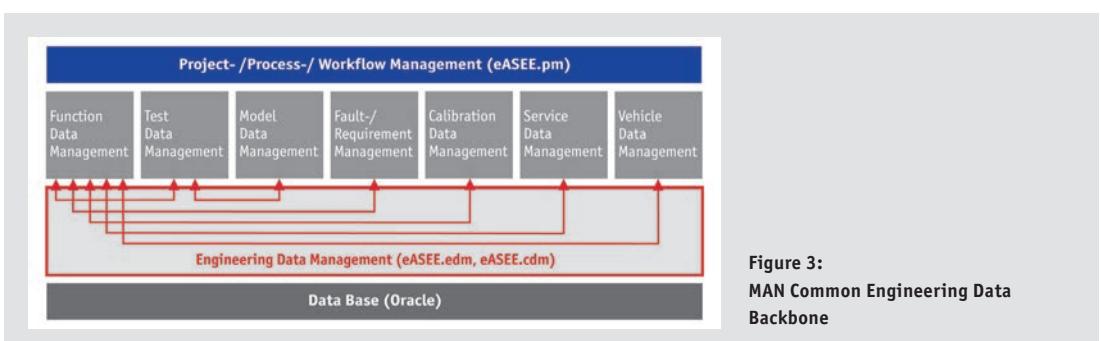


Figure 3:
MAN Common Engineering Data Backbone

Test Data Management – TDM

The right side of the V-Model is covered by Test Data Management (TDM). In this domain it is possible to map entire test projects. Among other things, TDM manages the test specifications, test execution information for each test, test results, test environments and test methods. Test data in the TDM are directly linked to development data in the FDM. Since the system is used cross-departmental and over all test levels, it is possible and very easy to draw conclusions about test coverage – from component testing to validation testing.

Fault and Requirement Management – FRM

Fault and Requirement Management (FRM) contains a complete change management. This system can be used to input issues into the development process. The individual issues may be classified as errors or new requirements.

Errors are linked to both the test cycle in which the error was found, in the TDM, as well as to the function in which the error occurred. This makes it very easy for function developers to look in the FDM to see which issues are still open for them. The test engineer, in turn, sees very quickly which errors were corrected in a new function version, and what form of post-testing is necessary.

Calibration Data Management – CDM

Calibration Data Management (CDM) maps the data of the calibration process. Based on the data of the FDM, in calibration projects this system makes it possible to manage data records of ECUs via the ASAM MCD-2MC file. Data records from established calibration

tools such as CANape, INCA or Caldesk – which have an ASAM MCD2MC interface and which support the CDF or DCM exchange format – may be read directly into the CDM. Completed and released parameter sets may be referenced in the FDM as initial data sets.

Service Data Management – SDM

Service Data Management (SDM) makes it possible to provide service-relevant data to the service area from the development area. Also represented in this system is a workflow mechanism that tracks the path from the released ECU to its production and use in series vehicles.

Traceability

Today, almost all data generated in the development process is saved in a meta model in the MAN Common Engineering Data Backbone. Integrated data storage in a database enables nearly perfect traceability of the data. For example, starting from a test cycle it is possible to determine which error was found with this test cycle and which requirement the test cycle covered.

Another potential use is to start with a CAN signal and have the system indicate which functions in an electronic structure utilize this signal. For example, it is possible to deduce whether a certain sensor can be omitted in the system.

3.3 Status of the MAN Common Engineering Data Backbone

The MAN Common Engineering Data Backbone has been in productive operation for several years and is currently utilized by about one hundred developers. Today it consists of the eight domains

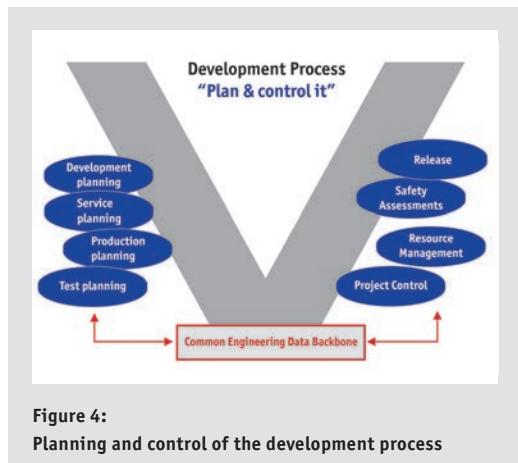


Figure 4:
Planning and control of the development process

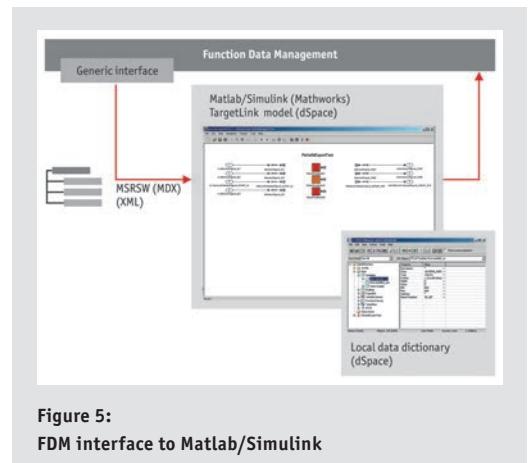


Figure 5:
FDM interface to Matlab/Simulink

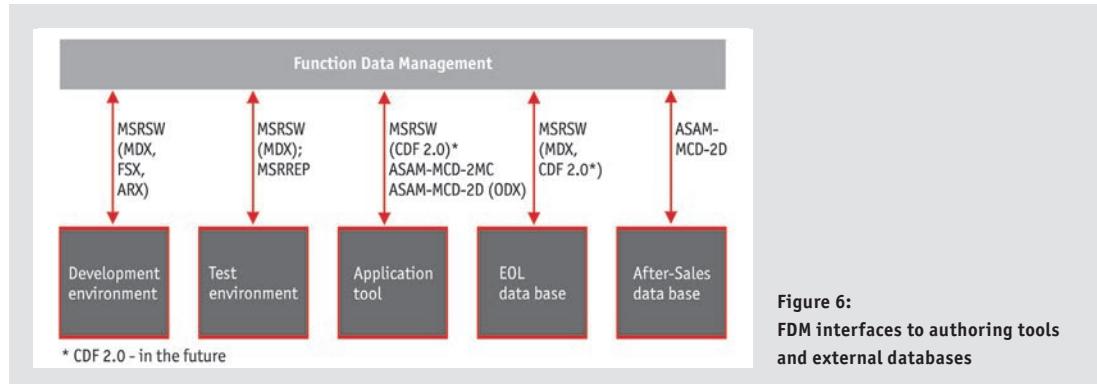


Figure 6:
FDM interfaces to authoring tools
and external databases

described and various interfaces [Figure 6]. Since developers are actively supported by the backbone in their daily work, from the first requirement to the last test, the Common Engineering Data Backbone is very well accepted by developers. It has been shown that it is very important for developers to work within the system from the very first minute, and to single-source their data here. This eliminates an additional maintenance effort that would otherwise be perceived as rather disruptive.

4 Utility

After about three years of productive use by the Common Engineering Data Backbone, the initial objectives were realized. Efficiency gains were achieved due to redundancy-free data input and data storage and due to the many different options for supplying information and preparing data. Previously, an engineer would have to work through dozens of documents to predict the consequences of transferring a certain function from ECU A to ECU B. Today, it takes just a few mouse clicks and the relevant information is available at the latest revision level. Another advantage is reusability of engineering artifacts and data, which is made possible by Variant Management. Other positive effects are the considerably improved transparency for engineers involved in the development process and the ability to establish role-specific views for one and the same data content. Thus, the MAN Common Engineering Data Backbone offers very specific views of development status, e.g. for the project leader, system architect, function developer, integration manager and test engineer.

5 The Road Ahead

In implementing the MAN Common Engineering Data Backbone, special attention was given to exchange formats to ensure that

established standards such as MSRSW and ASAM were used. Today, this is very beneficial especially at supplier interfaces. Therefore, MAN is following the development of the AUTOSAR standard very closely. If the advantages of the standard appear sufficiently great, the MAN Common Engineering Data Backbone would be adapted to the AUTOSAR standard and brought to a level of AUTOSAR compatibility. The interests of MAN and Vector overlap here too: The Stuttgart-based tool producer has set the goal of developing an eASEE module for AUTOSAR-compatible system data management, which combines the advantages of the standard with key functionalities of the Common Engineering Data Backbone.



Stefan Teuchert
(Graduate Engineer) manages the Software Development and Basic Technologies department at MAN Nutzfahrzeuge AG

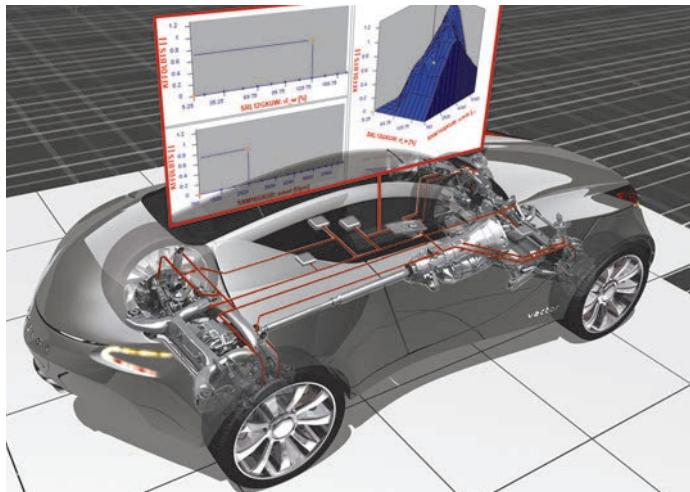


Georg Zimmermann
(Graduate Engineer) manages the "Product Line Process Tools" department at Vector Informatik GmbH

From Pilot Studies to Production Development

Efficiency and Quality in calibrating Transmissions

To soundly manage growth in the number of complex transmission calibration projects and their data at ZF Friedrichshafen AG, the company needed to introduce a new calibration data management system. Deciding factors for introducing the calibration data management system eASEE.cdm from Vector were the tool's high functionality, flexibility and potential. Another crucial factor was the companies' development partnership over many years with the goal of jointly meeting ZF targets for quality assurance and improved efficiency.



To be successful as a globally-active automotive supplier in a market characterized by innovation, competition and cost pressure, it is necessary to have products that meet the stringent technical and economical requirements of automotive OEMs.

Based on its comprehensive expertise, continuously optimized processes and process tools, ZF is able to position its products for powertrain and chassis systems very successfully in the market. In the powertrain area, ZF supplies transmissions for passenger cars, commercial vehicles, buses, rail vehicles, ships and helicopters. These areas are marked by a very high level of model diversity. This diversity in models and variants can be achieved by vehicle-specific parameterization (calibration) of the transmission. For example, ZF supplies the 6HP26 6-speed automatic transmission, which enables efficient gear shifting in different vehicles with vehicle-specific torque curves ranging from 300Nm to 800Nm; this is accomplished by individual parameterization (Figure 1). Short shift times, smooth gear shifts, reduced fuel consumption and low emissions are other objectives that can be met by optimally adapting parameters to the vehicle.

Requirements of ZF Friedrichshafen AG

Even the first microcontroller-based transmission control electronics included parameters for adapting the electronics to the environment (e.g. transmission hardware, vehicle). Starting with just a few calibration parameters, the number of parameters continually rose in response to increasing functionality. A growing model diversity in transmission and powertrain systems – as well as a rising number of parallel projects – required functions for central, efficient and process-conformant management of the calibration data. The existing system for data management developed in-house at ZF had been stretched to its limits by the complexity of projects.

Objectives for the new calibration data management system were:

- > Uniform and corporate-wide system for central management of all ZF drive and chassis projects
- > Introduction of a modern, market-established Engineering Data Management system (EDM system) with a database orientation
- > Support and standardization of defined calibration processes
- > Integrated checking and test routines for quality assurance
- > Mass operations to improve efficiency
- > Flexibility of data storage – from simple addressing via variant-encoded groups for multiple systems or vehicles to complex data storage for many vehicles

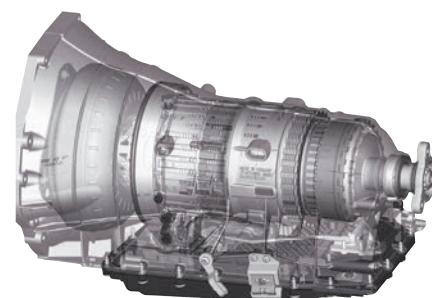


Figure 1:
The 6HP26 transmission from ZF

Vector's eASEE.cdm Solution

To fulfill the requirements outlined by ZF, a tool was needed that integrates the functions of EDM systems, process tools and calibration tools in a comprehensive system solution. In deciding on eASEE.cdm from Vector, ZF chose a mature and market-established system.

The eASEE.cdm calibration data management system consists of a data management system for engineering artifacts and a graphic Parameter Editor; it also contains calibration-specific functions and automated sequences for managing all program and data sets occurring in the calibration process. As a modular component of the eASEE tool suite, it is also easy to integrate it with other process domains (Figure 2).

Functions of the eASEE base system include:

- > Functions for versioning, and for forming variants and configurations
- > Flexibly configurable data model for productive data and the meta-data that describes it
- > A workflow engine for process-conformant flow control
- > Multi-site operation for cooperation between distributed work teams
- > A differentiated roles and rights concept

Flexible configurability of the data model makes it easy to implement application-specific extensions – eASEE.cdm becomes the data backbone of calibration processes.

Calibration-specific functions support project leaders, data integrators and calibrators in all phases of a project. From project definition to data preparation to release of the integrated overall parameter sets, users benefit from calibration data management system functionality:

Project leaders at ZF are able to clearly structure complex calibration projects that have many variants by property-supported variants management. Data set variants are formed by combinations of variant-relevant attributes. Variant criteria are also utilized as filter criteria in releasing parameters (Figure 3).

ECU suppliers often perform calibration projects jointly with the OEM, and this requires flexible handling of data. Calibrators at ZF process and manage ECU parameters – either parameter-oriented, function-oriented, component-oriented or variant-coded, depending on the specific calibration process. The CANape and INCA calibration tools that are used generate market-relevant data formats: both physical characteristic data formats (CDF, CSV, DCM, PaCo, PAR) and program formats (Intel Hex, Motorola-S) are supported. The calibration area at ZF implements the graphic Parameter Editor for checking, adjustment and data integration (Figure 4). It offers extensive options for visualization (table-format, 2D and 3D) and offline editing of ECU programs and parameters. Changes are saved directly in the eASEE data backbone.

Data integrators make extensive use of algorithms to improve data consistency and integrity by checking for completeness, unambigu-

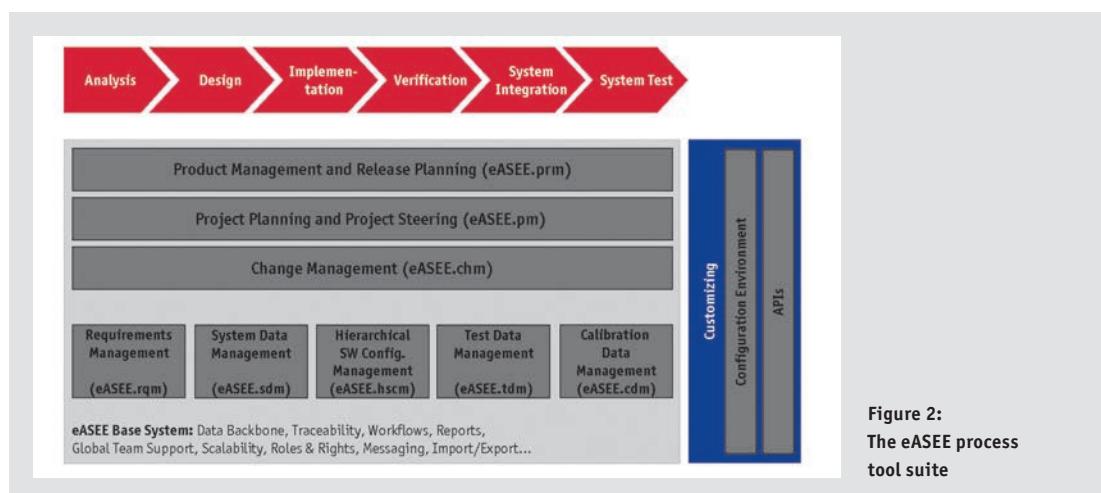


Figure 2:
The eASEE process tool suite

ity and physical limits. Project leaders at ZF appreciate the added security offered by label-based rights management, which prevents parallel releases of overlapping data. Other important functions have been integrated in the calibration data management system, such as checksum generation and the ability to interface to signature tools for protecting ECU software.

Introduction of the System

The decision to implement eASEE.cdm as a corporate-wide solution at ZF was made in 2003. Before roll-out in the individual business areas, eASEE.cdm was first introduced and evaluated in two pilot projects. After successfully completing the pilot phase, the system was introduced to daily business operations in 2004. The first production projects were the ECOMAT (automatic transmission for city buses) and ASTRONIC (automatic transmission for trucks and buses) projects. Today, approx. 150 calibration engineers manage project and calibration data for 20 projects in different domains. This enables customer-specific separation of project data.

Implementation of ZF requirements was based on a multi-year development partnership. Close cooperation made it possible to meet targets in a timely way. The focus in implementing requirements was on general usability of the jointly developed functions.

Utility

Production calibration projects at ZF are performed using calibration processes that are defined area-wide. The processes are structured so that many process steps can be run in parallel and executed by an automated tool. All aspects of this calibration process at ZF are fully supported by eSEE.cdm. It ensures process-conformant execution of calibration projects in practice, and it supports of calibrators, developers and project leaders.

In their daily work, users benefit from the following functionalities:

- > Improved cooperation of calibration teams due to uniform and corporate-wide system
 - > Data freeze time is reduced from one week to one day by automated checking and safeguarding functions
 - > Reduced manual effort by automated generation of the data exchange containers
 - > Reproducibility of data revision levels by automatic storage of test and signature configurations
 - > Effective variants management reduces number of time-wasting and error-prone actions
 - > Collision-free data integration by label-based rights management
 - > Full traceability of the calibration process by thorough versioning of all relevant data and protocols
 - > Reusability of calibration data by component library

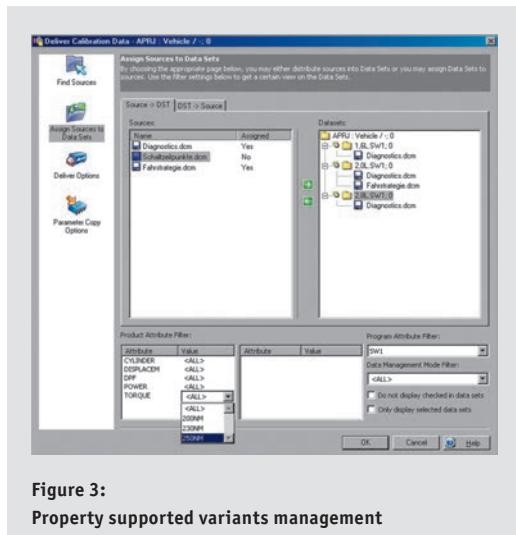


Figure 3:
Property supported variants management

Summary

The eASEE.cdm calibration data management system has been introduced corporate-wide at ZF, facilitating efficient, process-conformant data management while ensuring high data quality. Besides being used to support the business operations, the tool also serves as an important foundation in conducting (SPICE) assessments of ZF business processes in software development and calibration.

All of the objectives ZF sought to achieve were fully attained. Significant increases in efficiency of up to 90% were attained in safeguarding and signing-off data sets. Overall, eASEE.cdm is making an important contribution toward reducing costs in calibration.

Translation of a German publication in Automobil-Elektronik, 3/2008



Rainer Röhre

studied Physical Technology and Computer Engineering and has been employed at ZF since 1997. He is responsible for tool development in the Electronics/Software area. Since 2003 he has been project leader for the corporate-wide introduction of eASEE.cdm.



Christoph Heller

studied Electrical Engineering at the University of Stuttgart. He works at Vector Informatik GmbH as Business Development Manager in the Process Tools product line area.

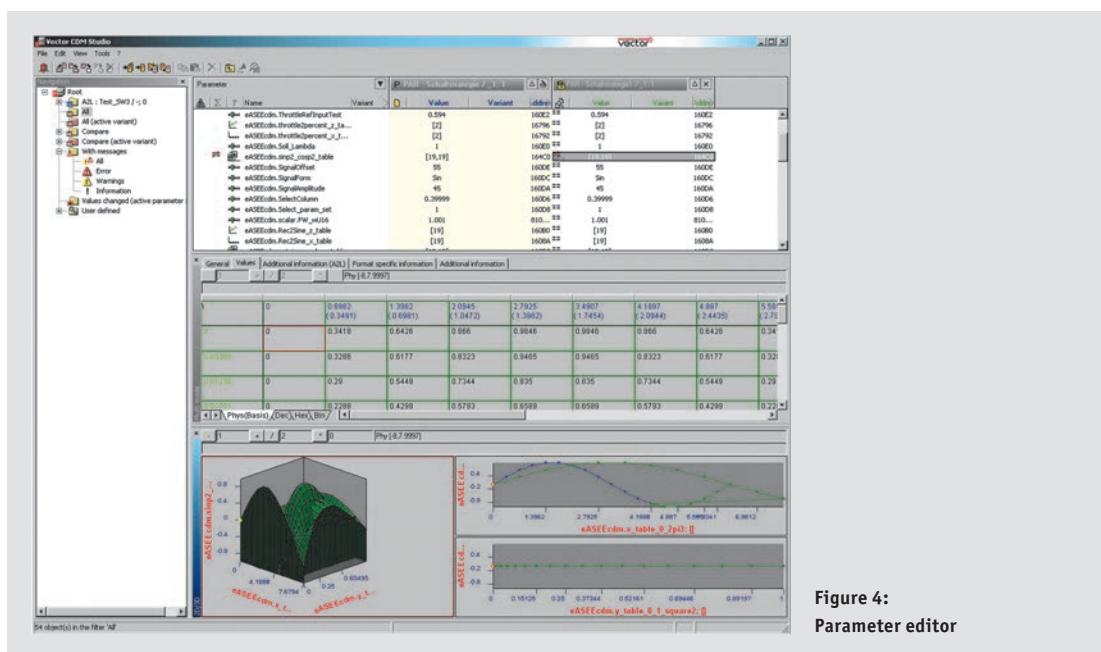


Figure 4:
Parameter editor



PREEvision

Model-based E/E Development from Architectural Design to Series Production



Develop, visualize and evaluate your E/E architecture with PREEvision.



Highlights:

- > Requirements management
- > Functional safety analysis (ISO 26262)
- > AUTOSAR support
- > File management
- > Product and release management
- > Change management



Optimize your E/E system architecture in an early stage of development with PREEvision.



► Information and downloads: www.vector.com/preevision



Case Study

Reusing functions – keeping consistency when complexity increases



The Customer

Continental is one of the world's leading automotive suppliers. The Powertrain Division of the Automotive Group integrates innovative and efficient system solutions related to the drivetrain. The product lineup of the Engine Systems business unit ranges from electronics such as ECUs and software to complex components such as high-pressure pumps or injectors.

The Challenge

Reuse components and functions with growing complexity and assured consistency

In development the growing complexity in conjunction with the large number of functions leads to large volumes of data. In addition, there are always new requirements and standards related to exhaust emissions and CO₂ output, which the Continental AG must also consider in its development projects. In designing an injector, for example, flexible reuse of all development artifacts is necessary. All components and functions must be uniquely defined and seamlessly relate to each other.

The Solution

Definition and linking of architecture, feature models and requirements

Continental uses PREEvision to define and set up its development platform. In the 'Reuse Unit', all functions and action areas such as an injector are defined and managed. Then the product types are derived from the platform. PREEvision supports selection of the hierarchically defined product features and also considers stored parameters and their conditions. In the next step, products are exported out of this product lineup. The 'Resolver' is used to check the selected product features for consistency, universality and completeness.

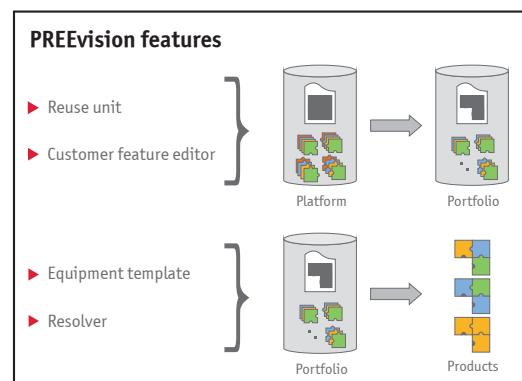
PREEvision utilizes a plausibility check to detect any rule violations, that can be corrected quickly.

The Advantages

Increased quality by flexible and controlled use of all development artifacts

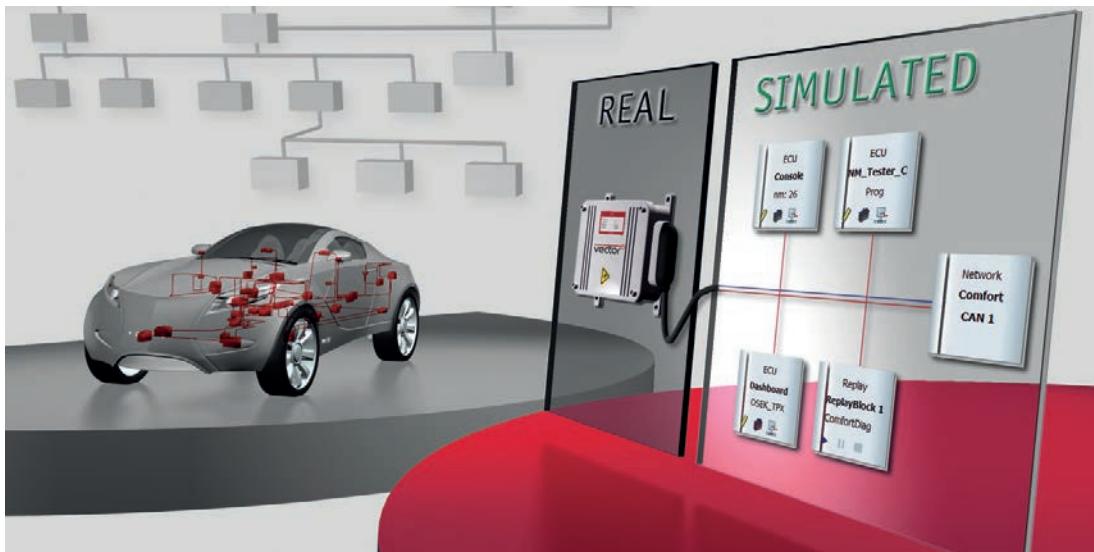
Continental traced four development goals which are supported by PREEvision:

- ▶ Reuse of components and functions accelerates development and supports the control of complexity (development goal: reuse)
- ▶ Configurable functions and components enable quick representation and evaluation of alternatives (development goal: flexibility)
- ▶ Checking consistency of functionality in use of variants (development goal: quality)
- ▶ Focused test activities, e.g. limited to integration tests (development goal: control)



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.

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

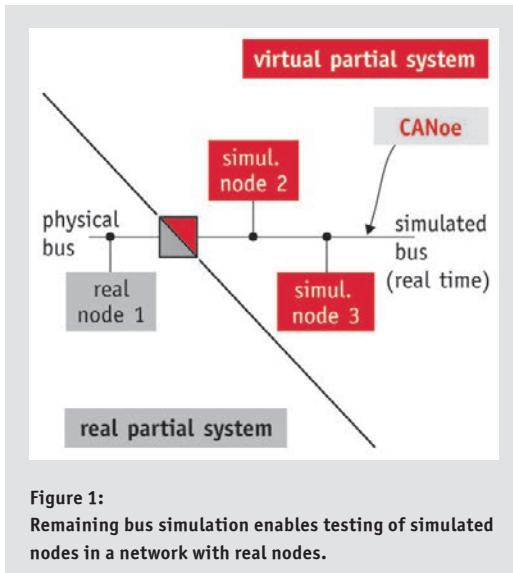


Figure 1:
Remaining bus simulation enables testing of simulated nodes in a network with real nodes.

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.

Stimulation by control panels and run sequences

Applications on the upper OSI layer include signal panels and signal generators, for example. They are indispensable 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.

Node Panel - easy - Engine			
Engine Messages Grouping Columns Edit			
Message			
<input checked="" type="checkbox"/> EngineState			
Signal Physical Value Raw Value Symbolic Value			
<input checked="" type="checkbox"/> EngineS...	0	0x0	-
<input checked="" type="checkbox"/> OnOff	0	0x0	Off

Figure 2:
Standard panels let users interactively modify signal values.

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 powerful 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 detect states, events and state changes (Figure 4). This type of representation is realized by an Art Logic Analyzer for automotive networks or ECUs.

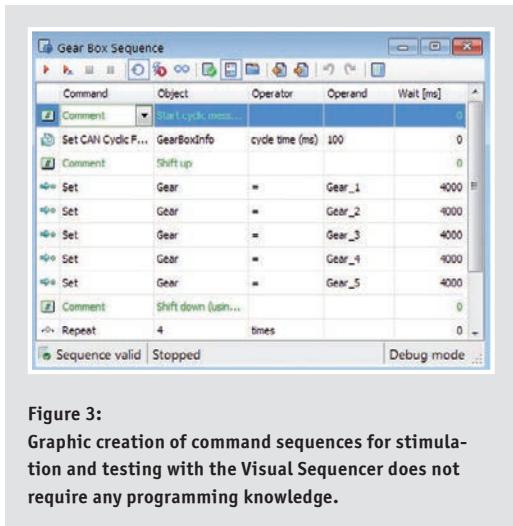


Figure 3:

Graphic creation of command sequences for stimulation and testing with the Visual Sequencer does not require any programming knowledge.

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.

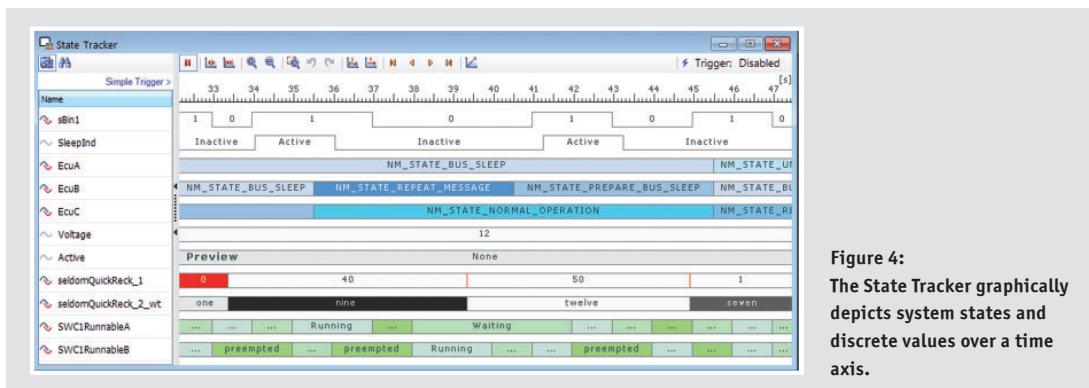


Figure 4:

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

Figures:

Vector Group

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.

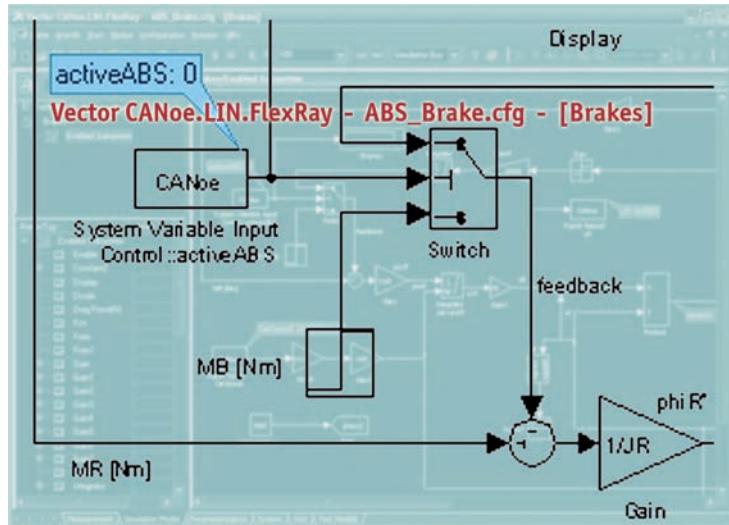


Peter Decker

has been employed at Vector Informatik since 2002 and is currently working as a product manager in the multibus development area for CANoe/CANalyzer.

Model-Based Development of ECUs

Software Simulation with MATLAB/Simulink and CANoe



MATLAB/Simulink is a tool that is widely used in many engineering and scientific disciplines. In the automotive field, for example, it is used to model dynamic systems such as control systems, and to describe states and their transitions. Since these algorithms run on ECUs that communicate with one another, it is vital to provide access to the vehicle network over the course of the development process.

Access is provided directly from the model via Simulink blocks that communicate with the bus hardware. Nonetheless, a number of disadvantages are associated with this approach:

- > The model is always network-specific. Although the network and application content can be separated by intelligent structuring with subsystems, it is more advantageous to have a generic model that can very easily be adapted to different networks.
- > Additional protocol layers such as Network Management (NM), the Interaction Layer (IL) or Transport Protocols (TP), as well as a rest-of-bus simulation, can only be implemented with enormous effort in a Simulink model.

Easy access to a network can be obtained, for example, by using CANoe a simulation and testing software tool from Vector. This software is typically used to simulate the ECU network with all its nodes (remaining bus simulation). The communication layers of each node perform network-specific tasks, such as sending with the correct send type in CAN or scheduling in FlexRay. Other protocols, such as NM or TP, can easily be added in the form of OEM-specific components. The application layer of a node, the behavior described by a MATLAB/Simulink model, is placed over a pure signal interface (**Figure 1**). There is no more reference to the specific network. The model only writes signal values to its output and reads signal values at its input. Whether the signal is defined in a

CAN message or in a LIN or FlexRay frame is irrelevant from the perspective of the model. This abstraction on the signal level lets users re-use their existing models practically unchanged. All that needs to be done is to map the model's input and output ports to the relevant signals on the network.

Not only can signals be exchanged between CANoe and the Simulink model; system variables can be exchanged as well. These are

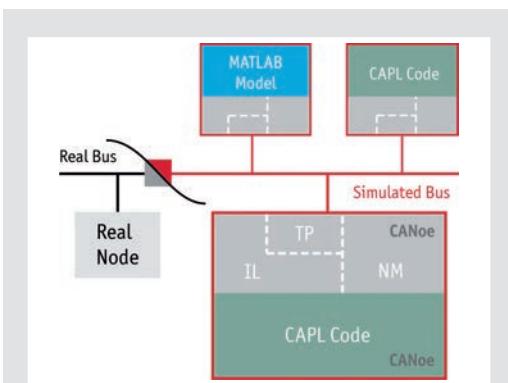


Figure 1:
CANoe rest-of-bus simulation with simulated nodes

internal CANoe variables that can either be explicitly created by the user or automatically generated. Automatic generation could be implemented by linking digital/analog hardware to CANoe. Here, each port of the D/A hardware is mapped to a dedicated system variable. This lets the model interact directly with the connected D/A hardware. After installing the CANoe MATLAB integration package from Vector, a Simulink Blockset is available with the relevant blocks for exchanging data with CANoe. These blocks primarily consist of functions for writing and reading bus signals, system variables and environment variables. In addition, they contain blocks for calling CAPL functions. It is also possible to react to calls of CAPL functions, e.g. to control triggered subsystems in Simulink.

Offline Mode for Rapid Prototyping



This mode should be selected for very early development phases, when frequent changes still need to be made to the model, and the real existing network does not play an important role yet. The application developer can still test model behavior in relation to a fully simulated network. A model developed in this early phase can also be re-used in all later phases without needing to make any further modifications to the network.

Within Offline mode the simulation itself is started in the Simulink environment. CANoe works in Slave mode, and Simulink is the master for the simulation session. The simulation is computed based on the specific computer speed. All Simulink debug features can be used here, and there is no need to connect real hardware to CANoe.

Synchronized Mode for Early Development Phases



This mode was also designed for early development phases, in which the model has not yet attained its finished state. Compared to Offline mode, the Synchronized mode offers the additional option of integrating real hardware. Here too, the simulation is started in Simulink. Unlike Offline mode, the time from CANoe serves as the basis for the simulation time here, and the simulation is computed in real time. One limitation worth mentioning is that the model may not be so complex that it is not possible to quickly compute it as real time. That is because adapting the "Simulink time" to "CANoe time" slows execution speed in Simulink and conforms it to CANoe's real time base. However, this slowing cannot be performed any longer if model computation requires too much effort. If the model does satisfy the condition mentioned above, full access to real hardware is assured in this operating mode. The debugging capabilities of Simulink can also be used. However, synchronization of the simulation time is lost whenever model execution is paused.

Online Mode or Hardware-In-The-Loop Mode



Model computation is executed entirely in the CANoe environment here. This involves generating a DLL from the Simulink model, which can be loaded and executed in CANoe. The Real-Time Workshop of MATLAB/Simulink controls code generation here. To generate code for CANoe, a special "Target makefile" was developed for the Real-Time Workshop environment that controls the generation process. The generated code is then compiled and linked using Microsoft Visual Studio compiler. This results in a Nodelayer DLL that implements a CANoe-internal API and can be added as a component to a node in the Simulation Setup. Other examples of such components are the OEM-specific Interaction Layer and components for network management or transport protocols. CANoe loads these components at the start of a measurement, and they are released at the end of the measurement. Therefore, when the model is recompiled, it is sufficient to end the measurement to integrate the changed components in the simulation.

Consequently, model execution takes place entirely within CANoe's real-time environment. All events, such as actions on the network, timers and user inputs, are computed to precise cycles. The model is part of the CANoe configuration, and it is easy to transfer to other parties. Later it will be shown how the model can still be parameterized at a later time, i.e. without re-compiling. First, we will present a more precise description of how the model is executed in CANoe.

Model Computation in CANoe

To understand model computation better, it is important to understand the execution model in CANoe. Essentially, CANoe computes in an event-driven way. In this context, events are incoming network messages or timers. With each incoming event, the simulation time in CANoe is set to the time stamp of the given event. The time base is derived from high-resolution, non-Windows timers at the network interfaces. This is a way to achieve a high level of precision of the time stamps, even though CANoe is running under Windows. If a generated MATLAB/Simulink model is now added to the configuration in the form of a DLL, it must be computed periodically. As mentioned above, timers are also events that can set the simulation time of CANoe. Therefore a timer is set in the model, with exactly the time specified in the Solver settings of the Simulink model.

Parameterizing the Model

Users often wish to modify the model afterwards, i.e. without recompiling. Use cases can be sub-classified as follows:

- > In the model, certain starting conditions are defined at the beginning of the simulation, which may not change the actual

model logic, but may be entirely different in different test scenarios. Example: It might be necessary to modify the gain factors in control loops to test different control characteristics.

- > The Real-Time Workshop and/or a Microsoft Compiler is unavailable, or the time required for frequent recompilation is too long, or the model file itself is unavailable. Here, users can still modify the models at runtime without a MATLAB/Simulink installation.

In such cases, it would be rather cumbersome to parameterize the model via separately generated DLLs – this would require continual, time-consuming reconfiguration of components at the nodes of the Simulation Setup. To properly handle these use cases, components generated for CANoe were made to be parameterizable after compiling. In code generation, system variables are generated for each parameter in the model, which CANoe then reads and writes. Model parameters are typically the properties of individual Simulink blocks. In the case of an Integrator block, this might be its initial state, or in the case of a Sinusoidal block its frequency, phase, offset or amplitude. A system variable, e.g. representing the gainfactor of a Simulink Gain block, can then be used in the analysis windows (Graphic, Data, Trace) as well as on panels, and in CAPL programs and tests. Another advantage is the ability to fine tune the model, since it is easy to vary internal parameters iteratively.

Viewing a Simulink Model in CANoe

MATLAB/Simulink models can be observed in CANoe, provided that they exist in the form of a generated component. A separate window is available for this at the specific node after it was configured accordingly (**Figure 4**). To modify the model at a later time, all generated internal model parameters may be moved to the CANoe analysis windows by drag-and-drop from the model display window. No MATLAB/Simulink license is required for this display.

Application example: Simulation of an active chassis control system with a FlexRay bus

The design of an active chassis control system, including a vehicle model, serves as a challenging example. The vehicle model should be able to simulate vehicle dynamics sufficiently realistically and serve as a platform for the chassis control system with active dampers. The goal here is to use the control system to attain the most comfortable driving behavior possible. The overall model is subdivided into two main blocks here:

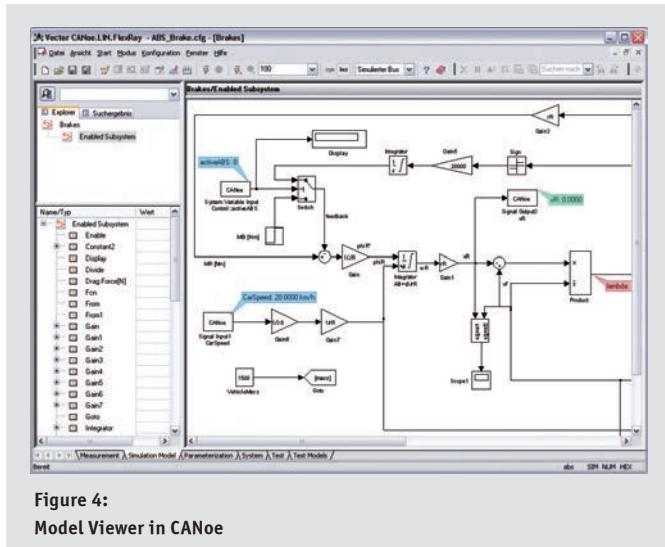


Figure 4:
Model Viewer in CANoe

- 1) Environment model (environment simulation for controllers), including:
 - > Multi-body model of the vehicle (mechanical vehicle model)
 - > Model of the powertrain (simplified engine model)
 - > Model of the brakes
 - > Four wheel models
 - > Road model
- 2) Chassis control system, including:
 - > Vehicle observer
 - > Main chassis controller
 - > Four subordinate power controllers for the active dampers

The environment model is subdivided into the vehicle model and the road model. The vehicle model is designed as a multi-body model with 15 degrees of freedom. The multi-body model is defined symbolically with the help of a computer algebra system, and motion equations are derived from this model (approx. 4,400 additions and approx. 20,800 multiplications). The vehicle model offers additional inputs for stimulation: Accelerator and brake pedal positions, engaged gear, steering angle, parking brake state and control target value (comfortable or sporty). These parameters are passed by system variables in CANoe. Driver inputs may be interactively set on a control panel with relevant controls. Suitable macro recordings may also be used as driving profiles for automated playback. The road model is modeled by a lookup table containing the elevation of the road surface and surface normals. Similarly, the surface composition is described by a friction coefficient at all road locations.

Chassis control consists of a LQ controller with observer. The controller computes target forces for the active force elements in the

wheel suspensions. Essentially, two control system goals are set here:

- > Reduction in body accelerations. This results in greater comfort for the car driver
- > Maintenance of constant wheel contact zone (minimizing variations in wheel contact forces). Results for the car driver: Improved safety and vehicle dynamics.

The observer reconstructs the non-measurable vehicle states by applying a simplified linear vehicle model (with seven degrees of freedom). The chassis control system needs measurement parameter values of the angular accelerations of the car body's roll and pitch from two gyro sensors, as well as the car body's lifting acceleration from an accelerometer. The vehicle simulation makes these parameters available, and they are also passed to the chassis controller via system variables in CANoe. The force controller regulates forces of the active force elements, targeting values specified by the chassis control system. This might be designed as a simple PI control system for current regulation, for example.

In CANoe, six simulation nodes are defined (**Figure 5**): These are the "Environment simulation" node (Vehicle model), the "Chassis controller" node and four "Force control" nodes. The six nodes exchange data via a FlexRay bus on the four target forces for the force actuators and four deflections of the wheel suspensions. Periodic transmission of these signals over a bus represents discrete sampling with a dead time in the feedback control loop. Due to its indeterminate nature and its size, this often has a negative effect on control quality. Nonetheless, these signals can be transmitted

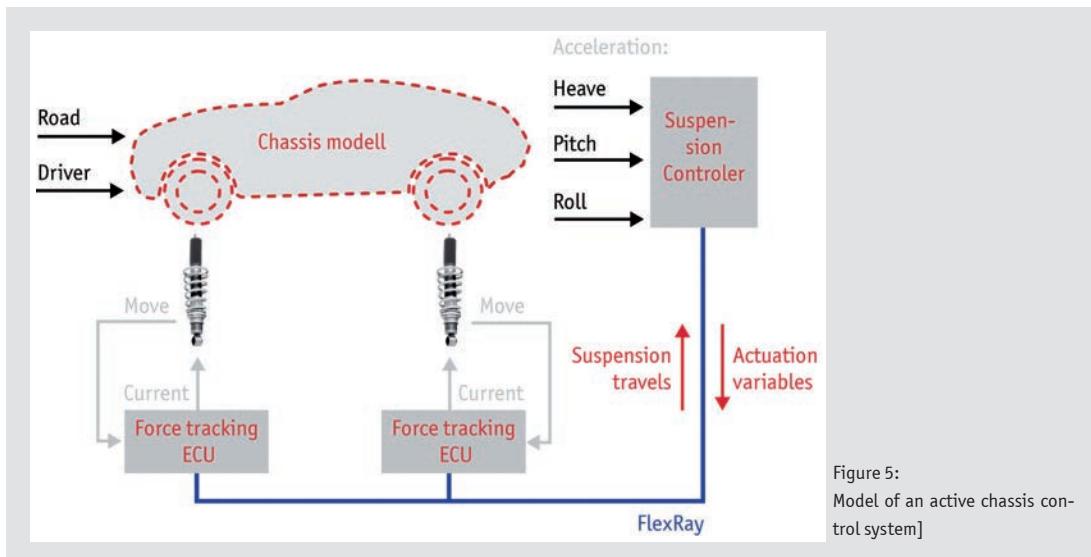
very reliably and without jitter using a FlexRay bus. It is also possible to transmit these signals at a high cycle rate (2.5 to 5 ms). This significantly improves control quality of the overall control system compared to a CAN bus.

Concluding Remarks

CANoe/Matlab integration enables simultaneous use of Simulink to model complex application behavior and integrate the relevant in-vehicle network within the same development process. Users can work in the familiar MATLAB/Simulink environment during development and do not need to concern themselves with network-specific details.

Solver

A Solver specifies the mathematical approximation method used to compute time-dependent variables in a model. Simulink provides Solver algorithms with variable step-width or fixed step-width. Solvers with variable step-width optimize the algorithm according to how quickly the data changes in the model. In an extreme case, if the Solver finds discontinuities then certain simulation time points are re-computed with a smaller step-width. On the other hand, large step-widths are used for times over which the model data does not vary greatly; this accelerates the simulation time. A Solver with a fixed step-width always computes with the specified step-width value. This type of Solver does not detect discontinuities.



Selecting a Solver

Due to the interface to CANoe, and in consideration of later code generation, the user should choose Solvers with a fixed step-width, because Solvers with variable step-width exhibit the following limitations:

- > Solvers with variable step-width cannot be used for code generation. With these Solvers, it is impossible to predict when the next simulation step needs to be computed and how long it will take. This fact violates the real-time behavior that is sought on typical target platforms. This limitation applies in general and for all target platforms.
- > Since the Solver has no knowledge of the change in data at input and output blocks – unless they are non-deterministic bus signals – a Solver cannot optimize with variable step-width here.

Three different operating modes are available for code generation with MATLAB integration in CANoe: In "Offline mode" and "Synchronized mode" both software tools are active; the simulation is started from Simulink. In "Online mode" or "HIL mode", however, code for CANoe is generated from the model.

Blockset

The individual blocks of the blockset are implemented as subsystems. They consist of a so-called S-Function, which implements the specific functionality. For the most part, an S-Function is user-specific code that implements an API provided by MATLAB/Simulink. This makes it rather easy for developers to extend MATLAB/Simulink functional features user-specifically.



Jochen Neuffer

studied Information Technology at the University of Applied Science in Esslingen. Since 2002, he has been working at Vector Informatik GmbH where he is a Product Management Engineer in the Tools for Networks and Distributed Systems area.



Carsten Böke

studied computer science at the University of Paderborn. From 1995 to 2004 he was a scientific assistant at the Heinz Nixdorf Institute, working in the Parallel Systems Design area. Since 2004, he has been working as a Senior Software Development Engineer at Vector Informatik GmbH where he develops tools for bus analysis and bus simulation of FlexRay systems.

**Translation of a German publication in Elektronik automotive,
March/2010**

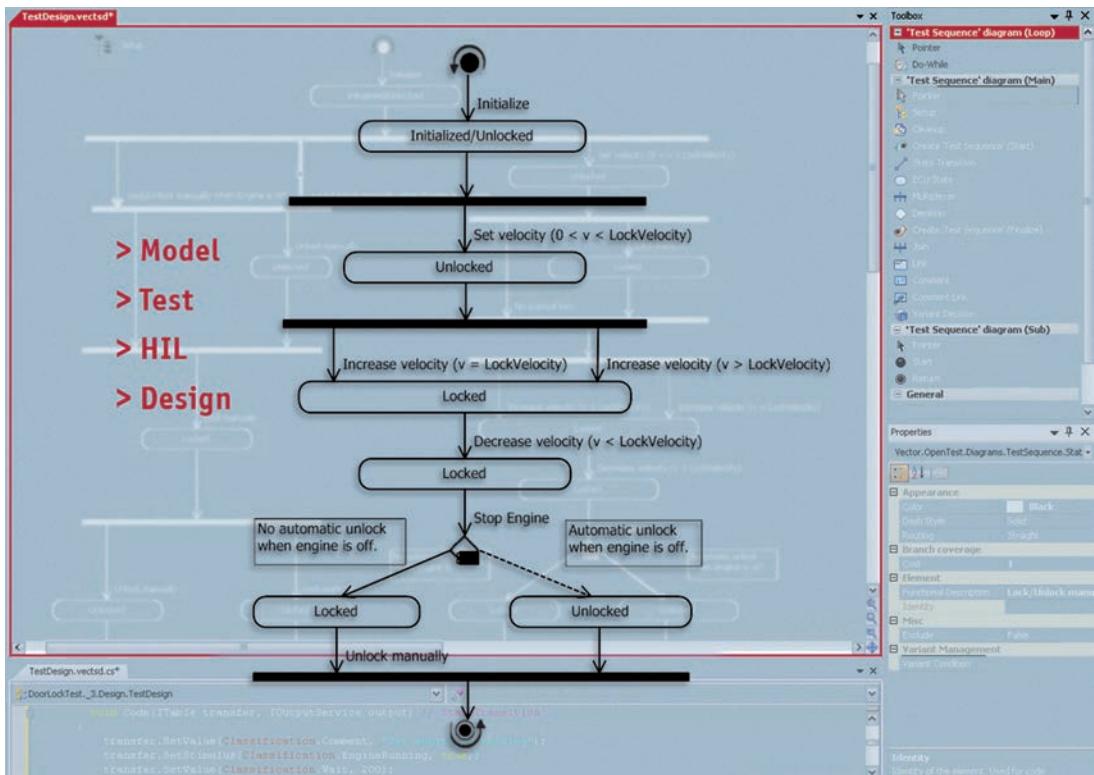
Links:

Homepage Vector: www.vector.com

Product Information CANoe Matlab Integration: www.vector.com/vi_canoe_simulation_en.html

Model-based Design Simplifies ECU Tests

Mastering Variation



HIL systems are playing an ever increasing role in automated ECU validation and testing. Structured test creation methods, standardized by the use of models, improves both test efficiency and quality. The OpenTest tool from Vector provides test engineers with the necessary infrastructure to develop model-based test scenarios. Tasks may be distributed among several specialists using a uniform system of graphic test notation, independent of the specific HIL test bench or test automation system.

ECU tests are most often created at the end of the module's development cycle. As such, they must be created efficiently and modified in a timely manner to fit new and changing test requirements. Such a challenging environment requires a structured approach to test creation. An increasing amount of tests is performed automated on Hardware-In-The-Loop (HIL) test benches. Each HIL system has its own test description method ranging from general programming languages to proprietary scripting languages and graphic notation. For complex systems tests have to be performed on different test benches depending on the test requirements. This fact reinforces the need for a test creation methodology that offers

a uniform test description format independent of the HIL system used.

Structured Test Creation Process

Typically, tests are developed in the context of larger test teams. These teams may work directly in software development, or they may represent independent organizational departments. Along with the actual testers - who execute, monitor and report test results via a test automation system on the HIL test bench - the roles of the test designer, HIL programmer and test framework

specialist are gaining in importance. Today, test designers create the test descriptions in HIL-system-specific description formats, or scripting languages, such as TCL, Python or CAPL. This means however, that all test designers need to master all of the scripting languages used. For efficiency reasons, it makes sense to formulate test designs independent of the test system and then have specialized programmers write the software interfaces just once. This means that only one specialized programmer is needed for each supported HIL system who adapts the test descriptions to the specific system.

ECU development is increasingly being conducted in the framework of what are referred to as software product lines. A software product line is a set of ECUs that are all based on a common set of re-usable functions. The individual implementations, or ECU variants, consist of both common functions and variant-specific functions. In testing software systems, common components should be identified and be made available to all designers in the form of a test framework.

OpenTest Methodology

OpenTest is a methodology and framework from Vector that supports structured test creation. Its test design notation is independent of the specific HIL system thus enabling the distribution of user roles. The executable scripts used on the test benches are automatically generated by HIL-specific generators (**Figure 1**).

The requirements represent the starting point in the test creation process. Frequently, these requirements only exist in text format. The first task is to derive the individual test cases for the system under test (SUT) from these requirements. Each requirement must be covered by one or more test cases. In OpenTest, this

is accomplished by a graphical notation that is used for modeling test sequences and for reviews.

In a structured test creation process, one or more specialists create a basic framework for one or more systems under test. This framework consists of these re-usable components:

- > A formalized description of input and output variables to stimulate and test the expected reactions of the SUT
- > Functions for basic test steps or entire test sequences
- > Cross-platform and variant-specific parameter sets

This cross-variant basic framework is provided to individual test designers in the form of a library. Test creation that is based on the basic framework increasingly leads to test designs with the same type of structure. This increases the share of re-usable individual designs, and it makes them more comprehensible for test reviews. OpenTest takes test coverage strategies such as branch and path coverage into consideration in automatically generating the test cases and HIL system specific test scripts from the test models.

OpenTest Notation

Tests are modeled using UML state diagrams (**Figure 2**). In OpenTest the UML notation is enlarged by a set of elements for describing tests. Thus it provides a graphic form of expression that is specialized for ECU testing, a so-called Domain Specific Language.

A state in the diagram represents a state of the ECU function to be tested. Each transition between states defines events that stimulate the SUT and elicit defined reactions. The individual test cases are defined with the help of multiplexer elements. The test cases can be specialized at any desired point in the test flow. The advantage of this method is that shared test steps and sub-sequences only need to be modeled once for multiple test cases.

Each path taken from the start node to the end node describes an independent test case. The sum of test cases should, at a minimum, cover the requirements to be tested. Also modeled are test cases that are not explicitly based on requirements in the Requirements Specification. Instead, they are based on the test designers' experience or the results of previously conducted tests. Explicit modeling of the tests makes this possible and further enhances testing depth.

The concept of software product lines is supported by variant branches

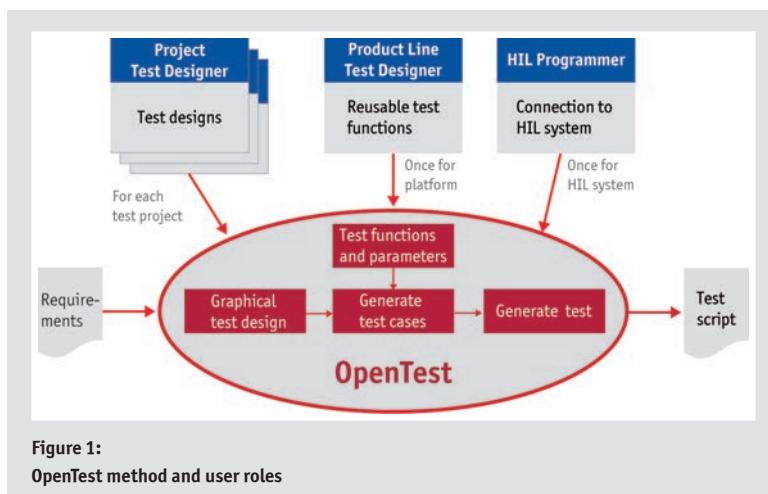


Figure 1:
OpenTest method and user roles

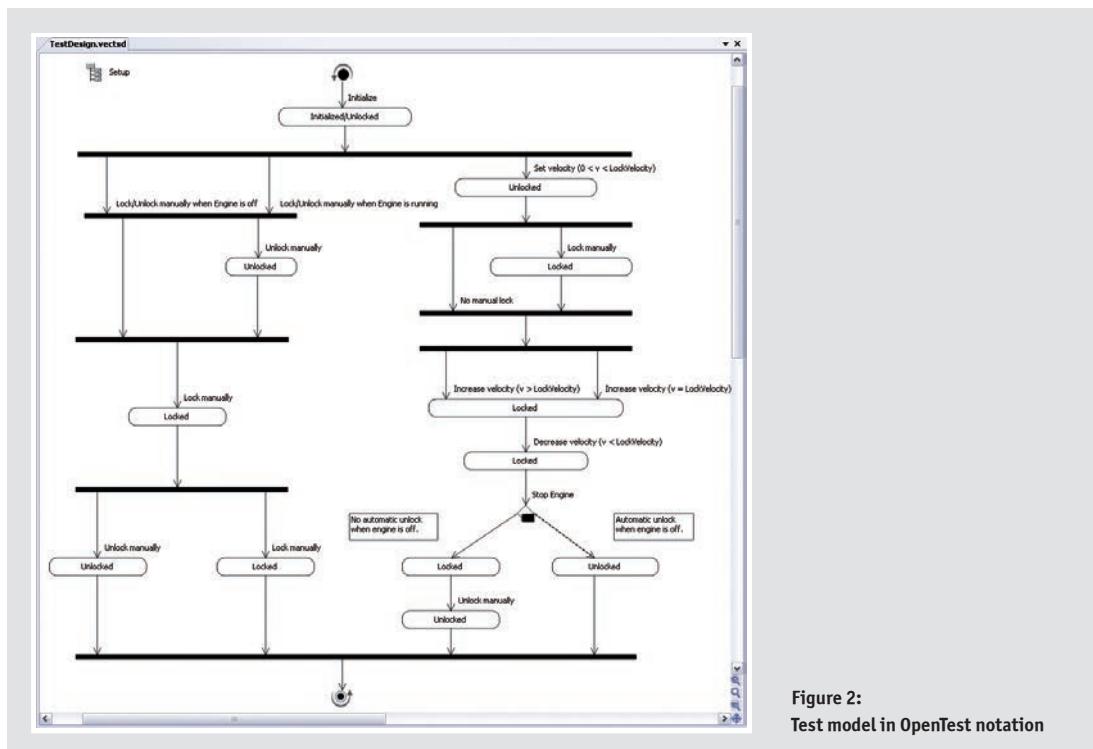


Figure 2:
Test model in OpenTest notation

in OpenTest notation. Variant branching adds specialized subsequences for the individual variants within the software product line. Cross-variant test steps apply to all software product line variants and are modeled jointly within the test sequences.

OpenTest modeling enables structured and well-organized test descriptions. This is necessary since the test model is not only used to generate test cases and HIL programs, it is also used for reviews. Communication with customers, software developers and quality managers is much easier on the graphical notation level than on sequential or text-based test scripting level. This approach enables graphic highlighting of failed test cases, critical test steps or test cases for regression testing. Furthermore, each test step may be documented with comments.

Independence from the HIL System

The models that test designers create in OpenTest are independent of the specific HIL system used. HIL-specific executable scripts are automatically generated. The programmers familiar with the specific system define mapping rules which convert the individual test steps to executable commands. These commands range from simple value stimuli or results comparisons to complex commands

for system diagnostics or extraction of internal ECU parameters and system states over XCP.

OpenTest currently supports such test systems as CANoe from Vector and TestStand from National Instruments. Its framework character makes it easy to integrate other HIL systems as well (Figure 3).

Practical Use

The methodology presented here and the OpenTest tools was initially developed for testing mechatronic systems by Robert Bosch GmbH, where it has been in operational use since 2006. The concept of the software product line is especially well developed in this area and leads to an intensified role for the test framework specialist. Efficiency in creating tests was significantly improved by applying a uniform methodology and implementing it with OpenTest as the software tool. Distribution of tasks within the test team – as designer, HIL programmer and framework specialist – also makes test engineers experts in specific work areas and improves identification with their respective tasks.

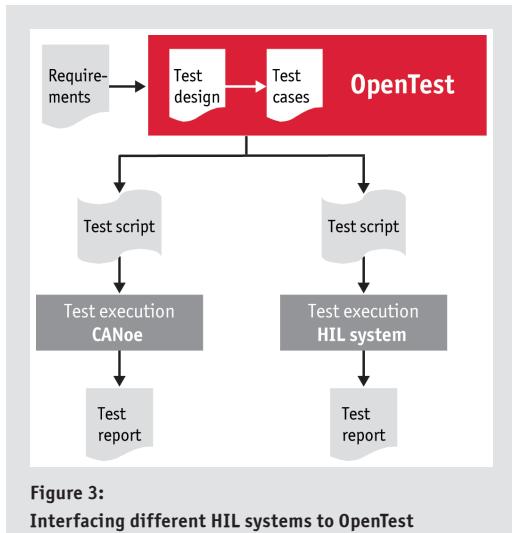


Figure 3:
Interfacing different HIL systems to OpenTest

Translation of a German publication in Automobil Elektronik,
April/2010

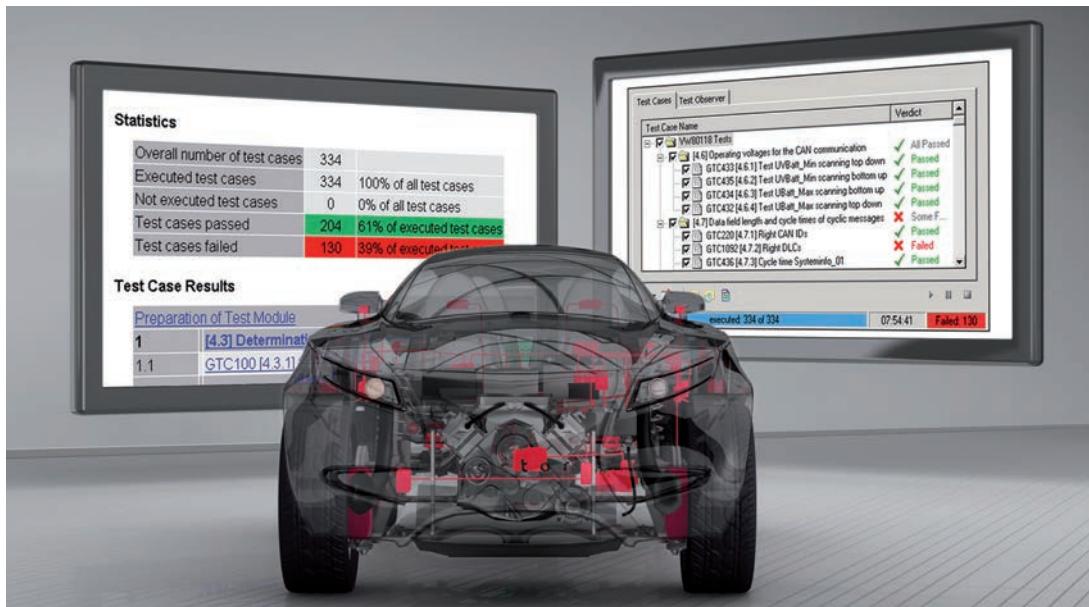
All Figures:
Vector Informatik GmbH

Links:
Homepage Vector: www.vector.com



Comprehensive Communication Tests for ECUs Developed at Volkswagen Group

Identical test environment for both OEM and suppliers



OEM-specific test implementations play an important role for suppliers in the ECU development process. They are used to check the network conformance, assure error-free communication between the numerous ECUs and are important criteria for the final acceptance from the automotive OEM. Suppliers of the Volkswagen Group (VAG) can now minimize both ECU development times and costs using a VAG-specific test software to automatically generate high-speed CAN tests according to the VW80118 specification.

ECUs not only need to fulfill their required functionality, but must also fit seamlessly into the ECU environment – while considering special OEM-specifics. This is why intensive communication tests are needed in addition to the functional tests. Such tests investigate the ECU's behavior, not only under normal conditions, but in numerous error situations as well, for example: drops in supply voltage, reception of faulty messages, protocol violations during transmission, irregularities in cycle times and disturbances in bus voltage level due to short circuits.

Reproducible testing at the supplier

The supplier assumes primary responsibility for correct and reproducible behavior of the Device Under Test (DUT) under all of these constraints. The costs for the implementation of the tests are not

insignificant considering the growing complexity of automotive electronics. Test applications must be created and maintained, or else a testing facility must be hired. Finally, each new ECU and each modification requires making extensive changes to the created tests. Problematic with this process is that some details of each party's test implementation – whether supplier, testing facility or OEM – may deviate to a greater or lesser degree. If tests yield different results in reference to the latest test specification, time-consuming troubleshooting will follow. Furthermore, it is then necessary to address the unpleasant question of who is responsible for the caused problem. Errors do not necessarily have to be caused by the ECU itself; gaps are also possible in the specification itself or its interpretation. The conventional wisdom of detecting errors as early as possible to avoid the much higher costs of correcting them later, is proved once again.

Testing at the push of a button

The VAG extension for the CANoe test and simulation system from Vector demonstrates that cost reduction and enhanced ECU quality do not need to contradict themselves. The CANoe Test Package VAG offers an incomparably quicker path to these desired goals for VAG developers, test houses and of course also in-house OEM departments. PC-based test configurations can be generated for the automatic execution of high-speed CAN conformance tests at the push of a button and without further preparation. Version 2.0 of the Test Package, developed in cooperation with Volkswagen in Wolfsburg, Germany, conforms to the latest test specifications released by the Volkswagen Group. It covers all VW80118 test cases, which are mandatory for VAG suppliers. The package supports both the current and previous versions of the VW80118 test specification.

The CANoe Test Package VAG consists of a test configuration generator with configuration dialog, CAPL (Communication Access Programming Language) test libraries for UDS (Unified Diagnostic Services) and KWP (Key Word protocol) as well as the VAG Interaction Layer for the rest-of-bus simulation (**Figure 1**). So, CANoe is not only responsible for the actual test sequences; it also implements the necessary rest-of-bus simulation using the provided VAG Add-Ons. Such OEM-specific extensions are generally available free-of-charge and consist of a simulation generator, Interaction Layer (IL) and Network Management (NM). To achieve full automation, the Test Package utilizes the VH1100 to supply voltage to the ECU and the CANstress disturbance module (**Figure 2**).

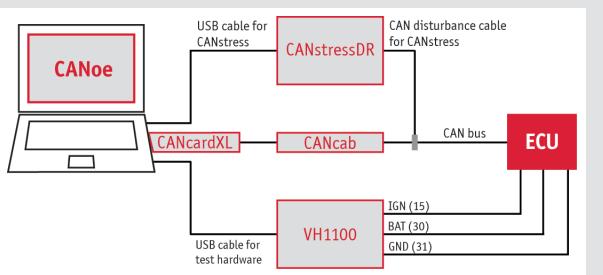


Figure 2:

Time-synchronous real-time acquisition and visualization of internal ECU signals via CCP/XCP, and of signals from CAN, LIN and FlexRay buses and external measurement equipment via CANape.

Automatic generation of disturbances and supply voltage control

The VH1100 is a controllable voltage supply used by the test system to simulate various supply situations for the DUT. It has relays for independently switching terminals ignition (15), battery (30) and ground (31), and can make precise measurements of the DUT's current consumption. Various voltage patterns and disturbances may be simulated by remote control via USB, to examine DUT behavior in reaction to overvoltages, undervoltages and voltage interruptions. The hardware module CANstress is used to simulate digital and analog disturbances on the CAN bus. This USB-hardware is connected directly to the bus line and is used to reproducibly manipulate physical properties and logical levels. Flexible trigger and disturbance logic lets users destroy any desired bit positions of CAN messages and manipulate bit fields. Full remote control capability via COM (Component Object Model) makes a valuable contribution toward automating VAG tests.

OEM delivers test base to suppliers

One of the key requirements for successful configuration of the test environment is a consistent description of ECU communication in the form of the TBD (Test Basis Documentation) file and the CAN network database (DBC). The OEM creates these two files and passes them on to the supplier. While the DBC file stores OEM-specific information on the network and rest-of-bus simulation, the TBD file contains detailed information on the ECU. A new file must be created for each ECU, which provides information about the ECU in XML

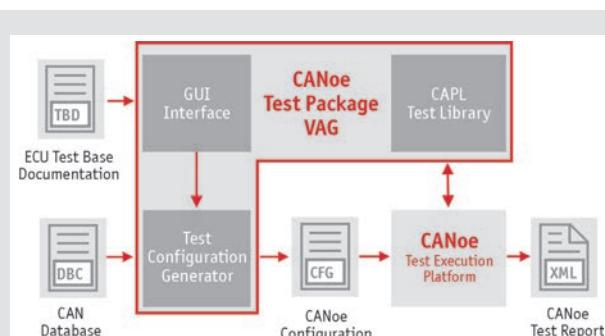


Figure 1:

At the push of a button, the CANoe Test Package VAG creates an entire test configuration, starts test execution and generates a report.

format. This information includes a list of the Tx and Rx messages, diagnostic parameters, Diagnostic Trouble Codes (DTCs), specification versions, etc. The TBD file supplies information about a VAG ECU that is needed to properly configure VW80118 tests. The supplier may also add information to this file using an editor provided by VAG, e.g. to add DTCs. The test configuration generator reads the TBD and DBC files and uses them to generate the CANoe test configuration (**Figure 2**), which consists of the remaining bus simulation and an XML test flow module.

Clearly represented HTML test reports

Up to several hundred test cases may be generated, depending on the number of Tx and Rx messages (**Figure 3**). Each test case returns a detailed HTML test report (**Figure 4**) and log files in ASCII format. The test reports are organized with section numbers corresponding to those of the VAG test specification. Color highlighting of errors gives users a quick overview of the success or failure of executed test cases.

Summary and Outlook

The CANoe Test Package VAG – together with the VH1100 test hardware and the CANstress module – creates a cost-effective test environment based on standard CAN tools. The described solution lets developers, suppliers and test houses perform the same tests as the OEM with minimal effort. The automated generation of configurations and test sequences saves time and reduces the development costs. At the same time, OEMs and suppliers benefit from early error

5.3 GTC436 [4.7.3] Cycle time Systeminfo_01: Passed

Test case begin: 2010-06-28 16:45:57 (logging timestamp 5306.402409)
Test case end: 2010-06-28 16:47:33 (logging timestamp 5401.687996)

Test Parameter

msgId:	0x123
cycleTime:	1000
testWithDiagnosticCommunication:	0
voltageDisturbance:	0
preliminaryDiagnostic:	0
preliminaryTrigger:	NONE

Test Case Check Statistics

Check/MsgCycleTime (5.3#1): Check Passed

Parameters	
Type of check	MsgCycleTime
Message	CAN message 'Systeminfo_01' ID = 291 (0x123) on bus CAN
MinTime	900 ms
MaxTime	1100 ms
Statistics	
Runtime of the statistic	90000 ms
Number of probes	91
Message ID	291 (0x123)
Minimal measured cycle time	999.998375 ms
Maximal measured cycle time	1000.001125 ms
Average cycle time	999.9996 ms
Failure ratio (in %)	0

Figure 4:

Detailed report example for a successful test case

detection, fewer iterations and improved quality. The CANoe Test Package VAG is the only test system available on the market that is certified by VW for high-speed CAN conformance tests and is explicitly recommended for in-house testing at suppliers. An extension of the existing solution by adding globally applicable VAG test specifications – such as for network management messages (NM-High) – is already planned for the next version.

Translation of a German publication in Hanser automotive, October/2010

Figures:

Vector Informatik GmbH

Links:

Homepage Vector: www.vector.com

Product Information CANoe Test Package VAG:
www.vector.com/canoe_tp_vag

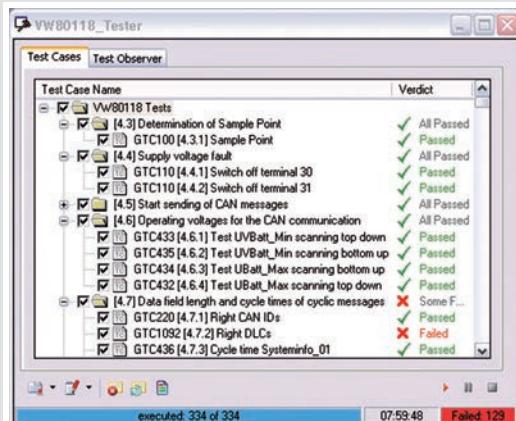


Figure 3:
Test execution control of the VW80118 test



Klaus Theobald

is Senior Software Development Engineer
for the "Tools for Networks and Distributed
Systems" product line at Vector Informatik
and has developed the VAG Test Package.



Gavin C. Rogers

is Team Manager in the "Tools for Networks
and Distributed Systems" product line and
Product Manager for CANoe Test Packages.

Test Bench for Complex ECU Networks

Individuality of omnibus features requires a flexible test system



ECU tests may be very complex depending on the work area and the range of functional features. This is even more relevant when testing must cover a network of five to nine ECUs whose functionalities are inseparably intertwined and vary according to the vehicle type, features and special customer wishes. That is why EvoBus has started up a new component test bench based on the VT System from Vector. Its flexible, automated tests enable a high level of test coverage and precise reproducibility of all test cases.

In the omnibus area, the challenge is to create a vehicle that fulfills the technical requirements of commercial vehicle while providing a level of comfort and convenience at least equal to that of a passenger car. The focus here is on the development of platform systems that implement up to 120 functions on a common hardware unit. Their complexity ranges from "very simple" to "very complicated." The many possible combinations of individual functions are another requirement. A high level of customization and consideration of specific special customer wishes, sometimes even

About EvoBus

The EvoBus company and the two omnibus brands Mercedes-Benz and Setra have been merged under one umbrella company since 1995. EvoBus is a subsidiary of Daimler AG, which is active in the omnibus business area. With a market share of 50% in Germany and 21% in Europe, Daimler is the market leader in the segment of buses over 8 metric tons. [Revised: 12-31-2011]

shortly before delivery, require special modularity and flexibility of the architecture in the body and convenience electronics areas. Very low volumes in comparison to the passenger car industry makes re-use desirable in all model series (urban, long-distance and tour buses). Synergy effects are exploited and costs are reduced by using standardized adaptable hardware.

In some respects, the E/E requirements of omnibuses differ significantly from those in the passenger car area, because the features of an omnibus are determined by individual customer wishes to a far great extent. This must be considered in the E/E concept, which requires especially well thought-out modularity and flexibility. The electronics (hardware) should lend itself to broad-based use within the overall range of omnibus products as much as possible and be adaptable in a cost-effective way. Other challenges include improving fuel economy, reducing exhaust emissions and performing advanced development of active and passive safety and assistance systems.

To flexibly cover all of these requirements in the E/E concept, EvoBus developed a scalable multiplex system (MUX system). Consisting of up to nine modules, the electronics architecture of this MUX system is a type of distributed system within a distributed system. The hardware and firmware of the MUX system are a sort of middleware that provides the fundamental components and tools for application development. This middleware is user-programmable by the use of IEC61131-conformant and OEM-specific logic chips, and it is positioned beneath the application level. The individual MUX modules are distributed to different bus branches of the overall vehicle network that consists of five main CAN buses for powertrain, chassis, interior, telematics and diagnostics areas as well as numerous LIN subnets.

Testing challenge: ECU network of up to nine modules

The scope of testing is exceptionally high, due to the high functional density and numerous degrees of freedom in configuring the MUX system. To optimize the time-intensive and complex test phase, EvoBus 2011 started up a new test bench that is individually customized for the needs of the MUX system. The test bench is based on the VT System from Vector. It is capable of simulating all components and system states of the MUX environment that are required for the tests. They include both the signals of the numerous hardware inputs and outputs as well as the CAN and LIN communication. Serving as a user interface is a PC with CANoe test and simulation software from Vector.

After tests by the supplier, component tests represent the first step in the test process chain within the V-model, which ranges from component testing to software module tests, Hardware-in-the-Loop and finally trials in the real vehicle. In component tests,

the focus is on correct functionality of the middleware, while the application is considered from a later milestone in the testing process. The MUX system itself is a scalable peer-to-peer system. The number of MUX modules used in the vehicle depends on the selected equipment versions and special customer wishes. Each MUX module has a number of largely configurable digital and analog inputs and outputs. The MUX modules are interconnected via the CAN buses to form an overall system (**Figure 1**).

With the help of the VT System, EvoBus engineers test the fundamentals of extensive functions such as valve control for automatic suspension leveling, door control ECUs for the luggage compartment and engine compartment doors and the complex interior lighting system. The latter also leaves much freedom for fulfilling customer-specific wishes. A key challenge is network management. Status transitions such as wake-up from the power-down mode, and in the opposite direction, shut-down of the network must run trouble-free and in correct interplay with the application and hardware. Besides simulating the CAN communication of the remaining bus, OSEK network management is also simulated here. Since each MUX module always needs to be informed of the current system states of other MUX devices in a timely way, synchronization processes are continually taking place. In the exchange of process images, specified timing and real-time requirements must be fulfilled, and this represents another test criterion.

Simulating environments and automating tests

The MUX system must also be tested to determine whether it can operate smoothly with various hardware changes. Test contents include data and diagnostic routing of the Sub-CAN and Sub-LIN buses of the MUX system. Testing of diagnostic functions

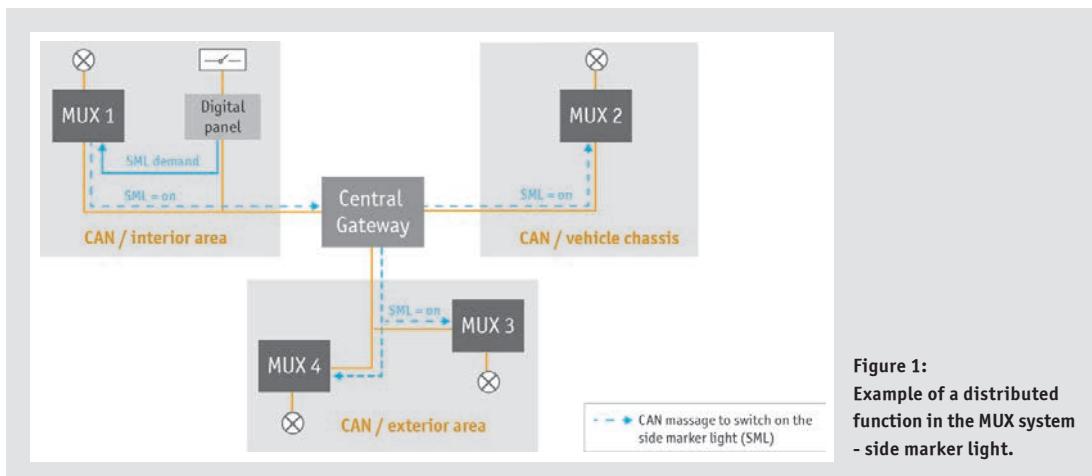


Figure 1:
Example of a distributed function in the MUX system - side marker light.

requires special attention here: The I/O channels of the MUX system permit flexible and varied configuration of diagnostic behavior. As a result, the number of diagnostic test cases is large. Testing the trouble codes of the MUX system itself currently requires implementation of 1,000 of the most important test cases. However, system variability can require more than ten times the number of permutations. This task cannot be performed manually and still be economical. The VT System now enables automated execution and documentation of this test within a few hours.

The complexity and diversity of the MUX tests can only be economically handled by systematic automation of test sequences. The VT System, which is modularly built in 19" industrial rack format, is optimized for the characteristic tasks of test automation

and, with six modules, it permits configurations of the compact bench-top device that range to a large HiL system for the test laboratory (**Figure 2**). The VT System implements comprehensive emulation of the ECU environment with regard to input and output circuitry as well as communication of the connected CAN and LIN buses.

The main advantage of emulation is that the behavior of drivers, passengers, vehicle equipment and other environmental components can be simulated reproducibly. Testing of door control, for example, requires repeated actuation of a pushbutton to open the omnibus door at precisely defined times. Instead of manual actuation, these actions are performed by the test system. This assures the proper timing sequences and is reproducible at any time.

Scalable test system enables customized hardware layout

The system supports all commonly used analog and digital input and output standards including more complex functions such as generating and processing PWM signals and determining effective values. While load and measurement modules of the VT1004 type are connected to the outputs of an ECU, the VT2516 and VT2004 modules contain the electronics for stimulation of digital and analog inputs. VT7001 power supply modules regulate the operating voltages and measure the current consumption of the MUX modules. In addition, VT6104 modules are used for the network communication of the CAN and LIN networks. The VT6050 PC module is the CANoe Realtime-PC, on which the simulations and tests are executed. Only the input/output level of the EvoBus test bench – which is built in a 19" control cabinet – consists of five logical levels that are each fully populated with twelve VT System modules. Currently, the system covers requirements for five of the nine possible MUX modules. The control cabinet also has a patch-board that provides individual access to all input pins of the MUX ECUs for manual measurements; break-out contacts for all CAN/LIN buses are also provided.

Comprehensive ECU tests generally also include test sequences in which conditions prevail that deviate from normal operation. That is why the VT System was designed to generate defined errors in the ECU environment, e.g. defective sensors at the inputs or atypical load behavior of actuators connected to the outputs. Upon request, the system can generate line breaks and short circuits in feed lines, short circuits to ground and battery potential or over-voltages and under-voltages. The five electronic loads represent a special feature. They can each conduct a current of ten amperes as a source or sink, so that the MUX system can be supplied with enough power for special test cases.



Figure 2:
VT System as component test bench of the Evobus MUX system.

Real-time capability by modular system layout

The remaining bus simulation utilizes the OEM package for the Daimler-specific interaction layer and OSEK network management. This assures full emulation of the sending behavior as well as realistic communication behavior on the data buses with low effort. Real-time relevant tasks, and the remaining bus simulation test sequences, are executed directly on the VT6050 real-time module. In turn, the VT6050 is connected via Ethernet to the CANoe host-PC that is used for user interactions and for viewing and analysis. This distribution of work makes an important contribution towards attaining the extraordinarily high scalability of the overall test system (**Figure 3**).

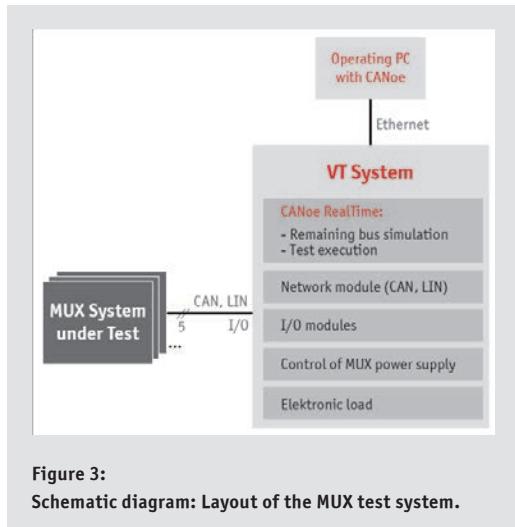


Figure 3:
Schematic diagram: Layout of the MUX test system.

Standard tool CANoe as front-end for test bench

CANoe on the desktop PC serves as a platform for all user operations, test definitions and evaluation actions. The dual monitor setup at EvoBus offers sufficient space for a well-organized display of the main program windows, output pins, input pins, measurement value displays, etc. CANoe has become established as an industry standard for electronics development related to automotive applications, and it offers numerous functions from which the MUX modules benefit. The intuitive user interface has made training in user operation of the test bench run quickly and smoothly. All parameters of the VT System are accessible from CANoe. Vector's 'Test Feature Set' meets the criteria for high-performance test automation. In addition, test sequences can be defined, tests executed and reports generated. In generating and executing reproducible test cases for the diagnostic protocol, CANoe Option

DiVa (Diagnostic Integration and Validation Assistant) performs valuable services. The Test Automation Editor contains an interface to a DOORS database in which the test specification is stored. After test execution on the VT System, the results are documented back into the database in XML format and saved. This assures traceability of abnormal findings, their correction and statistical information about the test object for every sample level. Universality of the tool chain and minimizing the number of interfaces were important to EvoBus. These requirements are fulfilled by the Vector approach, from definition of test requirements to test execution and evaluation of the reports.

In testing the MUX ECU network, the efficiency gains realized by the VT System are enormous. Extensive tests, which required about two weeks before – including preparation of the specific test software – can now be completed in one day. The pins of the ECUs were previously fed into specially fabricated purely passive test boxes that had to be manually rewired for each individual test. Since this approach could not be automated, it offered far few testing options and was inherently more susceptible to errors. When tests needed to be repeated, personnel resources were tied up for longer periods of time. With the VT System, on the other hand, it is possible to reproduce test runs at the press of a button. Today, using the new system, EvoBus has now performed 500 individual tests (without diagnostics) with greater test depth, test coverage and precision, where previously only about 100 spot check type tests could be performed. Another positive aspect of the test system is its tremendous flexibility. The VT test bench is not only used for automated testing of new software versions. It also serves as an analysis tool for feedback on abnormal findings to Customer Service and Production. Such situations require quick reaction. Therefore, the hardware can also be run manually over a CANoe user interface without having to write test scripts; in this way, it can be spontaneously used to troubleshoot problems and correct them.

Further extension of the EvoBus test bench planned

The VT System at EvoBus with its fully featured, approximately two meter tall control cabinet is one of the larger VT System projects implemented by Vector. The project timeframe from the specification phase to startup and training amounted to about five months. The progressive development of new vehicle types and the introduction of hybrid drives will make an extension of the test system necessary. EvoBus plans to extend the test bench by adding an input/output module. The test bench will then be able to test MUX systems with up to six modules. It will then be necessary to supplement the system with a second control cabinet. This represents an intermediate step towards full implementation of a system that can handle nine modules. That is because the functional features of large tour buses can assume considerable dimensions, where

comprehensive testing is always required for the electronics of comfort/convenience functions, climate control, interior lighting, distributed functions for hybrid vehicles, fast door opening and closing mechanisms, entertainment systems and other features.

**Translation of a German publication in Elektronik automotive,
October 2012**

Links:

Homepage EvoBus:
www.evobus.com

Homepage Vector:
www.vector.com

Product information VT System:
www.vector.com/vt-system

Product information CANoe:
www.vector.com/canoe



Michael Schneider, EvoBus

He earned a degree in General Electrical Engineering from the Ruhr University at Bochum and RWTH Aachen, Germany. Since 1998, he has been employed as a system developer in the Basic Systems area of DaimlerBuses.



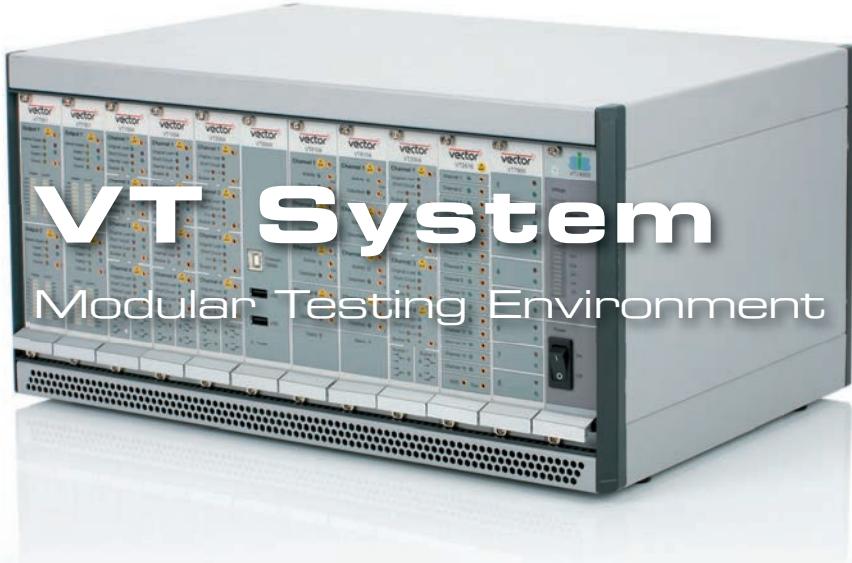
Philipp Merkle, EvoBus

He earned an engineering degree in Industrial Electronics specializing in Vehicle Electronics from the Technical College in Ulm, Germany. Since 2007, he has been working as a test engineer for Central Electronics at DaimlerBuses.



Katja Hahmann, Vector

Earned a degree in Electrical Engineering from the Technical University in Chemnitz, Germany. In 1997, she joined Vector Informatik in Stuttgart where she is group leader in CANoe application development for the Networks and Distributed Systems product line.



The Efficient Way to Test Your ECUs



Simplify your ECU testing by implementing a high-performance test environment with the VT System and CANoe. Benefit from flexible I/O modules and Vector's wide experience.



- > Simplify your test setup with minimal wiring effort, even for HIL systems
- > Simulate loads, sensors and short circuits in ECU tests and check the ECU's reactions
- > Generate efficient test programs using integrated test functions



Improve your test environment with the compact high-performance VT System from Vector.



► Get inspired: www.vector.com/vt



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

(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



Figure 1:
All-terrain mobility is the primary feature of the Unimog

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 re-used 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.



Figure 2:
Tire pressure control while driving is possible, e.g. to avoid site damage

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 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



Figure 4:
Modular setup of the VT System as a component HIL system

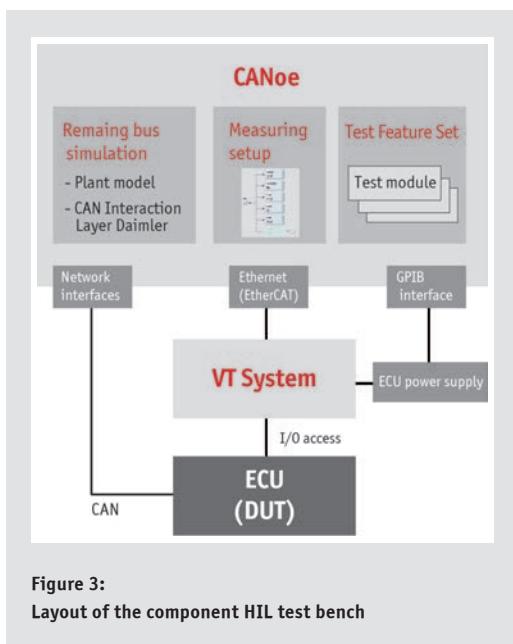


Figure 3:
Layout of the component HIL test bench

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, 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**

Figures:

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 unnecessary 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

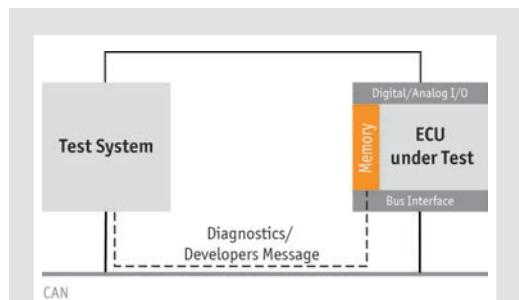


Figure 1:

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

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 as needed. The protocol can also periodically supply a defined set of measured values from the ECU 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 synchronized 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, or it may need to be generated in a separate step from the linker-map information. 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 XCP standard, 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, CANoe's Test Feature Set offers extensive options for creating test cases and automatically evaluating them. 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

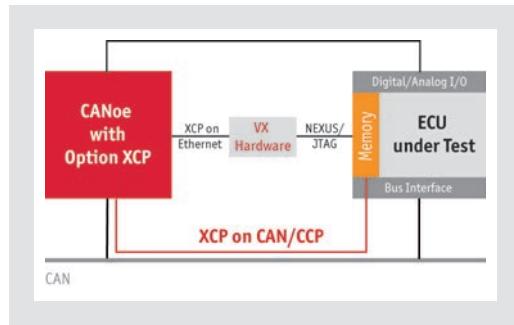


Figure 2:

Option XCP is used with a CANoe test system for direct access to an ECU's internal values.

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 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 circumventing the microcontroller. Taking this approach, the user can quickly read out very large quantities of data from the system without loading the bus or the ECU.

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.

All Figures:

Vector Informatik GmbH

Links:

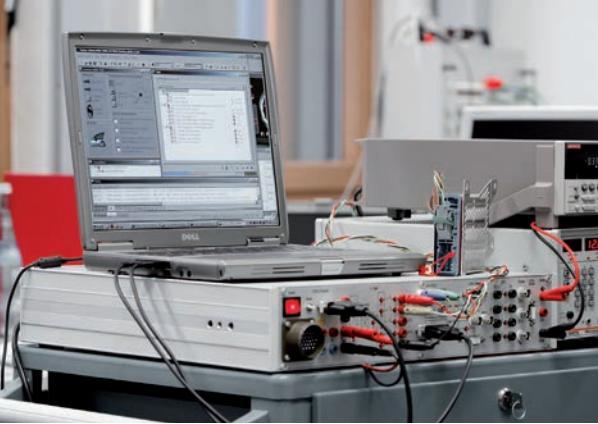
Homepage Vector: www.vector.com

Product Information CANoe Option XCP: www.vector.com/canoe_xcp



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 and Product Manager for CANoe Option XCP.



Case Study

Automating ECU Test Execution, Pass/Fail Detection, and Report Generation

Nippon Seiki Co., Ltd.

The Customer

From the time it was founded, Nippon Seiki has extended its business, focusing on the development and manufacturing in-vehicle instrumentation. Nippon Seiki's technological competence in this field is highly rated, and as a leading manufacturer of instrument panels it supplies a large number of automobile and motorcycle manufacturers.

The Challenge

Reducing ECU testing man-hours while simultaneously improving quality of overall testing

With growing computerization in automobiles, instrument panels are moving toward multifunctionality. That is why manual ECU test execution, pass/fail detection, and creating reports took a huge amount of time. This has resulted in increased ECU testing man-hours, and their reduction has become a major issue.

The Solution

Automation of ECU Testing using the CANoe Test Feature Set

CANoe, provided by Vector, is used by a large number of customers who develop in-vehicle ECUs. The CANoe Test Feature Set is designed to automate ECU stimulation, pass/failure detection and test report generation. To solve the issue of increased man-hours, the following proposal was made to Nippon Seiki:

- ▶ **Test samples:** Delivery of a total of five test samples, including a CAN Message Synchronization Test, Diagnostic Test, and Gateway Test.
- ▶ **Training of employees involved in testing.**
- ▶ **Technical support:** In the context of technical support for customers with maintenance contracts.

The CANoe Test Feature Set can also support fault diagnostics and serve as a diagnostic tester.

The Advantages

Test Execution Time Dramatically Reduced

Automating ECU test execution, pass/fail detection and report generation helped Nippon Seiki to successfully reduce total ECU test time from more than 18 hours to just over 12 hours. In particular, test execution time, which previously took 11 hours, has now been shortened to about 4 minutes! The more frequently testing is executed, the greater the gains in testing efficiency. Finally, use of the CANoe Test Feature Set has dramatically reduced the man-hours needed to perform the tests (see table below).

A System Design Department Manager described the advantages of automatic execution like this: "There is certainly some test case implementation work to be done when automating. But once the test case is created, the ability to reuse it again and again is a huge advantage. In turn, this improves overall testing quality."

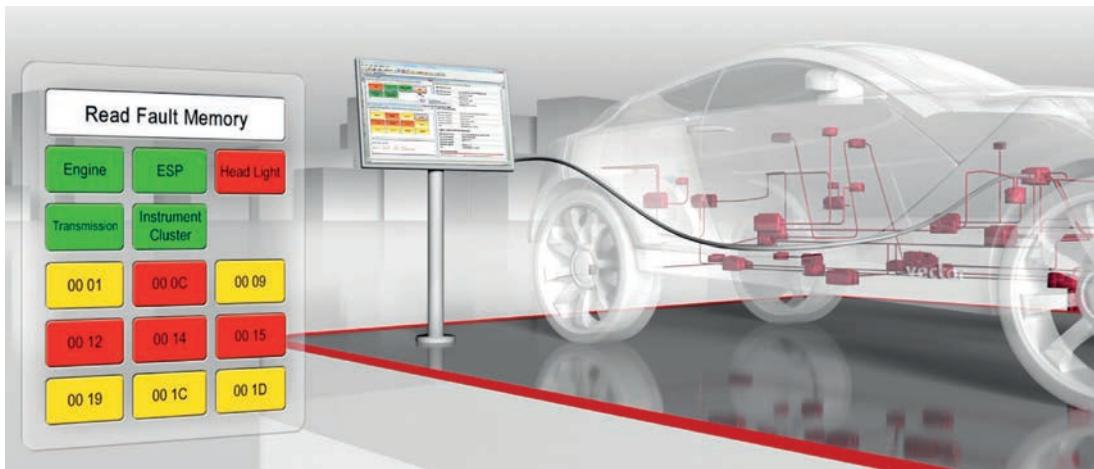
In this project, 30% of all ECU test cases are automated using the CANoe Test Feature Set. Nippon Seiki will be expanding its automated testing coverage in the future and targeting even greater efficiency in ECU testing.

No	Test Cases	Conventional testing (manhours) [hrs.]		Automated testing using CANoe Test Feature Set (manhours) [hrs.]	
		Test preparation (test spec creation, fixture setup, etc.)	Testing execution	Test preparation (test spec creation, fixture setup, etc.)	Testing execution
1	High-speed CAN -> Low-speed CAN gateway response confirmation	1	7	1	0.007
2	High-speed CAN -> Low-speed CAN gateway data values confirmation	4	1.5	8	1 (*)
3	Low-speed CAN frame transmission cycle confirmation	0.25	1	1	0.028
4	Reading and writing of internal data by DIAG (Diagnostics)	1	0.9	1	0.019
5	Reading of software information by DIAG (Diagnostics)	1	0.6	1	0.011
Subtotal		7.25	11	12	0.065
Total		18.25		12.065	

(*) Test No. 2 is executed simultaneously with Test No. 1.

Table: Comparison, in relation to test case and manhours, between conventional test method and the use of the CANoe Test Feature Set

Efficient ECU Tests with Fault Memory



Vehicle diagnostics is a very useful yet very complex topic over a vehicle's development and service life. Development and service department times can only be minimized if reliable, correct and precise data is available at all times. Suitable test tools are needed to reach this high level of quality while developing ECUs and their diagnostic functionality. This article describes both the test tools and the methods used to quickly and reliably avoid the causes of errors.

The importance of good diagnostic functions for successful vehicles is illustrated by countless reports of vehicle defects that required multiple repair attempts until they were finally corrected. This causes unnecessary aggravation and costs, and it is damaging to the manufacturer's image. Therefore, the motto must be: "If there is a defect, it must be corrected quickly and accurately."

The fault memory logs irregularities before they finally lead to component failure. With regular vehicle maintenance in a service department, defects can be detected and corrected before undesirable effects make their appearance. Diagnostic functions are also important during product development, because error states naturally occur more frequently during testing and trial phases. Relevant data from the vehicle makes it easier to perform diagnostics, possibly without even requiring any additional test instruments. Therefore, it is important to correctly log error states in fault memory.

Functional Principle of Fault Memory

Based on the widely used Unified Diagnostic Service (UDS) standard ISO 14229 [1], this article explains which data is provided for fault diagnostics. A central element of the UDS standard is 24-bit trouble codes. Most of the trouble codes identify OEM-specific error causes such as "Battery voltage – voltage too low." In addition, the

standard defines certain trouble code groups such as "Drivetrain." For each trouble code, an 8-bit status mask indicates whether the error occurred and when, such as "Test not passed since current power up." Even more supplemental details can be stored as optional environmental data. This data is OEM-specific and it records other helpful measurement data, such as the number of error events and at which odometer reading the error was observed for the first time and most recently.

During vehicle operation, each ECU collects the error states that have occurred in its fault memory. With suitable tools, you can use diagnostic services to access this memory. Typical requests are:

- > Report all trouble codes that fit a specified status mask
- > Read the status and environmental data for a specific trouble code
- > Clear the status and the environmental data for a group or an individual trouble code

The service department tester generates a diagnostic request and sends it to the ECU. The ECU responds with a diagnostic response. The response usually contains multiple error messages with their trouble codes, the associated status masks and environmental data (**Figure 1**). All data must then be evaluated. Causes of errors can be very quickly localized with the prepared information, and the defective component can be replaced.

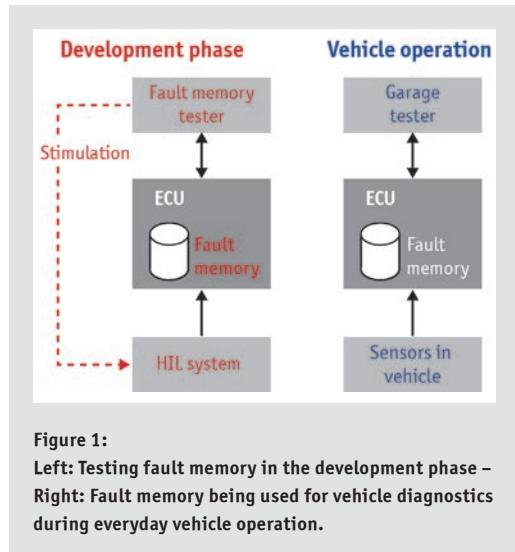
Testing fault memory functions

One strength of fault memory is that the vehicle provides the algorithms for detecting error states. However, this assumes that the vehicle's diagnostic functionality itself is operating correctly. Consequently, this functionality must be intensively tested during the vehicle's development. Tests of diagnostic functionality can be very complicated due to the very large number of existing trouble codes, and the extensive environmental data and error states. This is especially complex, since the parameters of the diagnostic requests must be set, and the response parameters must be checked for each test. However, this situation can be considerably simplified by having the tester use an abstract perspective. This fully hides routine activities for setting, checking and finding error entries from the user.

It would be ideal, if - after stimulation (**Figure 1**) of a situation logged in the fault memory - a test function were available that could hide all UDS-specific aspects and automatically check the diagnostic responses. Such a function might do the following: "Read the environmental data for trouble code XYZ, check whether it is active and whether the voltage was between 11.5 and 12.5 Volt when the error occurred!" The test code for this function would be extremely compact and clear.

Efficient Access to Fault Memory

The described abstraction level might, for example, be provided by the CANoe test tool in combination with the Test Automation Editor



test design tool. Both are software products from Vector, and they support both the UDS and KWP 2000 protocols.

A number of test functions are intended for fault memory tests, and they can:

- > Read out individual trouble codes with their environmental data, check them and document them in a test report. If necessary, allowable trouble code combinations or alternatives can also be shown.
- > Read out all trouble codes that have a specific status mask
- > Clear the fault memory
- > Read out the supported trouble codes
- > Read out the status of the fault memory

Test functions in the Test Automation Editor can be used intuitively, and CANoe automatically selects the correct OEM-specific diagnostic services. Automatic selection makes the test sequences easier to reuse for other ECUs, and they are easier to create overall. This lets users work very effectively, since they can focus on the actual goals of testing.

Figure 2 illustrates the method for accessing the fault memory. During execution, the ECU's fault memory is polled by entering the desired "DTC status mask". In the response, which might contain many different trouble codes in any sequence, a check is made of whether the trouble code P000004 was reported. If this trouble code is found, the status bit "Failed since last clear" must be set.

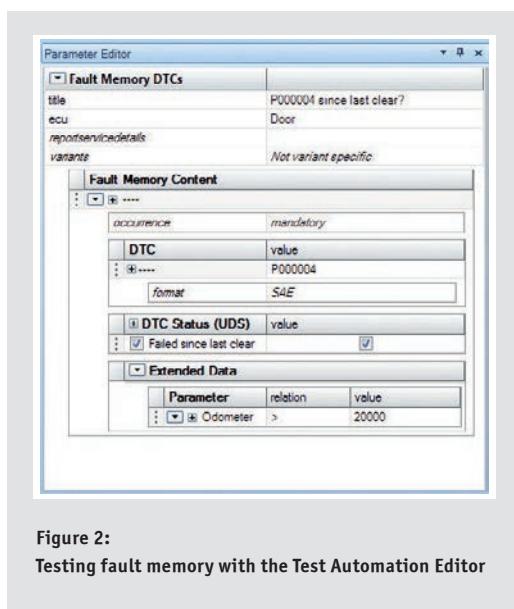


Figure 2:
Testing fault memory with the Test Automation Editor

Each trouble code has associated environmental data, as illustrated by the odometer reading in **Figure 2**, and this is very easy to check.

Although this test function controls a complex sequence, it is easy to set its parameters. In a conventional programming language without dedicated fault memory support, however, it would take considerable effort to realize the same functionality.

Fault Memory – Not a riddle wrapped in an enigma

The primary goal has been, and continues to be, a fault-free and robust vehicle. Vehicle problems can be detected early and corrected using the fault memory. Therefore, correct self-diagnostics with fault memory should be checked with extensive tests starting early in development. This checking can be formulated in an uncomplicated and very efficient way on an intuitive abstraction level using test design and test execution tools. In this way, fault memories can contribute towards early detection of vehicle problems before they lead to vehicle failure in everyday driving.

**Translation of a German publication in Elektronik automotive,
issue 2/2013**

Figures:

Vector Informatik GmbH

Link:

Homepage Vector: www.vector.com



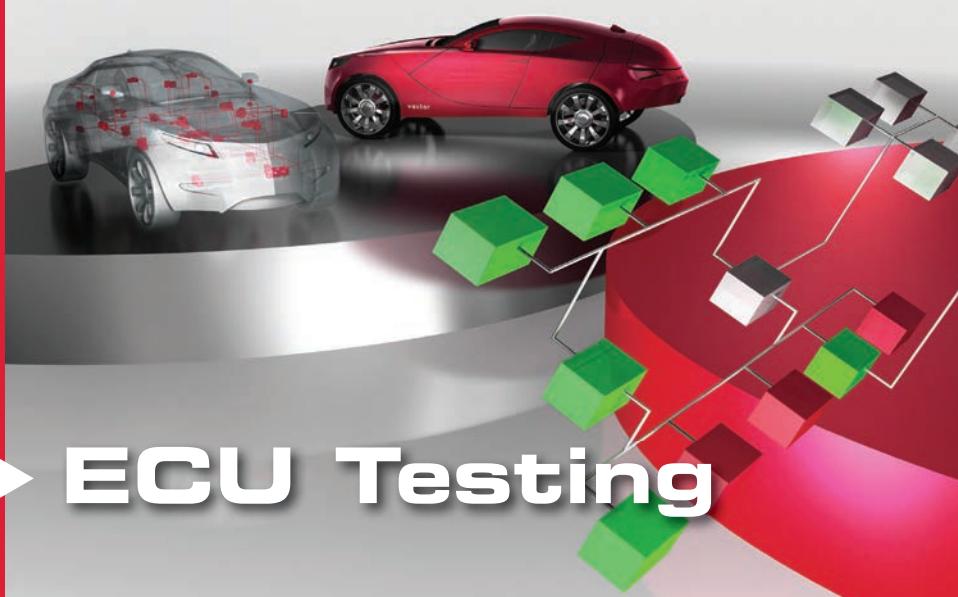
Martin Huck

studied electrical engineering at the Ruhr-University in Bochum, Germany. He was then employed as a development engineer in the communications industry for a number of years. At Vector, he worked on software development for CANoe's Test Feature Set until 2012, and he is now a technical trainer.



Siegfried Beeh

studied electrical engineering at the University of Stuttgart, Germany. Since 2002, he has been actively working in software development at Vector, and since 2006 he has been responsible for the "Test Tools & Diagnostic Features" group. As Product Manager for Testing Tools, he is responsible for such product components as the CANoe Test Feature Set and the Test Automation Editor.



►► ECU Testing

Systemize Your ECU Test Workflows Even More

with systematic and reproducible tests. Efficient test systems accelerate your CAN, LIN, MOST, FlexRay, IP/Ethernet and AUTOSAR development.



- > Easy test management and test data management for the planning and tracking of test projects, test phases and test tasks
- > Real test environments for very early phases of development – including generated remaining bus simulations
- > Automatically created test cases and reports for systematic component and system tests
- > Short test times, since identical platform is used for development and testing
- > Provide I/O ports and interfaces for calibration and diagnostics
- > Let you simulate and stimulate ECU functions

Vector tools and services help you achieve optimum test quality and test coverage within all development phases.

- Information and downloads: www.vector.com/ecu-test

Now available!

- **vTESTcenter** the tool for test management and test data management
www.vector.com/vtestcenter
- **vTESTstudio** the authoring tool for optimized test sequences
www.vector.com/vteststudio

vector

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.

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



Figure 2:
VT System modules enable setup of very compact test systems as well.

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 substan-

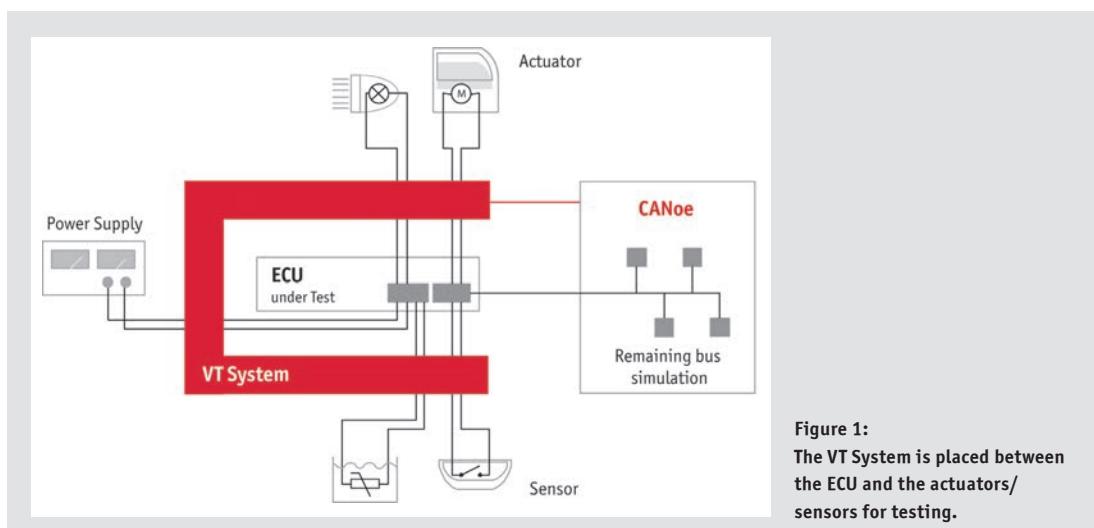


Figure 1:
The VT System is placed between the ECU and the actuators/sensors for testing.

tially simplifying the test bench setup (**Figure 3**).

- > 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

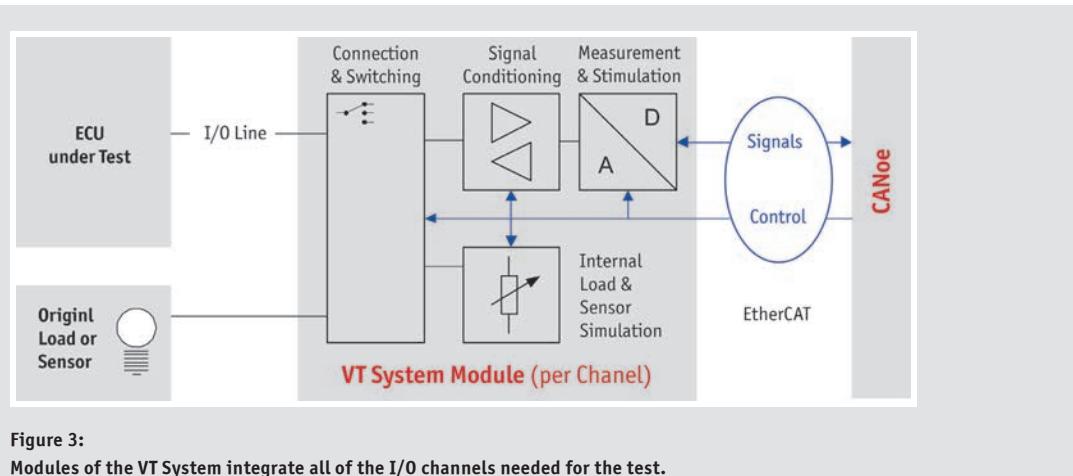


Figure 3:
Modules of the VT System integrate all of the I/O channels needed for the test.



Figure 4:
The new VT7001 module supplies an ECU's power supply inputs.

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.

Translation of a German publication in Elektronik Praxis, Special Edition 'Automotive', issue September 2011

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.

Porsche Validates Gateway ECUs Automatically

Real-time analysis in driving trials supplements laboratory tests



On the Panamera, Porsche is implementing the “Porsche Echtzeit (‘real-time’) TRace Analyser” (PETRA) for validation of gateway ECUs. PETRA, which is based on the CANoe test tool, automatically generates gateway tests from routing tables. These tests are used in driving trials, checking the routing functions online. PETRA can be used to conduct analysis later as well, based on recordings of the bus communication as a supplement to the scope of sequential testing in the laboratory. This gives Porsche a test tool for automatically detecting routing errors.

Architecture

The “Panamera,” available since 2009, is the first production vehicle in the history of the sports car manufacturer to unify a four-door Gran Turismo concept with classic sports car design. Engine options range from a 3.6-liter six-cylinder engine to a 4.8-liter eight-cylinder turbo, spanning a power range from 220 to 368 kW. Depending on the specific model version and options, features may include full-time all-wheel drive, dual-clutch gearbox and a model with hybrid drive, which is currently in development.

The electronics architecture of the Panamera is based on more than 30 CAN ECUs, which are distributed over six CAN networks. It is supplemented by twelve LIN networks with a total of 25 LIN slaves. A MOST bus serves to network the Infotainment systems. The six CAN buses for diagnostics, convenience systems, HMI

(Human Machine Interface), powertrain, chassis and crash safety are all connected – with a maximum of two LIN branches – to a central gateway ECU (Figure 1). Gateways interconnect the various networks in the motor vehicle and provide for fast data exchange across networks.

Intelligent gateway functions

In the motor vehicle, it is important to transfer information between networks as quickly and intelligently as possible. One must consider the different bandwidths of the network buses. As a rule, a periodic message arriving at the receiver end of a High-Speed CAN bus every ten milliseconds cannot be routed by the gateway as quickly as on a target network with a lower bandwidth. The copying rule here could be that the most recent value should

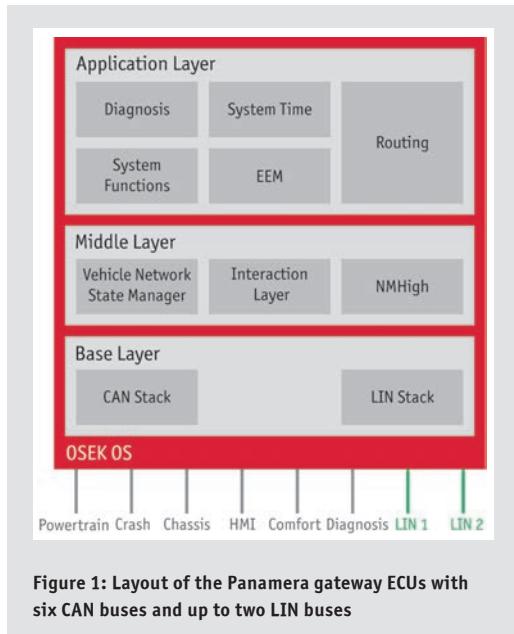


Figure 1: Layout of the Panamera gateway ECUs with six CAN buses and up to two LIN buses

only be sent every 100 milliseconds on the slower bus. Suppressing messages can reduce the volume of data and adapt it to the lower bus speed. In the reverse case – from slow to fast – the gateway may need to repeat the messages in its sending section, i.e. send a message more frequently than it is actually received. Other messages, on the other hand, should be routed as quickly as possible and one-to-one without multiplication or suppression. For this purpose, a routing rule is stored in the gateway for each message to be routed. These rules contain information on the source and target networks and the copying rule.

Integration of gateway ECUs in the vehicle is the responsibility of the automotive OEM. Error-free operation of the gateway is a key prerequisite to ensuring that the networks and ECUs work together properly. In verifying routing behavior, it is absolutely essential to run numerous tests in different driving situations and conduct a detailed analysis. Until now, the focus of laboratory tests was to sequentially check all of the gateway's routing rules by stimulation of the input signals. On the output bus, the expected results were checked against routing algorithms. At Porsche, these tests are also performed under different physical conditions such as under-voltage and over-voltage as well as various temperature profiles.

Limitations of laboratory tests

Although test data generated in the laboratory with special test or HIL systems does indeed test basic functionality, sporadic errors

are often only revealed under real stress scenarios and in the interplay of all bus systems and ECUs. In particular, stress situations under high bus load and the simultaneous occurrence of certain events can lead to complex errors.

The central gateway of the Panamera routes more than 3,000 different CAN signals over the six CAN buses in fractions of a second; in the process, it must consider 25 different transmit types. The time consuming manual error debugging in test vehicles led Porsche developers to seek a more efficient method of evaluating vehicle network traces in developing the Panamera. Key requirements for the new system were that it should be capable of verifying the gateway data traffic during real test drives and in real-time. However, after initial market inquiries with tool producers and test specialists it became apparent that a suitable tool or test system was neither available nor known.

Porsche Echtzeit ('Real-Time') Trace Analyser

A solution was created in collaboration with specialists from Vector, whose effort led to a tool known as the "Porsche Echtzeit ('real-time') TRace Analyser (PETRA). The testing and simulation software CANoe serves as the foundation for PETRA. This tool was optimized for ECU tests and it enables – by means of various configurations – automatic test sequences including rest-of-bus simulations. Furthermore, it offers the ability to replay logged network communication and to create detailed test reports. To implement automatic test sequences, the Test Feature Set (TFS) contained in CANoe provides test functions in the Communication Access Programming Language (CAPL) script language. CAPL uses a C-like syntax for formulating tests; CAPL recognizes special functions, which lets it react to bus-related events such as the receipt of a message. Tests can be set up and parameterized in XML files using predefined test functions and control functions. A complete test run is represented by a test module in CANoe.

PETRA was developed as a contract project by Vector, and together with standard functions of the CANoe TFS, it serves to validate the CAN gateway functions of the central Panamera gateway ECU (Figure 2). It generates the entire CANoe test module components needed to validate a gateway revision level; these components can be used both during test drives as well as offline in the evaluation of data recordings. Routing and copying rules for the messages of multiple buses can be evaluated simultaneously here so that they could focus on specific messages in individual tests. Porsche engineers insisted on wide-ranging filter options right from the start. The gateway developers expressly wanted a system that could check the totality of all messages and be able to analyze individual routing relationships with a simple configuration. In particular, developers wanted to intentionally filter out the special data traffic that result in exceptional situations, for example: in starting the vehicle, shutting down the network, since most test

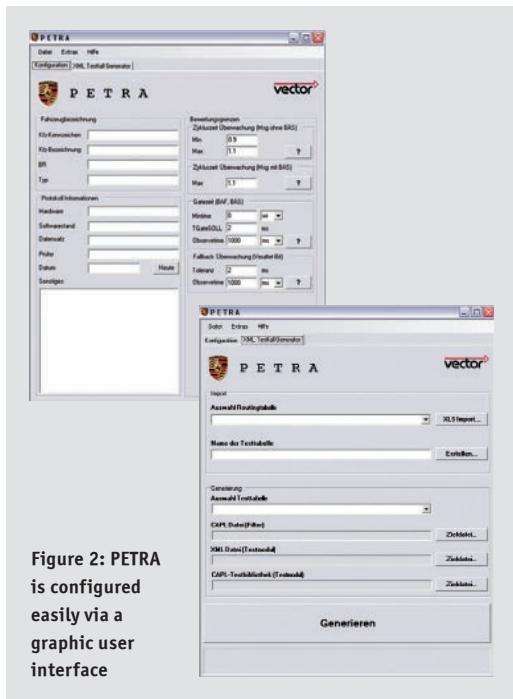


Figure 2: PETRA is configured easily via a graphic user interface

systems generate a massive number of irrelevant error messages in these cases.

Suitable for both online and offline tests

A routing table in Excel format describes all signals and messages to be routed by the gateway. Each line corresponds to a routing rule and defines the way in which a message should be routed. Messages that are routed in their entirety (1:1) are organized into

the following message types: Periodic messages, BÄS messages and BAF messages. BÄS messages (BÄS = immediately on change) are special cases of periodic messages, which the responsible ECU sends in addition to the normal periodic sending as soon as an input value changes – e.g. from a sensor. BAF messages (BAF = on active function) are one-to-one messages that the gateway should route directly and without modifying the cycle times. In addition, the gateway generates and sends messages with signals from different Rx messages (“combined messages”). For each Rx message, the generated message contains a so-called “aged bit.” If the gateway determines that a delay has occurred in incoming periodic messages that exceeds the allowable tolerance, it sets the aged bit before routing the combined message. Active aged bits indicate to the receiving ECUs that the informational content of the combined message has limited validity.

The user can use PETRA to take information from the routing table and automatically generate an extended configuration table, also in Excel format. The test engineer then uses this to select the messages to be analyzed in the gateway data traffic (Figure 3). If a routing relationship is selected in the first column, the test generator considers this line in creating the test configurations. All other messages are ignored in this test. Furthermore, various evaluation limits may be specified, for example tolerances for cycle time monitoring and the allowable gateway time for BAF and BÄS messages. PETRA then automatically creates all of the files needed to execute a test run (Figure 4). Examples include an XML test module with a CAPL test library as well as a CAPL filter that might be used upstream of the Trace Window in the CANoe Measurement Setup. The CAPL pass filter only passes those messages that are needed for the generated monitoring tasks. This covers the basic requirements for a CANoe test run (Figure 5). As a standalone tool, PETRA lets users create test modules without requiring that a CANoe license be installed.

Not considered

Considered

Considered

	Pos. No.	Rx ECU	Rx Bus	Rx Message/Function	Rx Signal
1	Airbag	CAN_Powertrain	Airbag_01	1 to 1 Message	
1	Airbag	CAN_Powertrain	Airbag_01	1 to 1 Message	
x 1	Airbag	CAN_Powertrain	Airbag_01	AB_Beltaert_VF	
1 1	Airbag	CAN_Powertrain	Airbag_01	Monitoring absolet	
1	Airbag	CAN_Powertrain	Airbag_01	AB_Beltaert_VF	
1	Airbag	CAN_Powertrain	Airbag_01	Monitoring obsolet	

Figure 3: Only the messages selected in the first column are considered in generation of the test configuration

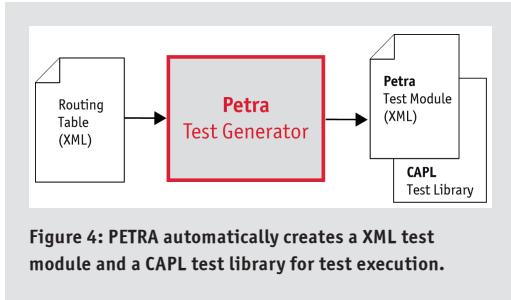


Figure 4: PETRA automatically creates a XML test module and a CAPL test library for test execution.

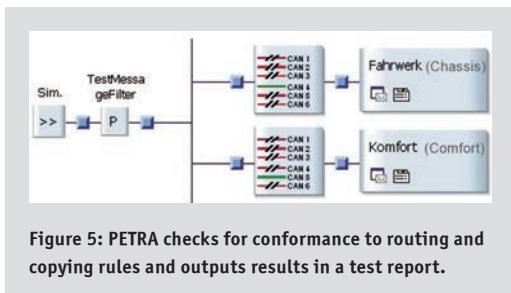


Figure 5: PETRA checks for conformance to routing and copying rules and outputs results in a test report.

Test report speaks the language of the automotive OEM

During the test, the test system creates a detailed report on events that occurred. The test report was optimized for Porsche-specific processes and terminology, so that it can be equally understood by both experts in gateway development and colleagues from other technical departments. The checks seamlessly cover all relevant copying rule violations for the specific message type, and they can operate on either the byte or signal level, depending on the situation. For road tests, the system can be configured so that when an error occurs, a relevant message immediately appears on the screen. This enables immediate repetition any routing errors resulting from operating and driving maneuvers.

In offline tests, log files recorded during the test drives are played back in CANoe via Replay blocks. One helpful debugging feature is the ability to jump directly to the point in the trace data at which an error has occurred. Frequently, events occurring before and after the error yield useful information on the cause of the error. Error statistics provide important indicators on the frequency of errors and "error quality." For example, it might answer questions about whether an error occurs continually or only once every 1,000 kilometers, or whether certain cycle time violations are very large or relatively small on average.

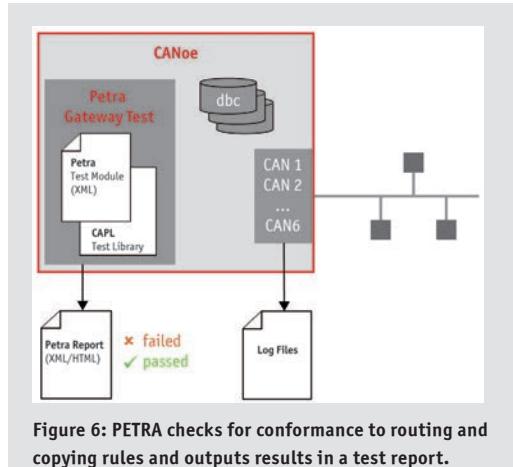


Figure 6: PETRA checks for conformance to routing and copying rules and outputs results in a test report.

Summary and Outlook

The CANoe tool has been used at Porsche for a long time now. In 2007, the existing tool chain was extended and continually enhanced with the addition of the "Porsche Echtzeit ('real-time') TRace Analyser." PETRA demands very little training effort for employees. The new automated approach to analyzing the routing of real communication data ideally supplements the stimulation-based test scenarios of laboratory tests. Engineers at Porsche and Vector Informatik closely worked together in developing the new gateway validation system. The tool enabled early detection of potential errors during ECU development for the Panamera and attained a high maturity level. The time advantage in analyzing communication data is about 65 percent compared to previous manual debugging. The generator approach lends flexibility to the PETRA concept, which can be applied to future vehicle series and generations. If necessary, other bus systems such as FlexRay or Ethernet could also be integrated in the PETRA concept.

Translation of a German publication in ATZ elektronik, issue 6/2010

Figures:

Porsche: initial figure, 2, 3, 5

Vector: figure 1, 4, 6

Links:

Homepage Porsche: www.porsche.com

Homepage Vector: www.vector.com

Product information CANoe: www.vector.com/canoe



Dipl.-Ing. (FH) Karl-Peter Spring
studierte Automatisierungstechnik an der FH Reutlingen. Er trat 1999 in die Dr. Ing. h. c. F. Porsche AG ein und ist dort in der Abteilung „Vernetzung, Diagnose und Gateways“ als Sachgebetsleiter für alle zentralen Gateway-Steuergeräte verantwortlich.



Dipl.-Inf. Thomas Hohmann
studierte Informatik an der TU Ilmenau. Seit 2007 ist er bei der Dr. Ing. h. c. F. Porsche AG in der Abteilung „Vernetzung, Diagnose und Gateways“ in der Gateway-Entwicklung tätig.



Dipl.-Ing. Katja Hahmann
studierte Elektrotechnik an der TU Chemnitz. Sie trat 1997 in die Vector Informatik GmbH in Stuttgart ein und ist dort Gruppenleiterin der CANoe Anwendungsentwicklung in der Produktlinie Networks and Distributed Systems.



Case Study

Testing FlexRay ECUs effectively and reproducibly

bertrandt

The Customer

Bertrandt AG is one of Europe's leading development partners in the automotive and aerospace industries. About 5500 employees at 30 business sites in Europe and in the USA work on customized solutions ranging from individual components and modules to complete systems and vehicles.

The Challenge

Testing FlexRay ECUs effectively and reproducibly

Proper behavior of FlexRay ECUs in normal operation and under fault conditions is to be tested beginning in early development phases. Test sequences of complex test cases need to be executed, but it must also be possible to manually start individual tests. The tests require a real-time capable remaining bus simulation, to enable reliable statements about the communication behavior of the ECU under test.

The Solution

ECU tests within a remaining bus simulation on a FlexRay test bench

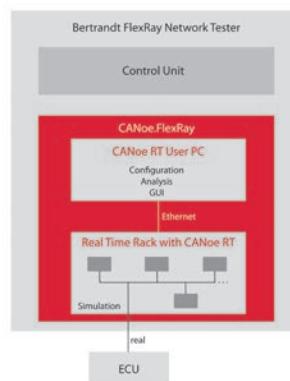
Bertrandt created a special test system to conduct reproducible testing of FlexRay ECUs. It contains a simulated vehicle environment that is implemented with an extensive and real-time relevant remaining bus simulation. CANoe.FlexRay is used here to simulate the remaining bus for the ECU under test. To fulfill demanding real-time requirements, CANoe RT is used to distribute CANoe.FlexRay functionality between two computers. The simulation is executed on a high-performance computer, the Real Time Rack. The configuration and user interface run on the host computer of the test bench. The two computers are interconnected via Ethernet.

The Advantages

A high-performance and flexible environment for complex ECU tests over the FlexRay bus

For conducting the simulation, Bertrandt uses CANoe.FlexRay together with CANoe RT and the Real Time Rack to implement a high-performance test system for FlexRay:

- ▶ Testing an ECU with remaining bus simulation ⇒ Testing is already possible in early development phases without requiring availability of real ECUs.
- ▶ Using OEM-specific modules such as the Interaction Layer, Transport Protocol, checksum calculation, etc. in the remaining bus simulation ⇒ Quick and easy implementation of the remaining bus simulation and standardized execution of OEM-specific functions.
- ▶ Execution of remaining bus simulation with CANoe RT ⇒ Simulation is characterized by deterministic execution, short latency and quick response times.
- ▶ Remaining bus simulation is executed exclusively on the Real Time Rack ⇒ No disturbance of the simulation by the PC's operating system.
- ▶ Real Time Rack is scalable ⇒ Flexibly adaptable to future test system requirements, e.g. number of FlexRay channels or modification for other bus systems.



Easy Access to Bus Analysis

Changing requirements are continually in flux when it comes to networking ECUs. 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 bus 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 bus. Here, the Trace function shows the time sequence of all bus events. It is also possible to graphically display individual parameters. Moreover, the user typically wants an overview of the bus statistics. In stimulation, on the other hand, specific messages need to be sent on the bus either spontaneously or periodically. And of course data needs to be logged for later offline analysis.

These three core tasks are the domains of CANalyzer Beginner, a special execution mode of CANalyzer from Vector. 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. This makes it easy to perform the tasks mentioned above quickly and effectively.

CANalyzer Beginner can be immediately used as part of any CANalyzer installation (CAN and LIN buses). 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 completely revised window layout in CANalyzer. The individual tasks are organized on fixed desktops, which do not need to be modified, and which already contain preconfigured windows (**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

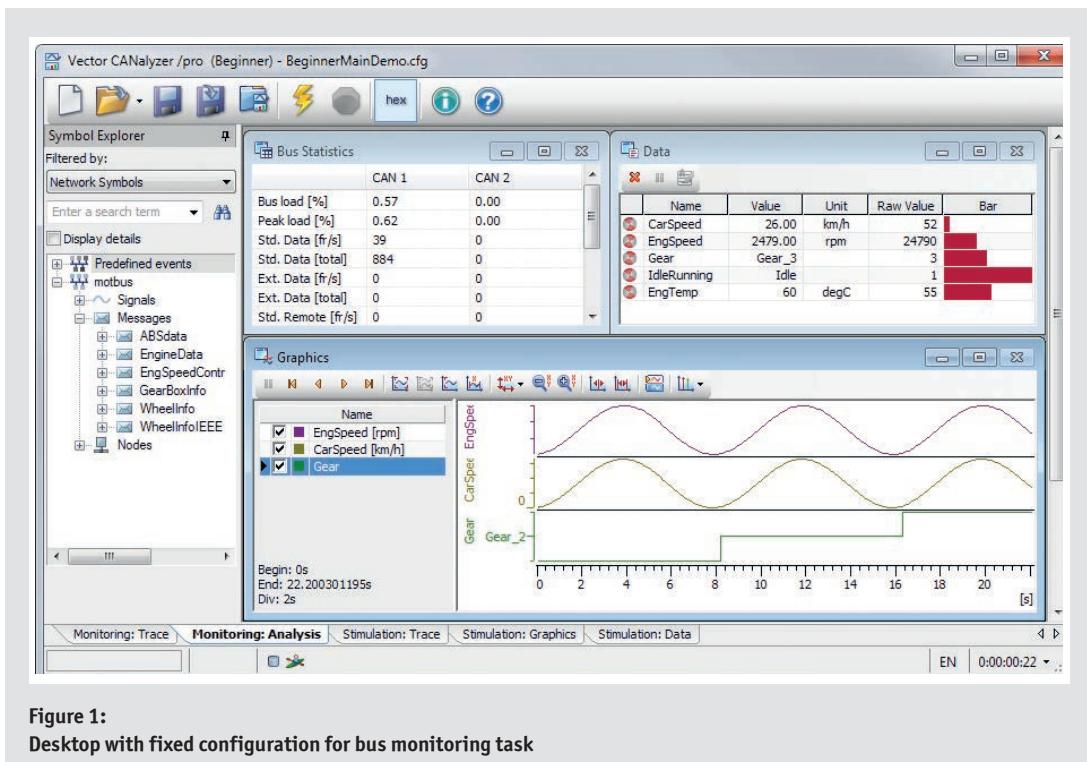


Figure 1:
Desktop with fixed configuration for bus monitoring task

closed, which eliminates searching for inadvertently closed windows. The windows are configured exclusively by drag & drop operation or with the help of functions on the toolbar.

You can create your own configuration with just a few mouse clicks. To do this, the user only needs to add - for each bus - a channel and a suitable network description file (DBC for CAN or LDF for LIN) to the central configuration window (**Figure 2**). If applicable, the baud 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 very long data history, so that even long-term measurements over several days can be fully preserved. The Statistics window offers a detailed summary of the current situation on the bus and can prepare statistical information on the node level or even the message level.

If complex tasks need to be performed, that's not a problem . A configuration created with the Beginner mode can be loaded in its full form in CANalyzer. It is even possible to use CANalyzer to perform further offline analysis of logged data. There is a seamless migration path from CANalyzer Beginner to CANalyzer – and for

that matter CANoe as well; because configurations that have been created with CANalyzer Beginner can also be loaded in CANoe.

Future planning for CANalyzer Beginner calls for supporting additional tasks and possibly adopting concepts into CANalyzer.

Publication in CAN Newsletter, issue December/2012

Links:

Homepage Vector: www.vector.com

Webpage CANalyzer: www.vector.com/canalyzer



Jochen Neuffer

studied Communications Engineering at the Technical College in Esslingen. Since 2002, he has been working at Vector Informatik where he works as a Product Management Engineer in the area of Tools for Network and Distributed Systems.

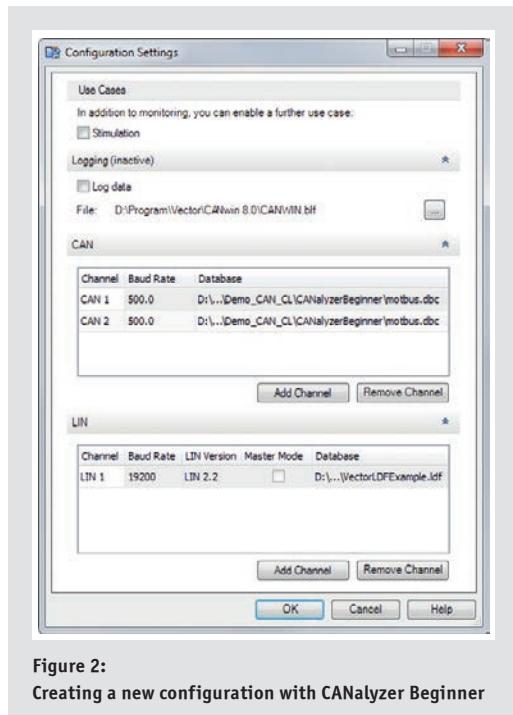


Figure 2:
Creating a new configuration with CANalyzer Beginner

Network Access with Short Reaction Times

Automotive interface with integrated real time computer



New bus systems and a growing number of networks are pushing conventional interfaces to their performance limits. A crucial factor in the quality of tests, analyses and simulations is the degree of real time performance in data exchange between vehicle networks and test systems. A modular hardware interface concept now enables extremely short cycle and response times plus, it offers the potential for interesting applications in a stand-alone operation.

Key characteristics of the ideal interface hardware for CAN, LIN, FlexRay or MOST are high level of real time capability, support for many different networks and easy handling. These somewhat contradictory requirements could not always be satisfied by previous concepts. A particular challenge for software and interface developers is to satisfy the wishes of users for a hardware interface with an uncomplicated plug & play connection to a PC via USB. One problem is that high latency times at the USB interface may have a critical effect on real time behavior, depending on the particular system.

Real time performance despite USB

Until now, network interfaces only permitted pure bus access – and all simulation, analysis and testing tasks were executed on the connected Windows-PC. This often rendered real time capability inconceivable due to the slow USB connections as well as the many PC

user inputs and possible IT background services that could be running. In finding a solution to this challenge, Vector took a new approach in developing its current generation of VN8900 interfaces (Figure 1). Using the advanced Z530 Intel-Atom platform made it possible to develop a cost-effective, high-performance, real time hardware architecture that utilizes a Windows XP embedded application for the specific application case runs. As a result, it is very easy to execute portions of the simulation or test environment on either a normal PC or the network interface, reducing the load on the Windows PC. Real time critical tasks are executed right on the interface hardware, while data preparation, saving, visualizing and other standard tasks run on the Windows PC as usual. This makes system reaction a factor of two to eight times faster compared to previous passive interface solutions, depending on the specific application and operation (Table 1).

The modularly constructed system consists of a base module and a plug-in module in a rugged, compact housing. It was specially

developed to operate together with the CANoe and CANalyzer simulation and analysis tools from Vector (Figure 2). This solution plays out its strengths especially well when parallel access to multiple bus channels, additional I/O performance or very short response times are required.

From the user's perspective, the new system operates like a conventional network interface: measurement processes run as they did previously, but at noticeably higher speeds. Handling and configurations also remain the same. CANoe and CANalyzer autonomously load the relevant test and simulation components onto the interface hardware. This also applies to the necessary communication and diagnostic databases as well OEM-specific extensions such as network management or the interaction layer. Even older and slower PCs can serve as a Host PC without users having to compromise on real time capability.

Standalone operation opens up new types of use cases

Besides classic test, analysis and simulation tasks, new types of use cases are now available in standalone operation of the VN8900. If a CANoe/CANalyzer application does not require any user interactions, it can be run autonomously on the interface hardware. This capability enables interesting uses such as remaining bus simu-

Network Interface	Min. Response Time (Resolution)	Max. Time Jitter	CAN Response Time
Basic Module Vector Real Time Boost	100 µs	50 µs	100 µs
Basic Module Vector Standard Modus	500 µs	250 µs	260 µs
Vector PCMIA	>1,000 µs	<2,000 µs*	<2,000 µs*
Vector USB	>1,000 µs	<5,000 µs*	<5,000 µs*

* Typical value, depending on Windows operating system

Table 1:
Real time performance at full load (20,000 messages/seconds)

lation in larger test stations and HIL systems that lack their own remaining bus simulation. With just a few mouse clicks, the user can create the remaining bus simulations based on the appropriate OEM extension and then load them onto the interface hardware. Other standalone applications include gateway simulations and stimulation in long-duration tests.

The Ethernet/UDP based interface referred to as FDX (Fast Data Exchange) is available for integrating standalone applications in other systems. It permits fast write and read access to bus signals and system variables. In addition, the application can be executed

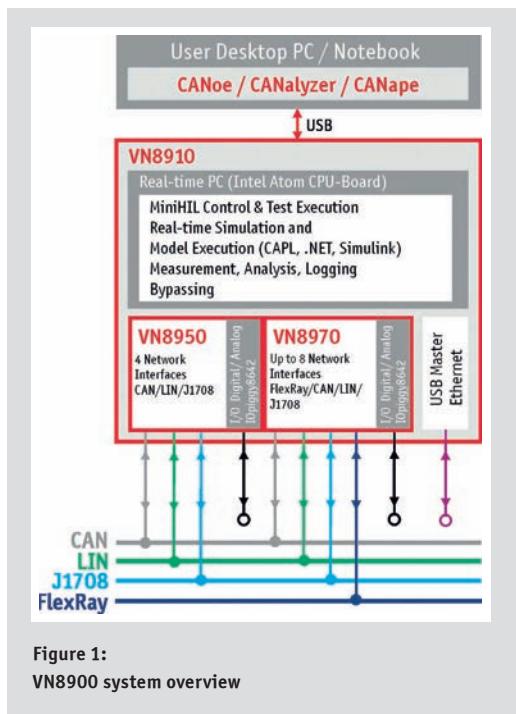


Figure 1:
VN8900 system overview



Figure 2:
VN8900 consisting of base module and plug-in module

over this interface. The Standalone Manager is used to load prepared CANoe/CANalyzer standalone configurations from any given PC to the interface module. This process can be automated for optimal integration in a specific environment.

Summary and Outlook

The VN8900 concept lets users implement high-performance, economical complete solutions with an easy-to-operate USB connection. Significant performance gains are realized in CANoe/CANalyzer simulations and analyses by swapping out time critical computations to the external processing unit of the VN8900 system. It is even possible to use the device as a Mini-HIL right at the workbench. This does not require any additional configuration effort by the user. Users also benefit from the uncomplicated USB connection to the Windows PC, avoiding IP address, security or firewall related conflicts. Other functions already in development illustrate the unexploited potential of the system. They include Vector Real Time Boost, which permits execution of CAPL nodes on the Kernel-Mode level. This, in turn, leads to further significant real time gains. Supplementing the CAN and LIN modules is another plug-in module that offers a FlexRay connection with Cold Start Helper.

Key technical characteristics VN8900 system

Base module in a rugged, compact housing:

- Intel Atom CPU Z530 operating at 1.6 GHz frequency
- 1GB RAM and solid state memory
- Ethernet port, 2 USB ports
- 4 programmable film keys and 6 controllable LEDs

Plug-in module for up to:

- 4 bus channels populated for different CAN physical layers or LIN
- I/O Piggy

Key characteristics of I/O Piggy:

- Up to 8 digital inputs or 6 digital outputs
- Up to 1 PWM input or 2 PWM outputs
- Up to 4 analog inputs or 2 analog outputs
- Time stamp precision: 5 µs
- Maximum input voltage: ±32 V

Translation of a German publication in 'Hanser automotive', issue 5-6/2011

Figures:

Vector Group

Links:

Home page Vector: www.vector.com

Product information CANoe: www.vector.com/canoe

Product information VN8900: www.vector.com/vn8900

Author:



Oliver Falkner

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 and Product Manager for CANoe.



Case Study

A secure and robust vehicle interface for a next-generation infotainment system



The Customer

Ford Motor Company, a global automotive industry leader based in Dearborn, Mich., manufactures and distributes automobiles across six continents. With approximately 176,000 employees and 80 plants worldwide, the company's brands include Ford, Lincoln, Mercury and Volvo.

The Challenge

Rapidly develop a next-generation factory-installed, in-car communications and entertainment system by leveraging consumer electronics suppliers who were mostly unfamiliar with automotive networks.

Infotainment systems, including mobile phones and digital media players are rapidly evolving to modern passenger cars. Therefore Ford developed a platform for consumer applications with a central driver interface. Many of the application developers are traditionally in the consumer electronics market and have minimal experience with the vehicle network interface and diagnostics standards required by an OEM like Ford.

The Solution

Develop a secure and robust vehicle network interface using a combination of standard and custom embedded software components, and support the rapid development and frequent releases by automating regression testing with CANoe.

The vehicle interface subsystem was developed using Vector CANbedded basic software for the Ford Network Operating System (FNOS), as well as AUTOSAR compliant modules for diagnostics, I/O and memory management, all hosted on an OSEK operating system (osCAN). To support the rapid development, an automated test system was developed using CANoe with the Test Feature Set.

The Advantages

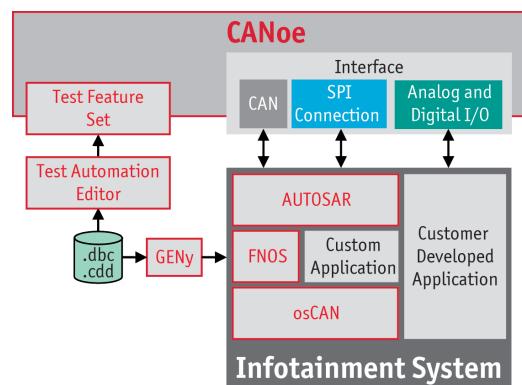
The solution allows Ford suppliers to concentrate on their areas of expertise – developing consumer feature content.

The ECU software approach provides:

- ▶ Isolation between the third-party applications and the vehicle interface.
- ▶ Rapid development by leveraging standard software components for many of the subsystem modules.
- ▶ The development team an opportunity to concentrate their effort on the module specific requirements for diagnostics and power management.

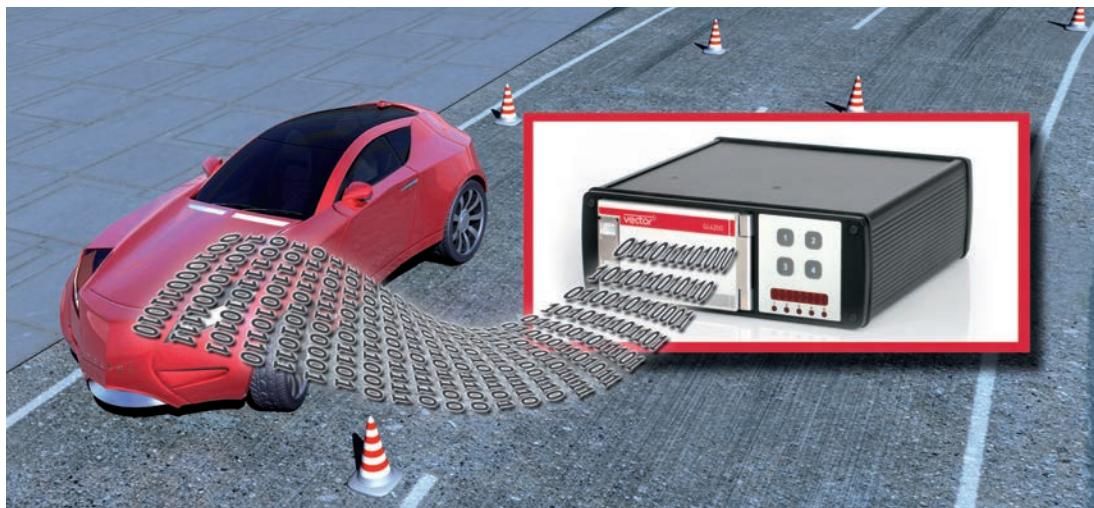
The advantages of using CANoe for automated testing:

- ▶ The testing cycle is decreased from weeks to days.
- ▶ The test coverage for incremental releases to Ford is significantly increased.
- ▶ The solution provides the capability to perform highly repeatable regression test scenarios.
- ▶ The development team is able to eliminate dependencies to other software subsystems by providing simulations to support advanced test cases.



Seamless Logging on Test Drives

Acquire vehicle data reliably with data loggers



When different bus systems are used in vehicles, the effort required for troubleshooting and analysis always increases. Laboratory tests are no longer sufficient to simulate real situations for communication in the total vehicle system. Only extensive test drives in the real environment can deliver the necessary testing depth. In test fleets, data loggers installed in the vehicles are the tool of choice for logging the data traffic of all buses and select I/O lines. This makes it possible to access the test drive data at any time in quality assurance tasks.

Shortly before production maturity, in-depth testing in vehicles is typically conducted in the context of test drives. To achieve the greatest possible test coverage, some of these tests are conducted under extreme environmental conditions. Whether they are winter tests in Finland at -30° C, hot weather tests in Death Valley at over 50° C or week-long drives through the Brazilian rainforest at high humidity and on rough roads, in the end the vehicle and all of its components must operate smoothly. The installed data loggers must be able to withstand these harsh conditions as well. This means that they must be mechanically rugged and operate reliably over a broad range of temperatures.

Various bus systems are used in motor vehicles or commercial vehicles: CAN, LIN and FlexRay. One technical requirement is that the data of all of these buses needs to be logged simultaneously, i.e. time synchronously. The logger must not influence the bus traffic here; it may only observe it. Since the loggers are often permanently installed in test fleet vehicles, and a test series may take several weeks, they must exhibit very low current draw in their quiescent states – another requirement of data loggers. Furthermore, the devices must be ready for operation as quickly as possible, so that the first occurring message can be logged too.

Not only are the loggers typically permanently installed, often they are mounted at very inaccessible points, e.g. under a seat or behind a trim panel in the cargo space, and they may be inaccessible because of other instrumentation. Therefore, it is advantageous if the test engineer can use a UMTS or WLAN wireless connection to read data from a logger. As an alternative, it should also be possible to read data directly via USB or by swapping out the memory medium. To permit clear traceability of certain driving situations to a specific error pattern in later offline analysis and troubleshooting, the test driver has the option of recording audio comments and camera images along with the regular data during the test drive. In parallel, GPS data can be added to the bus communication for geographic reference. After logging, the data is typically converted on the PC, so that it can be analyzed in other programs such as CANoe, CANalyzer or CANape.

Special Data Loggers for Test Fleets

At first glance, it would seem reasonable to use a notebook-based solution for in-vehicle logging. Using a notebook with a suitable network interface should be able to offer all required capabilities,

since logging functionality can be implemented in software. However, commercially available notebooks cannot handle the required temperature range and the system must first be booted, which takes some time – even with fast notebooks. This implies another requirement for data loggers: short startup times. Data must be acquired quickly enough for the first message on the bus to be logged. All of the noted requirements are fulfilled by special fleet loggers such as devices from Vector's GL3000/GL4000 logger product line (Figure 1). Their extended temperature range also makes it possible to use them under extreme environmental conditions. These special fleet loggers also have a real-time clock, ensuring clear time references for the acquired data.

Data Processing

To reduce the volume of incoming data, even during the test drive, these loggers allow users start logging only in response to predefined events. In triggered logging, data is continually written to a ring buffer, so when the trigger event occurs, this ring memory is closed, and the data is saved. Logging is then resumed in a new ring memory. This method substantially reduces data volumes compared to continual logging. Depending on the configuration of the ring buffer, logged data may be available for a time period before the trigger and possibly for a configurable post-trigger time after the trigger occurs. The ring buffer is usually configured with special software (Figure 2).

The special script language Logger Task Language (LTL) can be used to execute complex logging tasks. This can be illustrated by a simple programming example: Creating a classing table during logging. First, the symbolic signals Speed and Brake from a database

are automatically converted to LTL code. The test engineer only needs to add supplemental code for classing with the CLASSIFY operator:

```
VAR Speed = CAN1 DATA 200h [2 3]
Brake = CAN1 DATA 100h [3(0)]
CLASSIFY
    MyClassify COUNT (Brake)
        OVER Speed (20 CLASSES OF 10 BASE 0)
```

In this example, the Variable Speed value is defined in km/h over 20 classes, each class has a width of 10 km/h, and 0 km/h is set as the start value of the first class. This yields the following class distribution:

Class	Value range [km/h]
1	0 - 9
2	10 - 19
....
19	180 - 189
20	190 -

The data of each classing task is saved in text-based results tables that can easily be read into a program such as MS® Excel for post-editing.

Summary

Currently, there are many different data loggers on the market. However, only fleet loggers are suitable for the harsh operating conditions in the automotive field. Loggers should offer a wide range of features that cover the majority of requirements for today's vehicles during test drives. They include a large number of CAN, LIN and FlexRay channels, short start-up times and I/O ports on the logger. UMTS, WLAN, USB and Ethernet offer the necessary flexibility to configure the loggers and transfer the logged data. Fleet data loggers from Vector, with their extended temperature range and durable packaging, equip the test engineer with devices ideally suited for use under extreme environmental conditions.



Figure 1:
Special data loggers for in-vehicle use are rugged and have practical and effective interfaces.

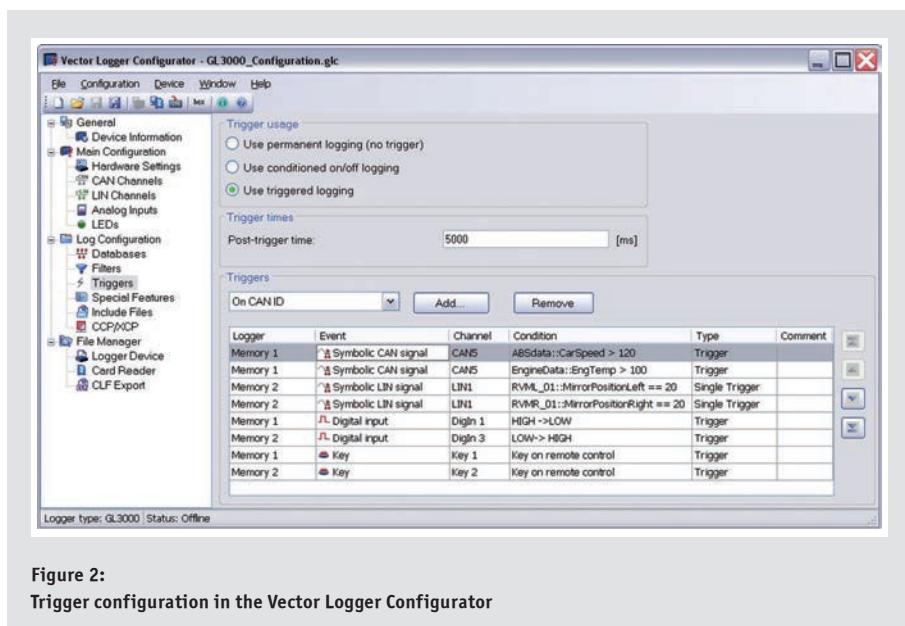


Figure 2:
Trigger configuration in the Vector Logger Configurator



Jochen Neuffer
 studied Information Technology at the University of Applied Science in Esslingen. Since 2002, he has been working at Vector Informatik where he is a Product Management Engineer in the Tools for Networks and Distributed Systems area.

Translation of a German publication in
Automobil Elektronik, February/2011

All Figures:

Vector Informatik GmbH

Links:

Homepage Vector: www.vector.com



Case Study

Rugged Data Logger Endures Tough 24h Race



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.





ARNE BREHMER



CARSTEN KELLNER

A BRIGHT IDEA

'Orange' avionic dataloggers log CAN bus data for flight tests in pursuit of sporadic errors

BY ARNE BREHMER AND CARSTEN KELLNER

Commercial airplanes are entering a new era in terms of the growing number of electronic components they contain. The systems, which are becoming increasingly more complex, need to be tested extensively in the laboratory and in flight trials. Dataloggers can make an important contribution in debugging and error analysis. However, they must fulfill the stringent technical and regulatory requirements of the aerospace industry.

Many requirements need to be considered in the development of modern aircraft cabins. First and foremost, all safety standards must be satisfied. Passengers have come to expect a very high level of comfort and comprehensive inflotainment services, the crew needs ergonomic workplaces, and airline companies need fuel-efficient and economical aircraft. This can only be achieved by the continually growing use of electronic modules, in which the various functionalities are implemented in software.

USE AREAS OF CAN

The CAN bus continues to gain in importance in networking LRUs (line-replaceable units) for cabin and aircraft systems. CAN is typically used in the system areas of air-conditioning, doors, fire detection, cabin management, aircraft galley and waste water.

The galley has a number of noteworthy electrical consumers with its refrigeration compartments, convection ovens, coffee makers and trash compactors. However, the amount of electrical energy that can be generated on board the aircraft is limited. That is why an intelligent power management system is used to prevent galley modules from all running at full load simultaneously. The Galley Master Control Unit does this by controlling all galley electrical consumers over the CAN-based Galley Databus. Power management distributes the available electrical energy within milliseconds, so that food and beverage services can run without interruption while not exceeding the defined maximum power consumption limit. The



communication system is specified by the ARINC standardization organization and is used in the development of new civil aircraft. ARINC 810 defines the physical interface, while ARINC 812 defines services and protocols for coordination of the galley modules.

GROWING COMPLEXITY

Along with an increase in the number of subassemblies, the number of potential sources of error increases as well. The growing complexity is not

only a challenge to development and production, but also to debugging and error analysis efforts. In addition to laboratory tests, wide-ranging flight tests are necessary. Here, the systems are tested, systematically and reproducibly, under the use conditions in which they will later be tested with customers.

These flight test trials are extremely time-consuming and expensive. Airbus, for example, uses five test aircraft to conduct flight test trials and obtain approval for the new A350, and several thousand flight test hours are



conducted. Flight test installations (FTIs) for new aircraft programs are also extensive and complex. However, the initial focus of flight test datalogging is on safety-critical systems such as the Flight Control System and Environmental Control System. Convenience functions in the cabin are not tested until later. The FTI might not log all data, messages and signals of the convenience-relevant systems in their entirety, depending on the specific test strategy. Therefore, it makes sense to utilize additional

small, mobile dataloggers. This enables acquisition of the specific and extensive test data that is needed to enable a detailed analysis. The test personnel can look for specific causes of functional errors and evaluate them in relation to specific operating states.

HIGH LEVEL OF REQUIREMENTS

Equipment that is used to test aircraft must meet the especially stringent requirements of the aerospace industry. Above all, they must assure conformance to requirements related to electromagnetic compatibility as formulated in RTCA DO-160E.

The test equipment must operate without any interference, and it must be constructed so that it does not affect systems aboard the aircraft under any circumstances. Furthermore, it must operate safely and reliably even under the extreme temperature and pressure conditions that occur in aircraft. A rugged housing must reliably protect the system from environmental factors such as dust and moisture, even under harsh ambient conditions. The test units must survive high-voltage surges caused by lightning without damage, as well as vibrations that can occur during tests on the runway, in turbine tests or in flight maneuvers. The same applies to the typical acceleration loads and fluctuations in temperature and pressure. The overall system must be designed so that no small parts such as screws or covers can detach and become lost in the airplane. Ideally, a mounting plate should be used to join the test equipment to the airplane. Paint in a signal color makes it easy to recognize which components are part of the test equipment and are not intended for regular operation.

In addition to the mechanical design, the device's electronics must conform to the special requirements of the aerospace industry – they must operate with absolute freedom from interference and must not influence bus communications under any circumstances. Suitable design measures must be taken to insure that the system does not start any activities on the bus, even in the event of hardware or software errors or total



ABOVE: The GL1020FTE datalogger is a special flight test device that is used to record avionic communication

device failure. An unpredictable external influence also must not result in any effects on the communication technology. Freedom from interference is tremendously important, because otherwise removal of the test equipment would represent a system modification.

LOGGING MEASUREMENTS

Vector has more than 20 years of experience in the testing, simulation and analysis of individual control modules, networks and distributed systems in the CAN field. This comprehensive know-how was an ideal prerequisite for developing the GL1020FTE Avionic Logger (above figure). Serving as a foundation was the compact GL1000 datalogger, which has been used successfully in automotive field testing for many years now. It can log two CAN channels simultaneously plus an additional four analog inputs. The ring buffer lets users choose between long-duration logging and event-triggered logging.

Engineers at Vector intentionally decided to base their logger design on a device series that had proved itself rather than develop an entirely new logger. The GL1020FTE underwent advanced development with regard to its housing, electronics and software so that it would fulfill the especially stringent requirements of aerospace engineering – maximum reliability

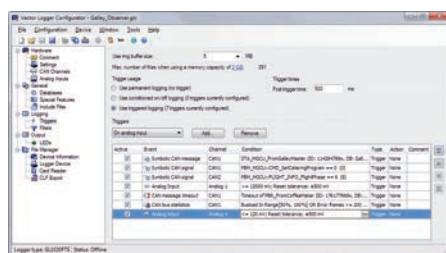
and a very high level of electromagnetic compatibility (EMC).

The datalogger is equipped with a rugged housing, which was specially developed for use in aircraft, even outside of the pressurized cabin. The datalogger is easy to recognize as a piece of test equipment at a glance because of its RAL 2004 Pure Orange paint color. It is mechanically sealed. A welded Gore-Tex membrane which lets air through but not moisture is used for pressure equalization. The Avionic Logger is designed for ambient temperatures between -40°C and +85°C, and it can be operated up to a flying altitude of 36,000ft. It fulfills aerospace engineering requirements related to EMC. The datalogger itself does not emit any significant electromagnetic noise, and its operation is unaffected by typical ambient electromagnetic interference.

FLEXIBLY CONFIGURABLE

In test flights, the GL1020FTE is installed directly at the point at which the CAN communication should be logged. That may be in the cabin, in the cargo hold or in the wing. Its compact dimensions of 208 x 120 x 37mm, including assembly plate, allow the device to be installed in even the tightest of spaces.

The Avionic Logger can simultaneously log two CAN channels and up to four analog inputs. Simultaneous means time-synchronous, so that the user can directly evaluate the time relationships between different events. Vector offers various software tools for this purpose. The logger works on Layer 2 of the CAN protocol and can also acquire error frames, bus load and bus timing (time sequence of messages). The baud rate and the CAN channels to be logged are parameterized as fixed values. The logger has a maximum start-up time of 300ms to ensure that all data is acquired from the start. To



LEFT: Setting up trigger conditions in the Vector Logger Configurator

save data, the logger has a permanently installed SD (HC) memory card with 8GB of memory, or it can be equipped with memory for storage of a maximum 32GB.

The data is logged either continuously or event-triggered. Continuous long-duration logging is used if an entire process needs to be acquired, or if sporadic errors have not yet been localized. This type of logging results in larger volumes of data. In event-triggered data acquisitions, the user defines extensive filter and trigger conditions. In this method, the data is acquired continuously and is constantly written to a ring buffer until the defined event occurs. The user can define the size of the ring buffer and the action to be executed when the trigger event occurs. If this action is to immediately close ring buffer and save its data, then the saved data, which is available for later evaluation, precisely matches the data stream up until the trigger event. If a post-trigger time was parameterized, the signals in the ring buffer are logged after the trigger so that data before and after the trigger time point is saved. Writing to the ring buffer is continually repeated until the next trigger event. This logging principle reduces data volume significantly compared with continuous logging.

User-defined states can be indicated directly on the device by four configurable status LEDs. The logger

has four analog inputs for acquiring analog values such as temperatures. They are led to a plug connector and have an input range of 0–16V. The maximum sampling rate is 1kHz.

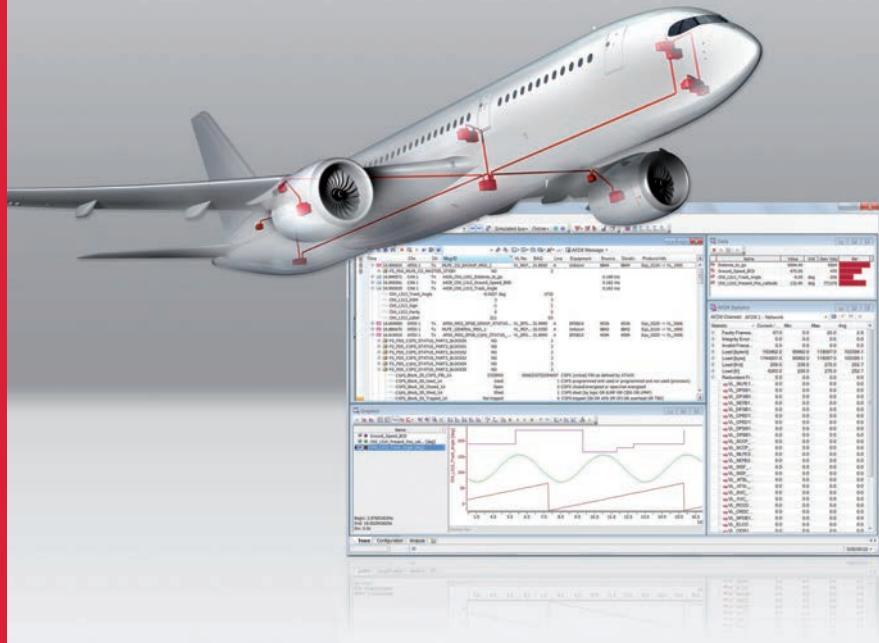
The GL1020FTE is configured using the Vector Logger Configurator (above figure). This user-friendly software can be run on any standard Windows computer. Its connection to the logger is via a USB 2.0 interface. The user can quickly and easily configure logging tasks with triggers, filters and ring buffers using the configurator's graphic user interface. CAN identifiers, symbolic messages and signal values may be selected directly for use as filter and trigger conditions. The Logger Configurator also supports CANdb databases. CANdb has become established as a de facto standard that is broadly used for symbolic description of communication relationships in a communication matrix. The user can work much more efficiently with plain text names and symbols than with cryptic byte values. The data can be read out over the connection to the logger and be converted to different formats, for example BLF, ASC and MS Excel. Various analysis tools such as CANoe, CANalyzer, CANape and CANgraph from Vector, as well as suitable solutions from third-party suppliers, may be used for offline evaluation of the logged data.

The datalogger is supplied with 28V DC over the electrical system and is designed for voltage peaks of up to 33V. In addition, it can buffer voltage drops lasting as long as 200ms, such as those that typically occur when switching over the aircraft's systems from ground power to the generator of the auxiliary turbine (auxiliary power unit). Specially developed for use in aircraft, the datalogger works autonomously in the background, and it does not require any test personnel to be on board to operate it. Together with modern analysis tools, it lets engineers perform efficient error analysis, and it supplies feedback for further development steps. ■



ABOVE: Simple observation of the data traffic and comprehensive network analysis with CANoe.CANAero

Dr Arne Brechner and Dr Carsten Kellner are from Vector Informatik GmbH, based in Germany



Master the complexity of your AFDX and CAN systems

With focused support, development processes in electronic networking can be implemented in a more time-saving and cost-effective way:

- > Use protocol-specific tools to systematically and reliably develop your network design.
- > Find and avoid errors early in development conveniently and cost-effectively.
- > Validate changes made to the software and the communication system – quickly and easily – with regression tests.
- > Exchange data with colleagues and development partners – easily, securely and without overhead – whether from databases, models or test scripts.
- > Benefit from a multi-bus tool approach to developing, testing and implementing gateways.

Improve the efficiency of your overall development process – from design to testing and analysis – with the practice-proven tools CANoe and CANalyzer and our comprehensive services.

- More information and a demo version of CANoe with AFDX® support: www.avionics-networking.com

► Supported bus systems and protocols:
AFDX®, CAN, IP / Ethernet,
ARINC 429, ARINC 825,
CANaerospace, CANopen

AFDX® is an Airbus' registered trademark

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.

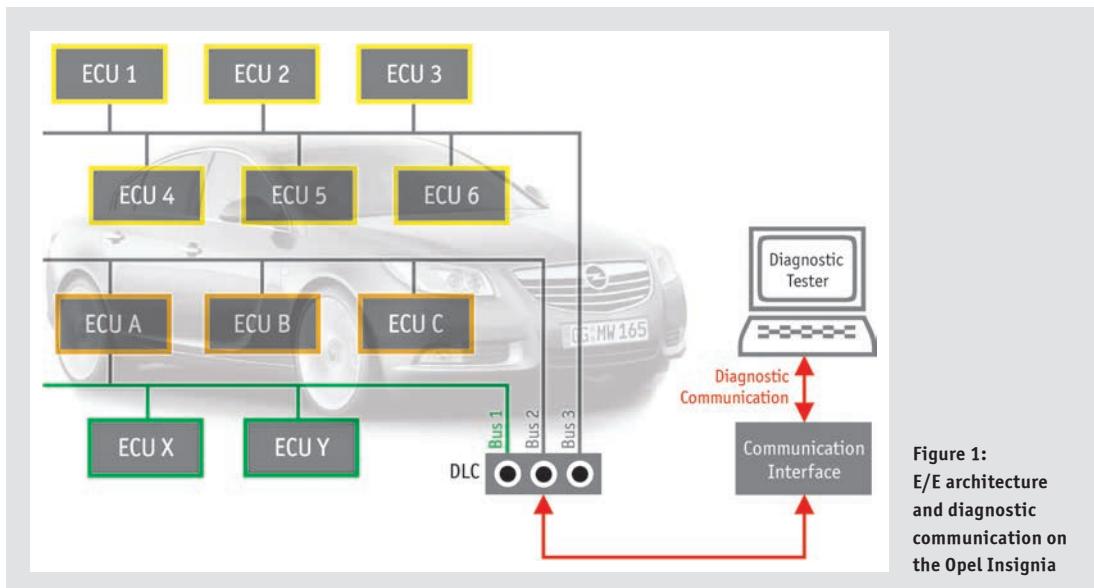
Validation process and tool environment at General Motors Europe

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.



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**).

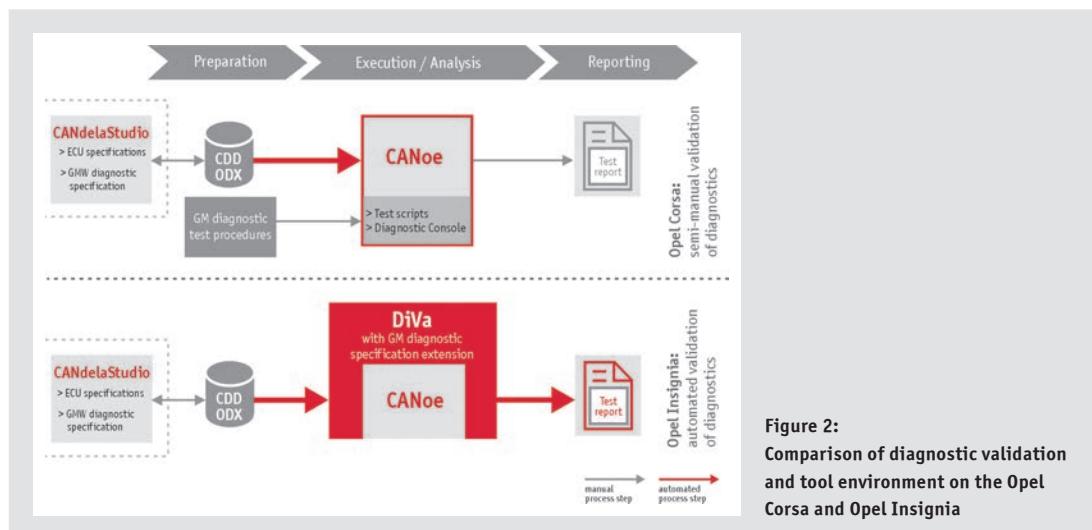


Figure 2:
Comparison of diagnostic validation and tool environment on the Opel Corsa and Opel Insignia

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

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.

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).

Testing depth is further enhanced by including execution of additional non-GM-specific test cases.

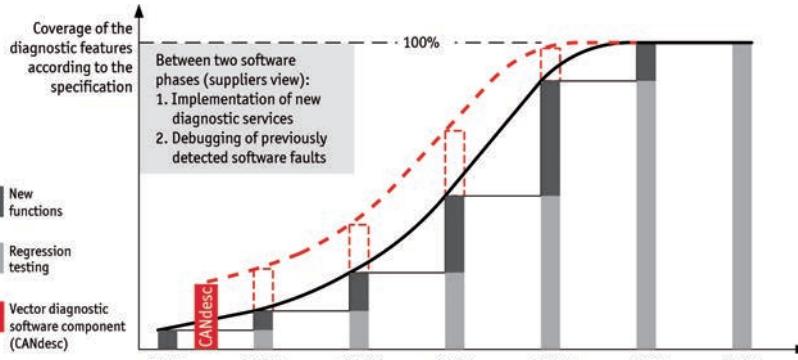


Figure 3:
Scope of diagnostic functions in various phases of ECU development at GME

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 essential, i.e. a test is only as good as its specification. Furthermore, there must be conformity to the

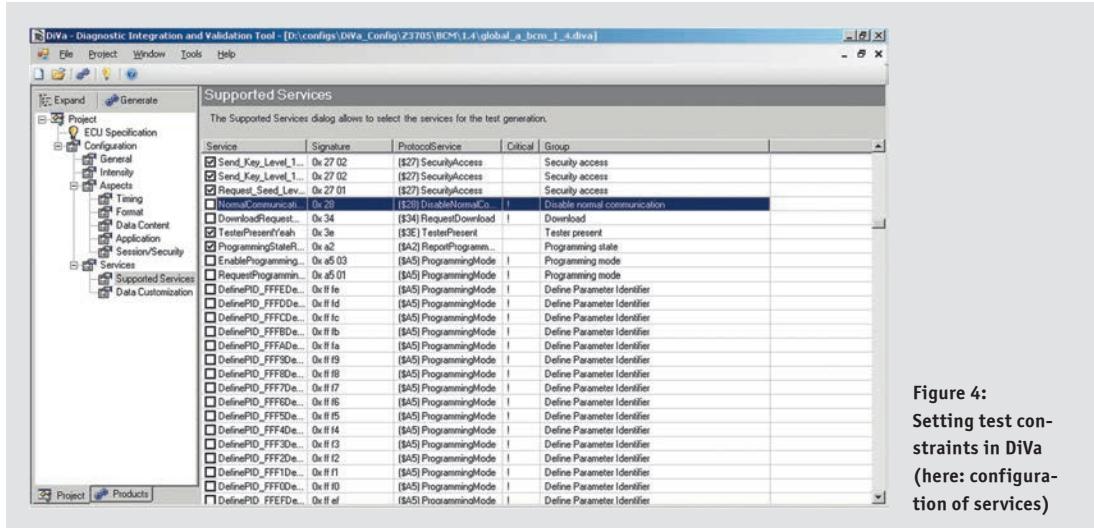


Figure 4:
Setting test constraints in DiVa
(here: configuration of services)

The screenshot shows the DiVa TestModule interface in Mozilla Firefox. On the left, a sidebar lists the 'DiVa TestModule' structure under 'GMU Europe'. The main area displays a test report for '7.4.1.3 TC1302__DID_99_Programming_DateRead: Passed'. It includes a timestamp range from 219.046883 to 219.077044, a table of test steps with descriptions and results, and a list of failed test cases. Below the report is a terminal window titled 'ECC.ASC' showing diagnostic log entries.

Timestamp	Test Step	Description	Result
219.046883	Step 1	Send a \$1A message with each \$dataIdentifier supported and verify proper response (test data formatting). (ECU_Identification__DID_99_Programming_DateRead)	-
219.077044	Step 1	Positive response received as expected.	pass
219.077044	Step 1	Response format correct.	pass
219.077044	Step 1	Received data for service parameter "DATA" in defined range (see Bcd_4Byte)	pass

Failed Testcases:

- 1.1.5 TC5 DID 90 Vehic
- 1.1.35 TC35 DID 41 EO
- 1.1.36 TC36 DID 41 EO
- 1.1.39 TC39 DID 42 Cor
- 1.1.42 TC42 DID 44 Cos
- 1.1.45 TC45 DID 48 Cos
- 1.1.48 TC48 DID 49 Cos
- 1.1.51 TC51 DID 50 Cos
- 1.1.54 TC54 DID 51 Cos

Figure 5:
Automatically
generated test
report in DiVa

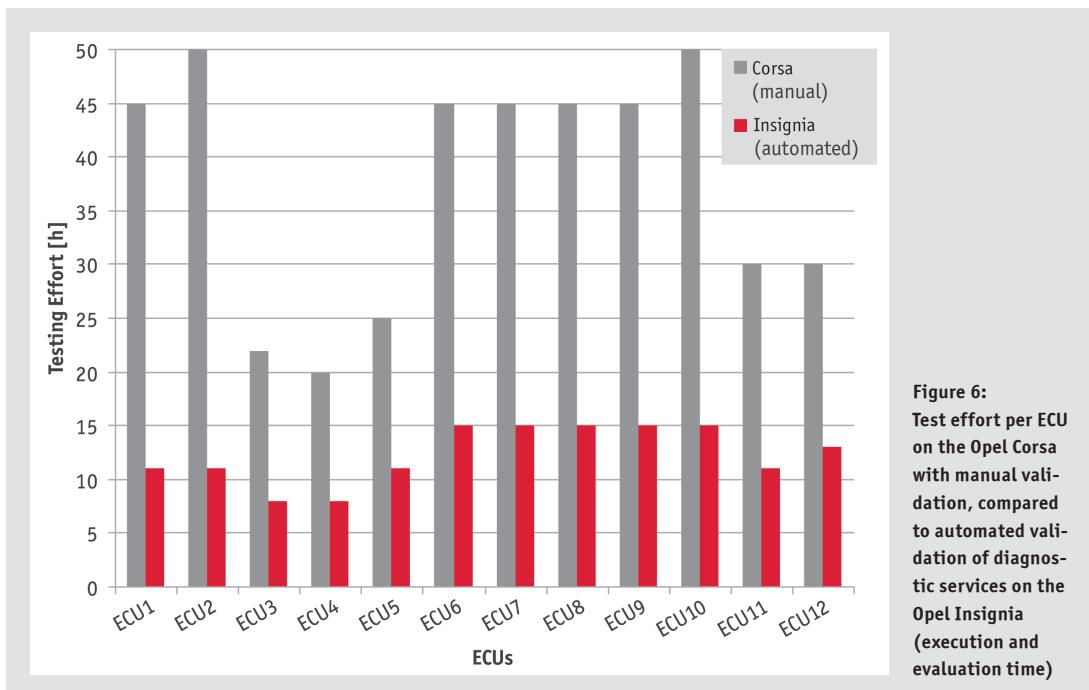


Figure 6:
Test effort per ECU
on the Opel Corsa
with manual vali-
dation, compared
to automated vali-
dation of diagnos-
tic services on the
Opel Insignia
(execution and
evaluation time)

- 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.

Translation of a German publication in ATZ elektronik, 6/2008

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 Petri

studied Computer Science at the Vienna University of Technology. He earned his doctorate degree in Computer Science, also at TU Vienna. Dr. Petri 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.

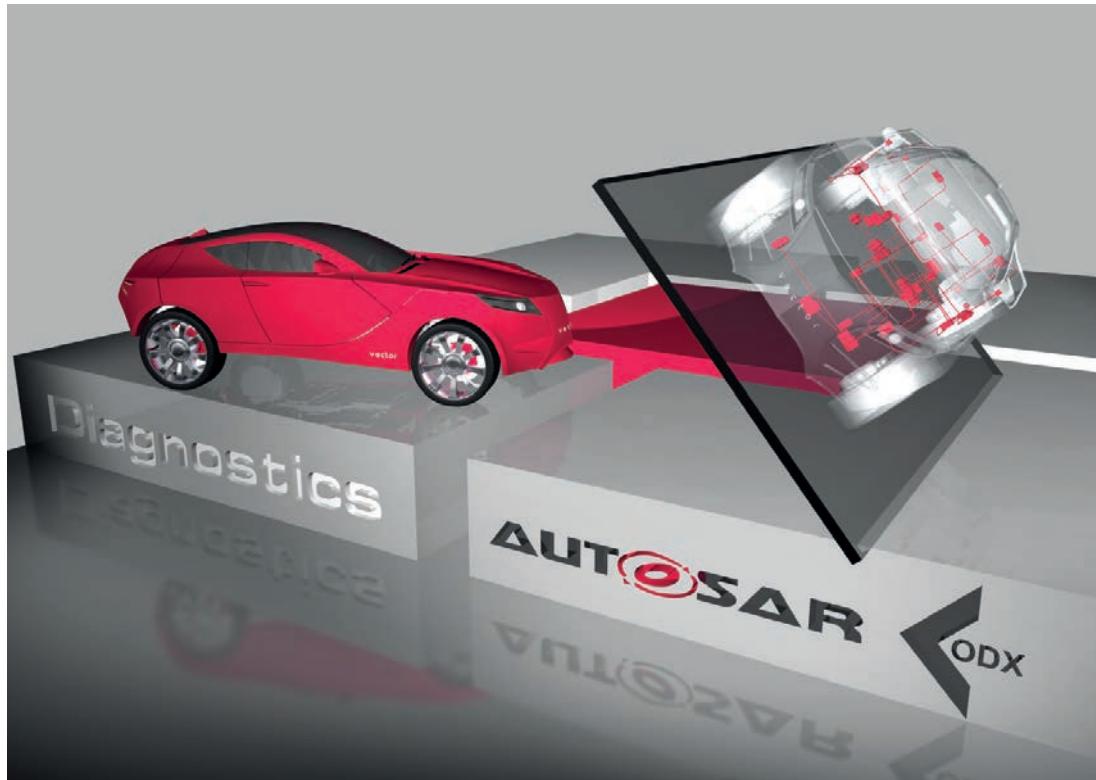


Christoph Rätz

studied Computer Engineering at the University of Cooperative Education at Stuttgart. He works at Vector Informatik GmbH in Stuttgart as a global product line manager for the Vehicle Diagnostics product line division.

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)

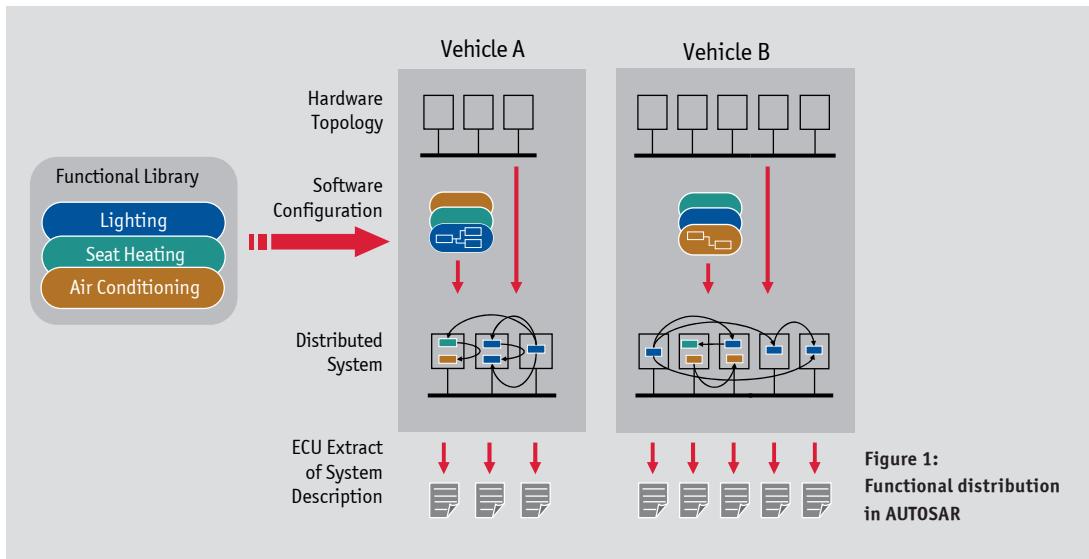


Figure 1:
Functional distribution
in AUTOSAR

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 increased 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

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 Manager) 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.

	SID	DID	Data
Request	22	FF EE	
Response	62	FF EE	AA BB CC
AUTOSAR Version	2.1	3.x	4.x

Figure 3:
DCM in different AUTOSAR versions

A standardized interface and various debounce algorithms for

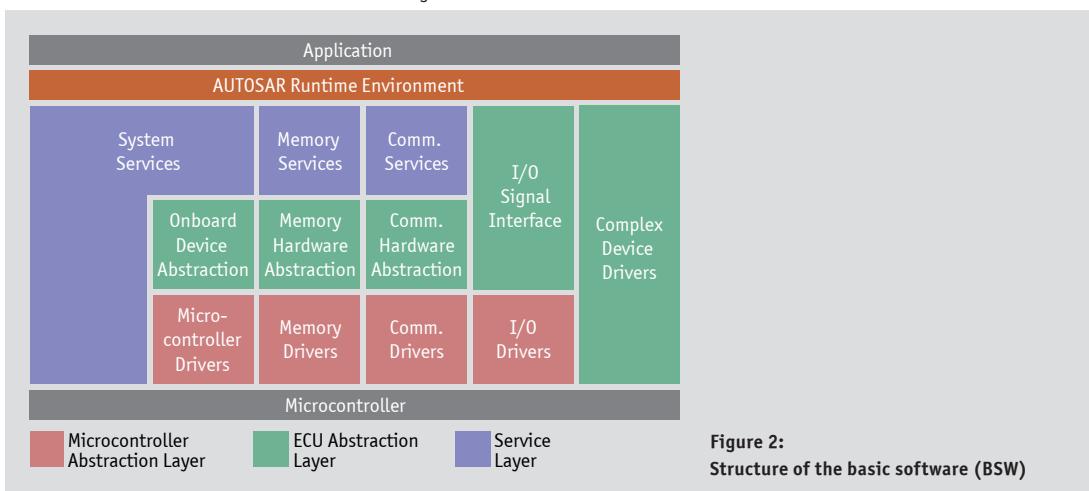


Figure 2:
Structure of the basic software (BSW)

Technical Article

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.



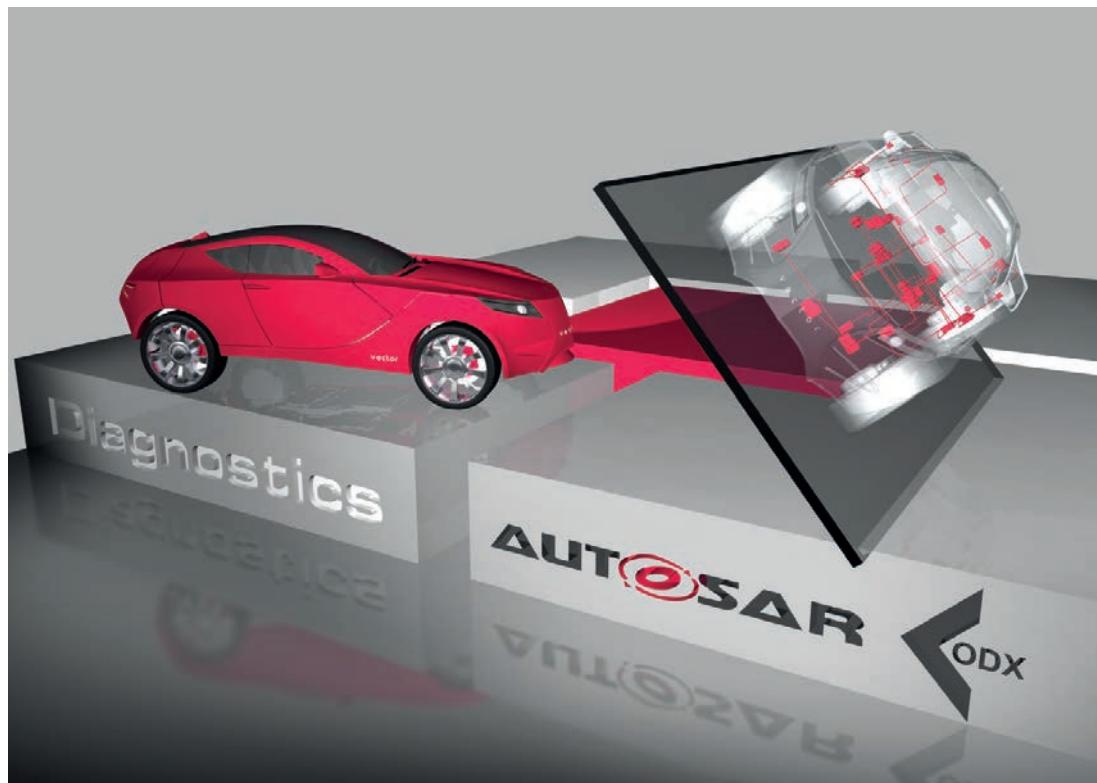
Oliver Garnatz (Dipl Ing. (FH)) is employed at Vector Informatik GmbH as a product manager in the Embedded Software Components area. He is a member of the Automotive Diagnostics area of ISO and the AUTOSAR area.



Christoph Rätz (Dipl-Ing. (BA)) graduated in Computer Science at the Cooperative State University of Stuttgart. He is the Global Product Line Manager 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 submodels (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).

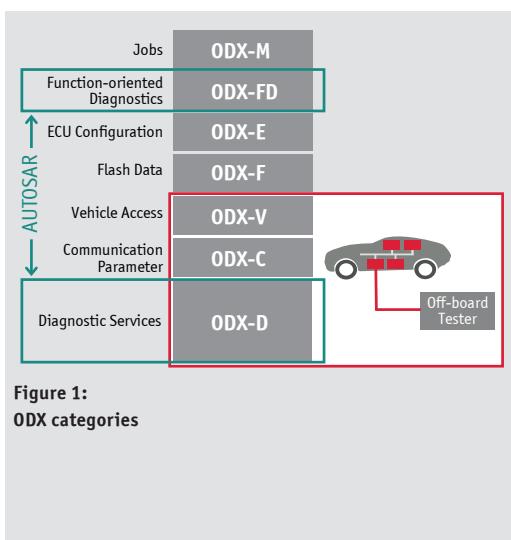


Figure 1:
ODX categories

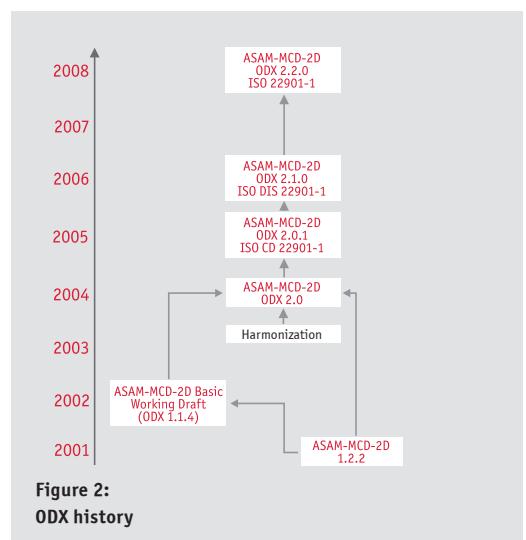


Figure 2:
ODX history

- > 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.
- > 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.

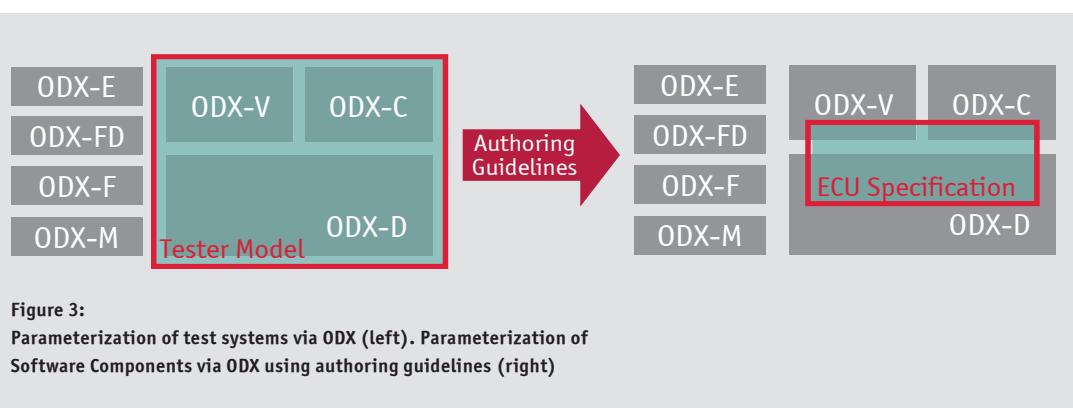
In summary, the following picture emerges: ODX was designed to fulfill the requirements necessary for parameterizing test systems (see **Figure 3, left**). However, parameterization of software components assumes that the possible degrees of freedom are limited to the degree required by a specification (see **Figure 3, right**). This can be achieved by means of authoring guidelines.

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.

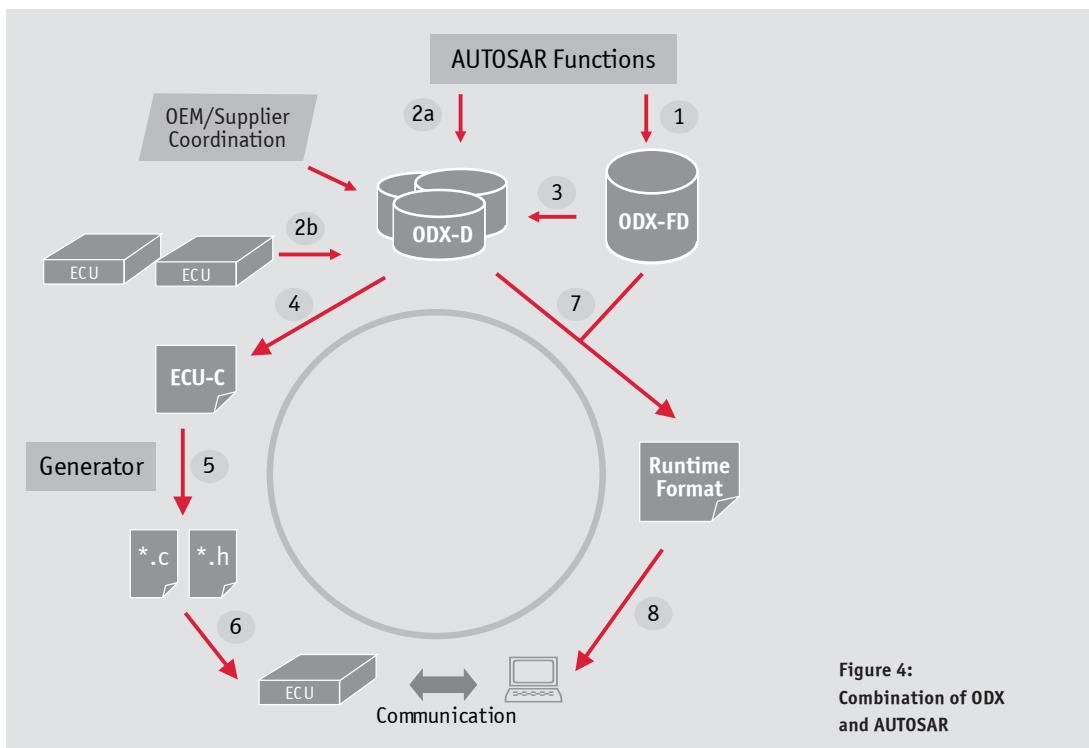


Figure 4:
Combination of ODX
and AUTOSAR

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 (Dipl Ing. (FH)) 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 (Dipl.-Ing. (BA)) graduated in Computer Science at the Cooperative State University of Stuttgart. He is the Global Product Line Manager of the Diagnostics product line at the company Vector Informatik GmbH in Stuttgart.



►►► Diagnostics

Increase your Development Efficiency even more!

In diagnostic development, benefit from solutions that span the entire development process. Increase your efficiency and improve your quality by using optimally tuned and coordinated tools.



The CANdela product family supports you by:

- > User-oriented specification of diagnostic data
- > Single source principle – re-use of created diagnostic data in all process steps
- > Auto-Code for the diagnostic software in ECUs
- > Automated validation of the individual ECU software
- > Diagnostic tester perfectly matched for your application area
- > Standards and open interfaces, e.g. ODX

With the Vector diagnostic tools you reduce the effort significantly and get much faster results – at the highest quality level.

► More information: www.car-diagnostics.com/efficiency/

Get more info:



vector

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.

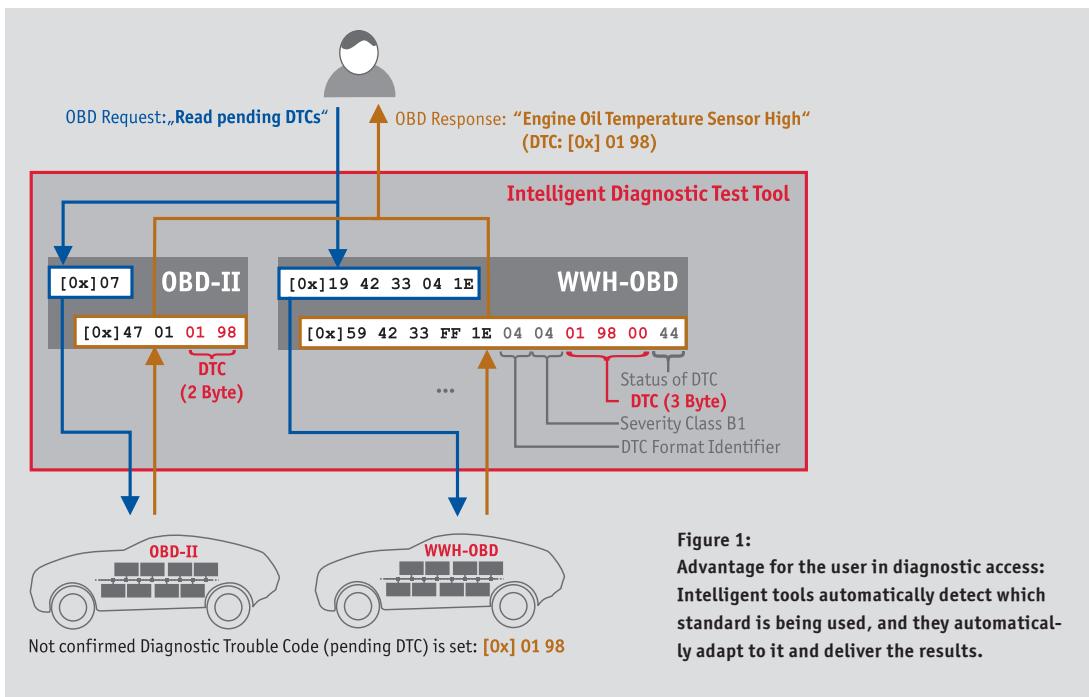


Figure 1:
Advantage for the user in diagnostic access:
Intelligent tools automatically detect which
standard is being used, and they automatically
adapt to it and deliver the results.

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 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 *reportWWHOBDTCTByStatusMask* [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!

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



Helmut Frank (Dipl.-Ing.) 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.



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 network interfaces (PassThru API)

Remote diagnostics with Indigo Remote quickly gives you an expert opinion and shortens diagnostic and repair times. This reduces costs and increases customer satisfaction.

► More Information: www.vector.com/remote/

More Information:



vector

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

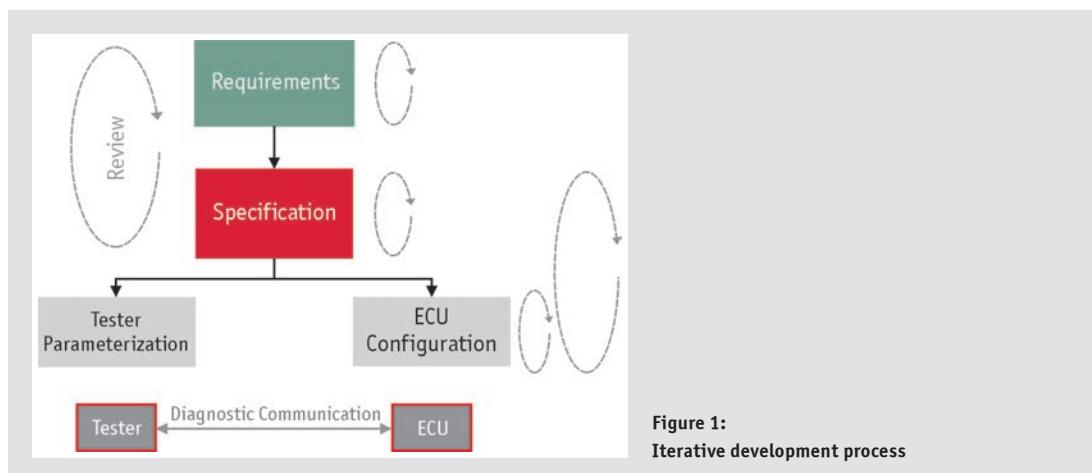
Today, AUTOSAR (AUTomotive Open System ARchitecture) is the reference architecture for ECU software in the automotive industry. AUTOSAR standardizes the description of individual component or vehicle functions and the description of the overall system.

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.

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

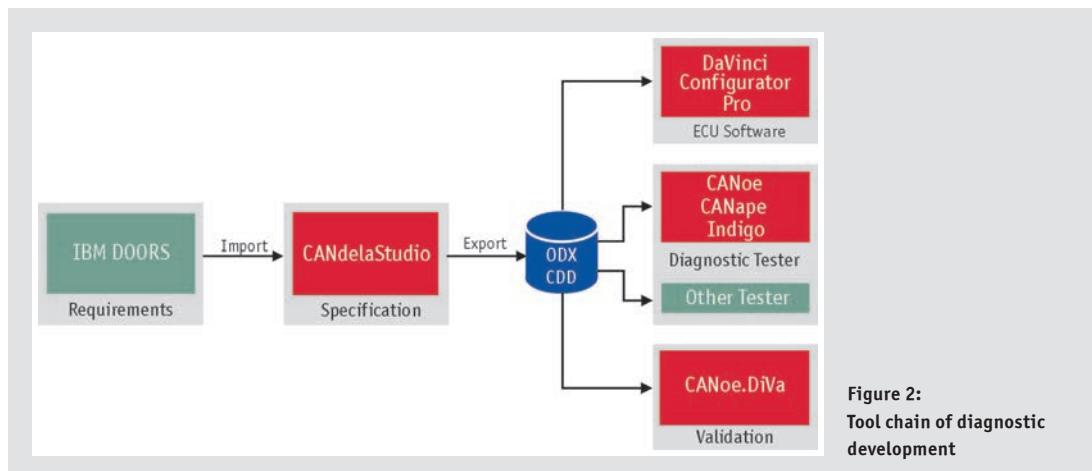
During diagnostic development, the following tasks are performed, which are supported by the tool chain shown in **Figure 2**.

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.

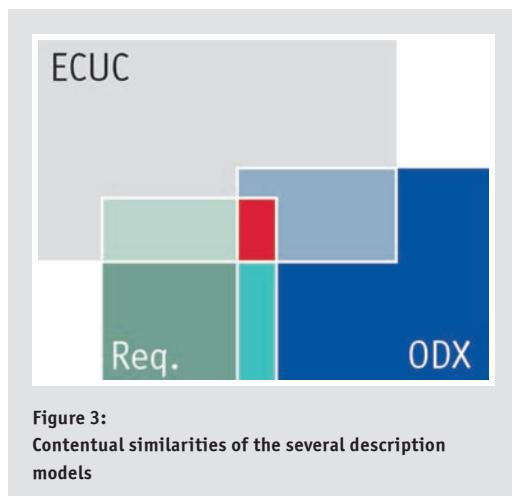
The finished diagnostic specification serves as the input to subsequent steps in the process chain. CANdelaStudio saves the native diagnostic specification in cdd format, and an ODX file can also be exported at the press of a button.

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.

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.

>> Contact information of the Vector Group:
www.vector.com/contact



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 diagnostics 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.

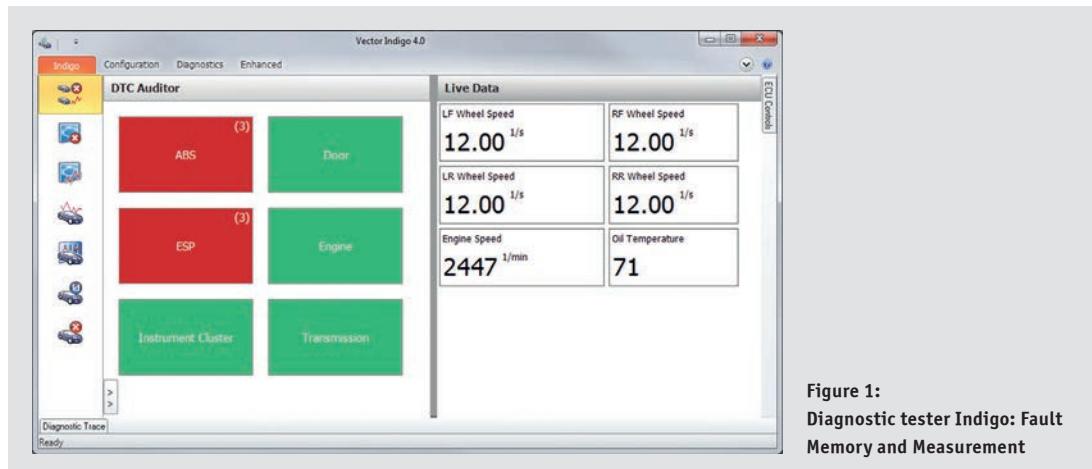


Figure 1:
Diagnostic tester Indigo: Fault Memory and Measurement

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.

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.

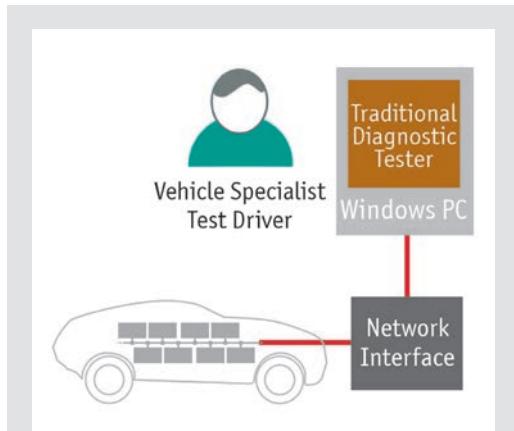


Figure 2: Classic Diagnostic Tester

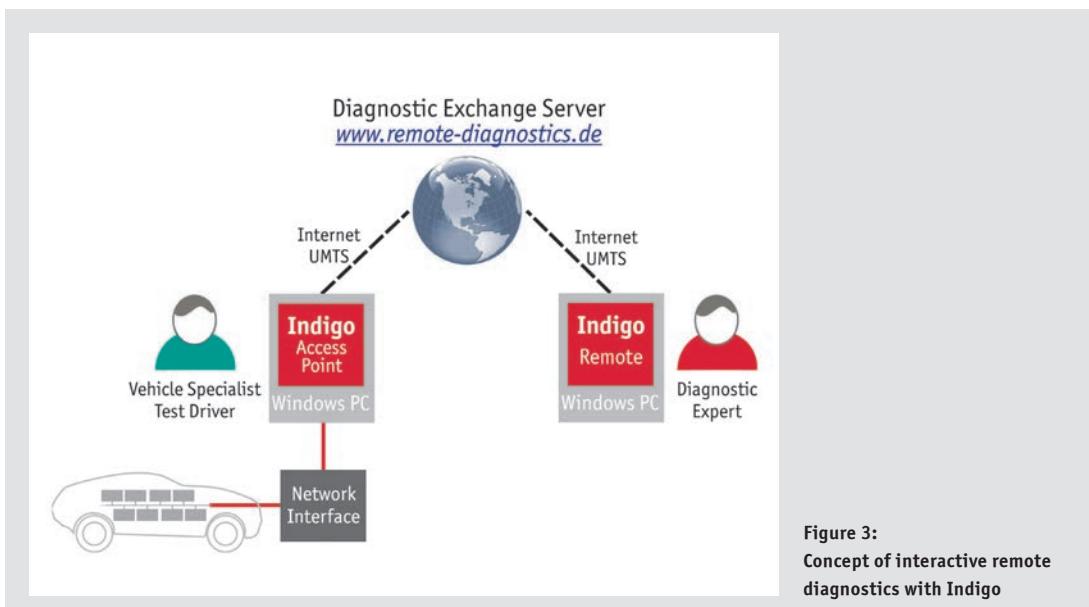


Figure 3:
Concept of interactive remote diagnostics with Indigo

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.

>> Contact information of the Vector Group:

www.vector.com/contact



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.



AdaptiveDrive in the new X5 utilizes sensors and performs computations to acquire data on vehicle speed, steering angle, longitudinal and lateral acceleration, body and wheel accelerations as well as their heights. The swivel motors of the stabilizer bars and electromagnetic valves of the shock absorbers are controlled based on this information. This provides full-time control of body roll and damping based on the given driving situation. The new BMW X5 is the first vehicle in the world to utilize FlexRay technology.

Mr. Peteratzinger, why FlexRay and why now?

Peteratzinger: That was a strategic decision. We have already reached the limits of reasonable bus loads with CAN, and would otherwise need to install multiple CAN buses, and not just for high-end products. Our analyses indicated that up to eight CAN buses would be needed for just the powertrain and chassis areas in the next 7-series car. But we do not want to implement even more sub-buses; at some point they become unmanageable. Therefore, some years ago BMW decided to build up its development experience with FlexRay. We now have a foundation for future electrical systems and will expand them to include additional vehicle systems in upcoming car models. If we did not start on this now, we would have had to utilize a large number of CAN systems and sub-buses to carry us through, and this would last for many years due to the development cycles for car models.

Steiner: Another criterion of course is transmission rate. In the current application sensor signals are acquired and processed both by the central ECU and by the satellites. On the one hand, this decentralized approach leads to an increased demand for communication, and on the other hand shorter cycle times are needed on this and future architectures due to their separation of sensors, control system and actuator. In addition, many of the new functions and architectures place stricter requirements on real-time capability and communication availability. This can only succeed with a bus system like FlexRay.

Translated by Vector Informatik

Use of XCP on FlexRay at BMW

This fall the first FlexRay production application will hit the streets. The Munich-based automotive manufacturer BMW is introducing the innovative bus system for the first time on its new X5 vehicle. From December 2004 to January 2006 tool producer Vector Informatik worked together with BMW on the FlexRay solution. FlexRay experts Martin Peteratzinger and Florian Steiner of BMW and Roel Schuermans of development partner and software provider Vector discussed their experiences in HANSER automotive.

Why did BMW chose suspension control as a pilot application?

Peteratzinger: FlexRay is a completely new development, and we had no knowledge base from other production projects. In this case, the first step was to build up this knowledge. It was therefore important for BMW to be able to quickly apply this acquired knowledge and implement necessary changes during development. In a system such as an engine controller the adaptation effort would be considerable due to the large number of interfaces. This new suspension system has a well-defined and limited functionality. In a small team, together with our ECU suppliers and development partners such as Vector, we were able to make decisions and introduce modifications over short decision-making paths. Furthermore, aspects of this application made it possible to implement many new FlexRay features meaningfully.

When did you decide to eliminate CAN entirely as a fall-back solution in this project?

Peteratzinger: That was done in a relatively early phase of the project in the summer of 2004. After we had successfully started up FlexRay as a bus system to network the five ECUs in a stable manner and had all identified and assessed unresolved risk issues,



Martin Peteratzinger, Graduate Engineer (University of Applied Sciences), develops electronics in the vehicle dynamics area of the BMW Group and has functional responsibility for the FlexRay application.

VECTOR'S CANAPE



Optimization of ECU parameters with CANape

the decision was made to eliminate CAN altogether beginning with the next hardware level. In the process this raised the question: How would we calibrate the application? Initially calibration was still performed via CAN, but in the end this fall-back level was dropped. Therefore we first developed a CCP-CAN-FlexRay gateway that converts CCP messages to FlexRay and in the reverse direction too. In the ECUs we also "restructured" the CCP module for FlexRay. This made it easier to continue utilizing CANape with CCP (CAN Calibration Protocol). But that too was just a transitional solution, since first it is necessary to create such gateways, and then they need to be maintained for a number of years. Therefore it was important to find a product that would not just work exclusively in one project, but would be available to the entire market. In the ASAM working committee XCP on FlexRay was driven by Vector with the support of BMW and attained specification status in February 2006.

Schuermans: Technically we had gotten a handle on the "measurement & calibration via XCP on FlexRay" concept relatively quickly. For Vector though it was clear that we had to turn XCP on FlexRay into a standard as quickly as possible. ASAM has been working on XCP for quite a long time now. XCP itself was finalized in 2003. The aim of the standard is to only require adaptation of the underlying transport protocol. The specific solution needs at BMW allowed us to implement both the XCP stack in the ECU and on the tool side CANape as the XCP Master very quickly.

XCP on FlexRay was standardized in February. What was involved in this process?

Schuermans: Of course the work in ASAM requires a certain amount of time: First a project application is submitted, then a working committee is formed, and finally a draft is developed. XCP is split up into several subdocuments. Part 1 is an overview of the protocol family, XCP features and

basic definitions. Part 2 of the document describes the XCP command set as a bus-independent protocol layer. Whenever a new Transport Layer is added, as is being done now with FlexRay, a Part 3 document is created.

What does all this discussion about dynamic bandwidth control really mean?

Schuermans: A part of the XCP on FlexRay specification defines dynamic bandwidth control. Since the XCP protocol is essentially a Master-Slave communication protocol, the XCP Master can distribute XCP Slots to individual Slaves depending on how much bandwidth the Slave needs. Dynamic bandwidth management requires that the Master knows which slots are available for this purpose. That must be a part of the FIBEX file.

Why is that good?

Steiner: Compared to the potential bandwidth of FlexRay the current bus load is still very small.

We therefore had the luxury of making three XCP slots available to each ECU: One to communicate from CANape in the direction of the ECU, and two to send measurement data from the ECU in the direction of CANape. That means that 15 slots are used just for XCP. In the next car models the bus will be utilized more intensively with more ECUs. Then we will no longer have this freedom. All ECUs must then share one slot or just a few slots for XCP.

The only limitation here is that it will no longer be possible to calibrate all ECUs simultaneously. However, this is of no consequence in practice, because developers generally only calibrate their "own" ECU. With dynamic bandwidth allocation this can be done very quickly.

Florian Steiner,
Graduate Engineer (University of Applied Sciences), develops electronics in the vehicle dynamics area of the BMW Group and as a FlexRay development expert he is responsible for production-level ECU development.



That is XCP will remain in the ECU?

Steiner: Yes! Although there is no need for direct measurement on the FlexRay bus while it is in service, the XCP module will be retained in production versions of the ECU. Diagnostics of the FlexRay ECUs is performed via OBD, the standard diagnostic interface in the vehicle. XCP plays a central role in testing and validating ECU functions and is therefore a component of the production software.

Doesn't that pose a risk? Certainly there are people who might want to reprogram the chassis.

Schuermans: XCP, and I am not just referring to XCP on FlexRay here, has a so-called Seed&Key security mechanism. XCP is of course based on a Master-Slave principle.

The Master must ask the Slave for a "Seed", and this is used to compute an associated key. The Master does not grant the Slave access until the correct key is obtained.

Peteratzinger: Our hardware supplier has also installed another software protection mechanism. So protection is twofold.

What did Vector bring to the table as a partner with FlexRay?

Peteratzinger: Important was this. We need to measure and calibrate signals internal to the ECU. It might have been possible to turn to other suppliers for a solution. In this context there were many open

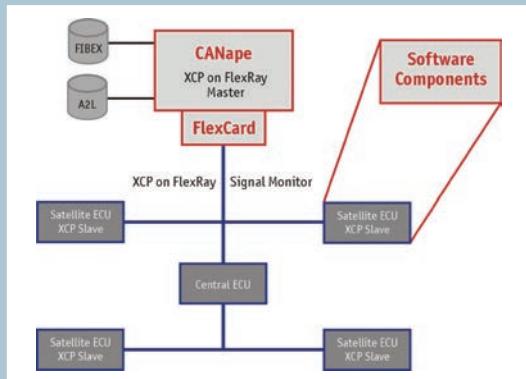


Roel Schuermans, Graduate Engineer, is Senior Product Management Engineer at Vector Informatik for the "Measurement & Calibration" product line. As an expert in the FlexRay protocol he has been involved in the development of XCP on FlexRay in the ASAM working committee up to its standardization and worked on its implementation in CANape.

issues. During HANSER automotive's FlexRay product Day in 2004 we met with experts from Vector, discussed these issues and formulated our requirements. Although no standard existed yet at that time, Vector showed great interest in working with us, and wanted to implement a solution for us based on XC on FlexRay and then make it an official standard in the ASAM working committee. It was also a good fit, because Vector's CANape product was already being used in the existing BMW tool environment, and this meant that our developers already had experience with this tool.

Mr. Peteratzinger, in retrospect what is your assessment of this joint effort?

From our perspective the collaboration with Vector was very good. Vector handled most of the standardization work and the first implementation. So I would like to express my sincere thanks to Vector Informatik. Calibration of ECUs with CANape and the XC Stack performed flawlessly from the start, requiring just minimal modifications.



Application of suspension control system with CANape as the XCP on FlexRay Master

© automotive

FIRST FLEXRAY APPLICATION AT BMW

The application is an active suspension control system. Each of the four shock absorbers has a valve with dedicated electronics to control it. The valve is able to generate different damping properties, since its continuously variable aperture determines how much and how quickly shock absorber oil is pushed from the upper oil chamber to the lower one. The central control module is interconnected with the damper modules, the so-called satellites, via FlexRay. Different than in previously implemented suspension systems, while driving the vehicle the shock absorber characteristics can be independently adapted to the specific driving situation at all wheel suspensions. The satellite electronics controls excitations at the individual wheels, while the central ECU with its higher-level algorithms monitors the overall effect on the chassis, and its control system intervenes when pitching or rolling movements occur. Direct access to internal ECU parameters is necessary to tune the suspension system in driving trials and to assure system functions in system integration.

A special measurement and calibration protocol is needed for this XC on FlexRay. In efforts toward the standardization of XC on FlexRay in February 2006, Vector helped to give shape to its fundamental principles on the ASAM working committee and implemented an application concept in the Vector measurement, calibration and diagnostic tool CANape. This solution provides access to all relevant ECU parameters over the FlexRay bus. The ECUs of the suspension system transport all functional control data in the static area of the FlexRay protocol. Although XC communication is by definition allowed in both the static and dynamic areas, BMW only utilizes the dynamic area of the FlexRay protocol for XC. It is also used for non-time-critical network management messages and the Transport Layer mapping of FlexRay (i.e. diagnostics, coding, flashing). Due to strict time requirements of the control functions, the cycle time is 5 ms. Scheduling is not just set for the currently implemented application, it will also be applied in precisely the same form in the next planned vehicle startups at BMW.



ECU Calibration

Your efficient all-round solution for measurement, calibration and diagnostics

Universal tool support simplifies your calibration of ECUs. The versatile CANape tool lets you cover all application cases effortlessly:

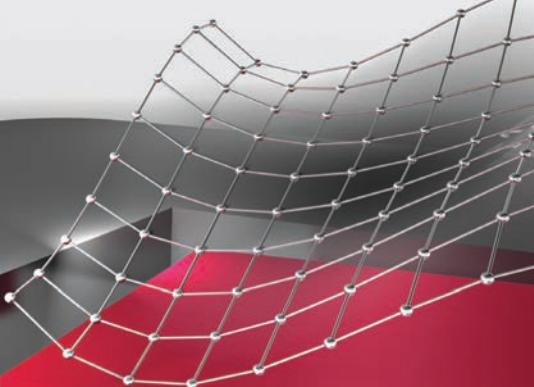


- > Quickly and reliably capture measured data from various sources – synchronous and time-precise. Whether via CCP, XCP-on-CAN/FlexRay/Ethernet or from external test equipment
- > Conveniently calibrate the parameters of your ECU algorithms, either online in the ECU or offline in the Hex file
- > Easily manage large amounts of calibration data – with full traceability at all times – even on large project teams
- > Simplify your tool environment by seamlessly integrated diagnostic services and flash solutions
- > Benefit from a universal tool chain with extensive rapid prototyping capabilities and MATLAB/Simulink integration



Vector supports you from functional development to production-ready ECU, in the laboratory, on the test bench and during driving trials.

► **Information and downloads:** www.ecu-calibration.com



XCP-on-FlexRay at Audi

AUTOSAR-compatible XCP software modules for FlexRay ECUs



To adjust parameter values in FlexRay ECUs, Audi calibrates them via XCP-on-FlexRay. One of Audi's requirements was AUTOSAR compatibility of the XCP embedded software modules in the ECUs. For this purpose Vector modified both the XCP master and slave software so that Audi's electronic developers could perform efficient measurements and calibrations. This was possible thanks to dynamic allocation of the XCP bandwidth for FlexRay.

Starting in 2009, Audi will be implementing the FlexRay communication bus on its next generation of sporty luxury limousines. Since FlexRay – compared to CAN – offers a significantly greater bandwidth of 10 MBit/s. Many electronic chassis and driver assistance systems are connected to this deterministic and time-triggered bus system. For Audi developers, this decision meant that several thousand parameters needed to be directly parameterized via an AUTOSAR FlexRay stack. Compared to CAN, more than twice as many measured values can be acquired simultaneously using XCP-on-FlexRay. Furthermore, it is possible to transmit large quantities of data with a higher throughput.

XCP on FlexRay

A laboratory model by itself is of limited use in determining the parameters of a control algorithm. Although the algorithms of the functions are permanently stored in the ECU-specific software

program, parameter values such as characteristic maps, curves and values need to be recorded and optimized in measurements made on the test bench and in driving trials. Audi engineers tune their chassis and assistance systems in the framework of ECU calibration, and they then load the parameter set files into the ECUs' application memory.

To calibrate ECUs centrally from a single master and via a uniform interface – over the entire course of development – a standardized measurement and calibration protocol is necessary. In 2003, ASAM (Association for Standardization of Automation and Measuring Systems) defined the universal measurement and calibration protocol, XCP, to serve this purpose. It is a logical advancement of CCP (CAN Calibration Protocol) [1]. Communication via XCP is executed by the Master-Slave principle. An XCP software module integrated in each ECU serves as a slave. The greatest advantage of the XCP protocol is that it offers separation of the transport and protocol layers. The protocol layer is the same on all bus systems, regardless of whether

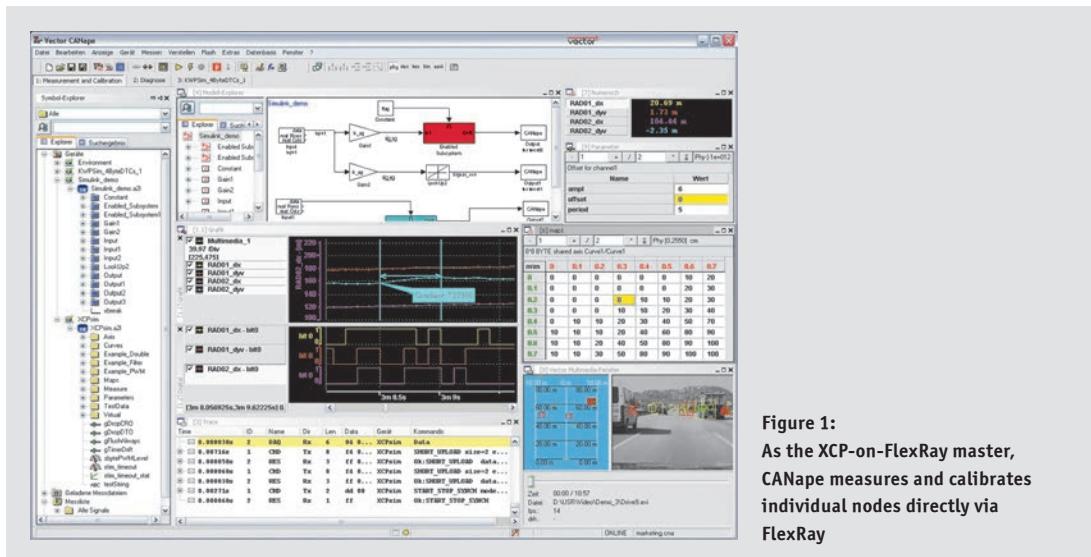


Figure 1:
As the XCP-on-FlexRay master,
CANape measures and calibrates
individual nodes directly via
FlexRay

they are CAN, FlexRay, Ethernet or SPI/SCI. In February 2006, ASAM officially released Version 1.0 of the Transport Layer specification for "XCP on FlexRay".

On earlier CAN projects, the Audi development team had already worked with XCP and CANape, the all-round tool from Vector for ECU measurement, calibration and diagnostics (**Figure 1**). CANape has supported an XCP-on-FlexRay interface since 2005. Audi decided to source both the XCP master (CANape) and the protocol and

transport layer software modules for the slaves using XCP-on-FlexRay from a single supplier.

XCP integration in the AUTOSAR model

Audi integrated the XCP software modules in the ECUs of various suppliers. Even after the ECUs calibration is over, the XCP software modules shall remain available. This makes efficient memory

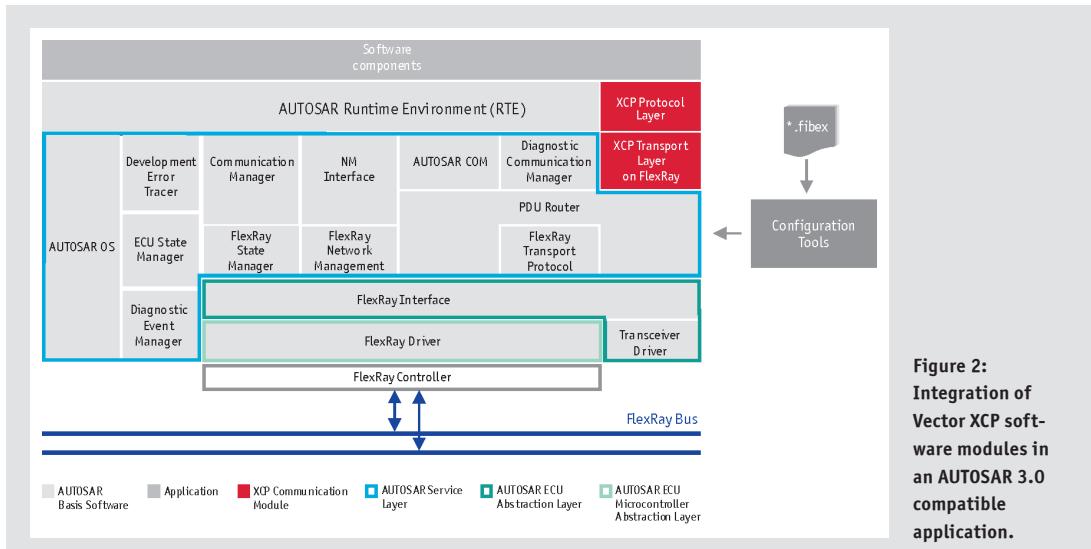


Figure 2:
Integration of
Vector XCP soft-
ware modules in
an AUTOSAR 3.0
compatible
application.

utilization and minimal execution time essential. In addition, the XCP software modules have to be AUTOSAR-compatible. Vector implemented this requirement on the XCP transport layer such that it is placed above the AUTOSAR communication stack (FlexRay or CAN) using the PDU router (**Figure 2**). In the integration phase, the two XCP software modules are configured with the help of the GENy configuration tool and a network description file in FIBEX format.

Dynamic management of FlexRay bandwidth

Since AUTOSAR compatibility is required for the XCP-on-FlexRay software modules, this means that the PC-supported master must perform special tasks. During ECU calibration, the XCP master and slaves exchange FlexRay messages. These frames contain either Command Transfer Objects (CTO) with control commands or Data Transfer Objects (DTO) with measured or stimuli data. When such an XCP object is transmitted to the master (**Figure 3**), the “XCP transport layer” transfers the data to the PDU router and thereby to the “FlexRay interface”.

Because of the requirement for AUTOSAR compatibility, this transfer must be made in the form of an AUTOSAR-conformant PDU (Protocol Data Unit). Since the PDU originates from the XCP module, it is called the XCP-PDU. The FlexRay interface completes the received XCP-PDU by adding its own specific information in the

form of a PCI header (Protocol Control Information), thereby forming an L-PDU (Data Link Layer PDU), which in turn is routed to the “FlexRay driver”. That is how each participating software module completes data received with module-specific information, making it possible to reconstruct the data at the receiver. At the end of the chain, the FlexRay controller transmits the XCP data as a frame within a FlexRay slot (time window). Per the XCP specification these frames must exclusively contain XCP data. Therefore, in the cross-system FIBEX network description file, some slots in the FlexRay schedule are exclusively to be reserved for XCP-PDUs and cannot be combined with PDUs issued from the application.

For the control commands (CTOs), two individual XCP slots are sufficient for all ECUs thanks to the slave-referenced node address (node address for XCP: NAX). The exact number of DTOs needed for the measured data or stimuli data depends on the specific measurement being executed and may vary widely over the course of a calibration. So the need for XCP slots also varies for each ECU.

To ensure that Audi engineers can efficiently transmit XCP data from multiple ECUs with a limited number of available XCP slots, it is necessary to dynamically allocate the available bandwidth at runtime to all participating ECUs. However, AUTOSAR does not allow reconfiguration of the “FlexRay driver” at runtime. Therefore, in the integration phase the “FlexRay drivers” are configured so that all XCP slots are allocated to all of the ECUs. At the same time, the XCP-PDU/L-PDU/XCP slot allocation is set in each slave (**Figure 3**).

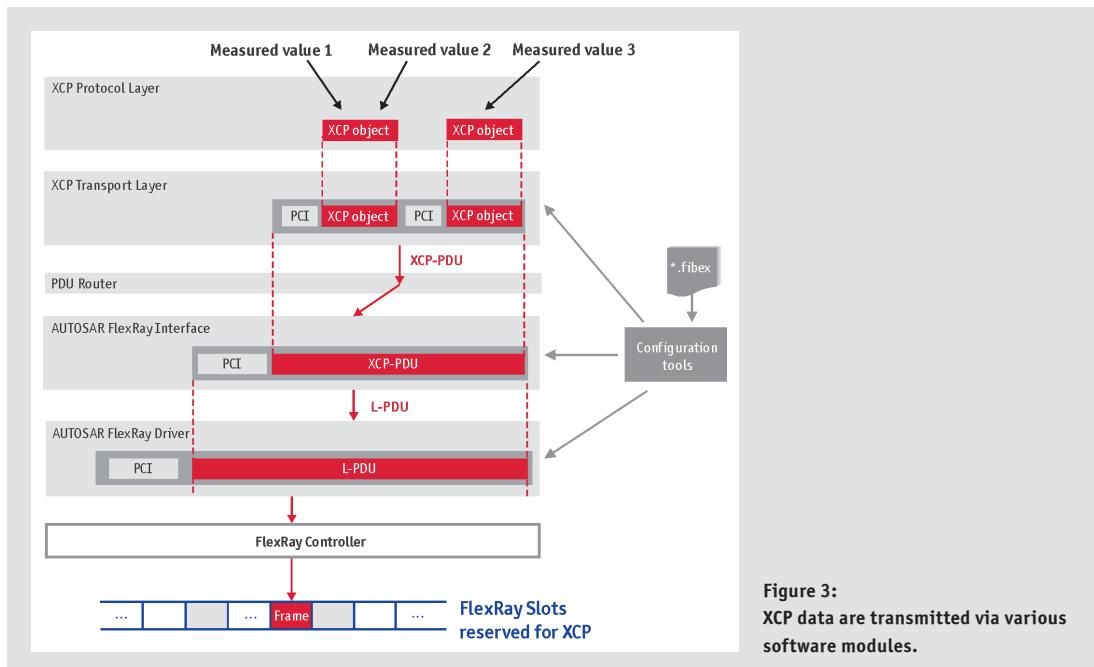


Figure 3:
XCP data are transmitted via various software modules.

As a result, at the end of the configuration process of an ECU, a unique XCP slot in the FlexRay schedule is available for each individual XCP buffer. To attain the necessary flexibility over all ECUs, the XCP transport layer command “FLX_ASSIGN” is used to modify – on the XCP level – the allocation of XCP buffers to the different L-PDUs (or FlexRay slots) before each measurement (**Figure 4**). What is important here is to configure all participating ECUs with the maximum available XCP slots, during software integration. This results in the identical allocation of XCP slots on each ECU. Before each measurement, only those XCP buffers that are actually needed are selected. A central “dynamic bandwidth management” ensures unique ECU slot allocations for XCP among all slaves. CANape handles this task with the help of the ECU-specific A2L description files that provide information about the internal ECU buffers.

XCP use is optimized thanks to FlexRay

The new dynamic bandwidth management is only one of the FlexRay-specific CANape functions that support efficient ECU calibration at Audi. In three additional functions, Vector specifically exploits the fact that FlexRay, with its up to 254 bytes of data

payload, offers XCP frames that are many times longer than those of CAN (8 bytes). The “Short Download” function simultaneously encodes the address and contents in a single L-PDU, so that memory areas can be exchanged between the master and slave much more quickly than with CAN.

Furthermore, XCP enables oversampling, which is independent of the FlexRay cycle, in order to measure very dynamic signals (**Figure 5**). CANape uses the so-called “In cycle multiple DAQ list transmission” to acquire measured signals of a predefined DAQ list and their time stamps multiple times per FlexRay base cycle (generally 5 ms long).

To significantly accelerate the start procedure before each measurement, the necessary initialization also had to be optimized thanks to the extension of the previous single “WRITE-DAQ” command. The new command “WRITE-DAQ-MULTIPLE” from the not yet released XCP Protocol Layer 1.1 has been already used to configure multiple signals by a single command.

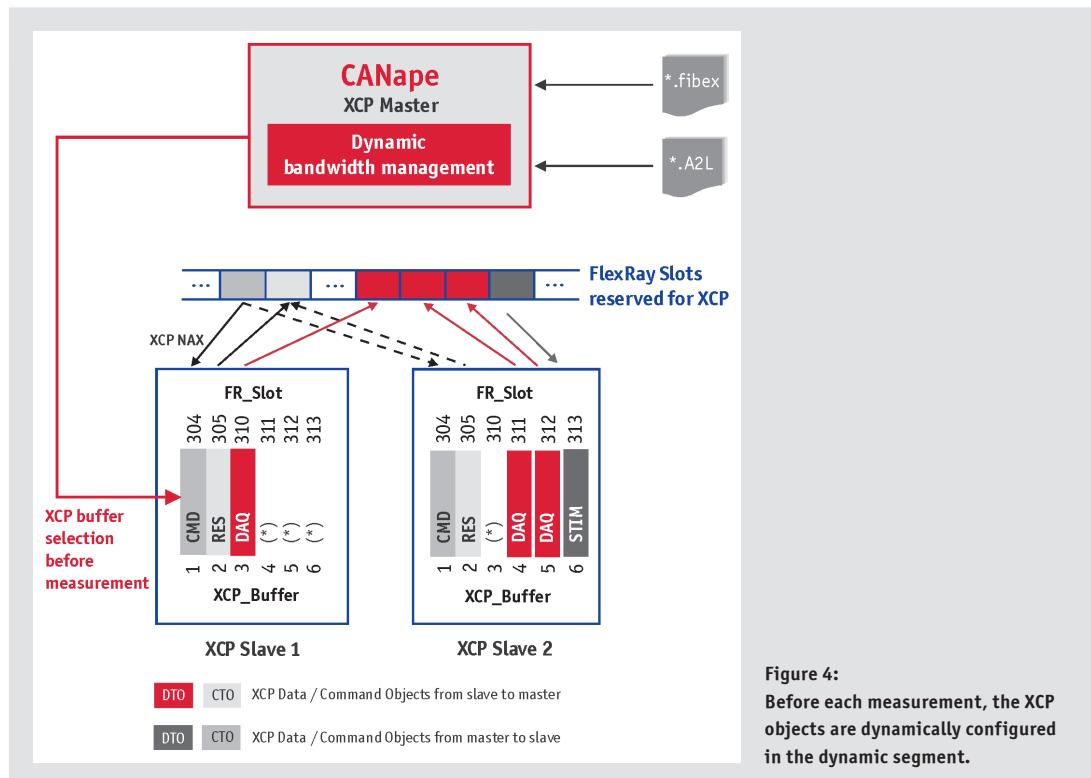


Figure 4:
Before each measurement, the XCP objects are dynamically configured in the dynamic segment.

Summary

Audi engineers rely on the MCD tool CANape to develop new models in the context of test trials and calibration drives, in order to measure and calibrate internal ECU parameters with XCP-on-FlexRay. Vector has extended its CANape and XCP software modules for this purpose. Besides extending the XCP software modules for AUTOSAR compatibility, a major task was to implement dynamic bandwidth management for FlexRay. Choosing Vector as software supplier and development partner was easy for Audi, since both the XCP software modules for the slaves and the XCP master, CANape, come from a single source – Vector – and are perfectly tuned to one another. All extensions can be obtained for the current version of CANape and the XCP-on-FlexRay software modules.

Translation of a German publication in
Hanser Automotive, 7-8/2008



Christian Braun, Audi

studied Electrical Engineering at the Technical College of Munich, specializing in Data Technology. After graduating in 1996, he joined Audi AG where he was responsible for electronic development of various ESP systems. Since 2006, he has been working in the area of Chassis Electronics Development at Audi where he is responsible for measurement tools and system networking.



Pascale Morizur, Vector

studied Physics-Electronics at the Grande École ICP in Lyon (France). After graduating in 1986, she worked for 10 years at MAN Commercial Vehicles in advanced development for the area of CAN, J1939 and Diagnostics. Since 2005, she has been employed as a product management engineer at Vector Informatik GmbH in the area of Embedded Software Components.

Literature:

[1] www.asam.net

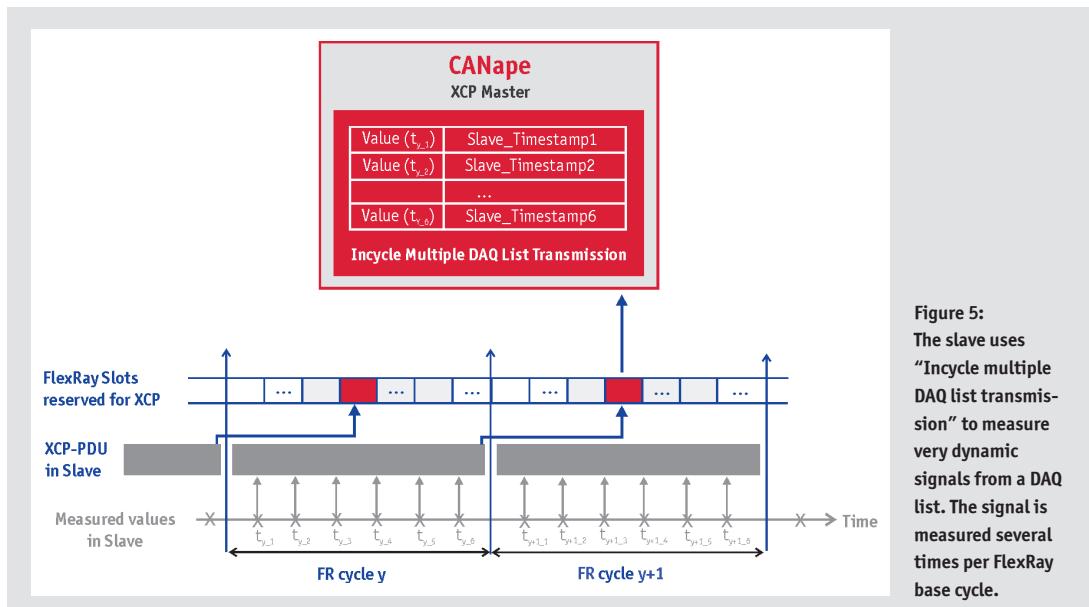
Links:

Homepage Audi AG: www.audi.com

Homepage Vector: www.vector.com

Product Information CANape: www.vector.com/canape

Product Information XCP on FlexRay: www.vector.com/canape_flexray



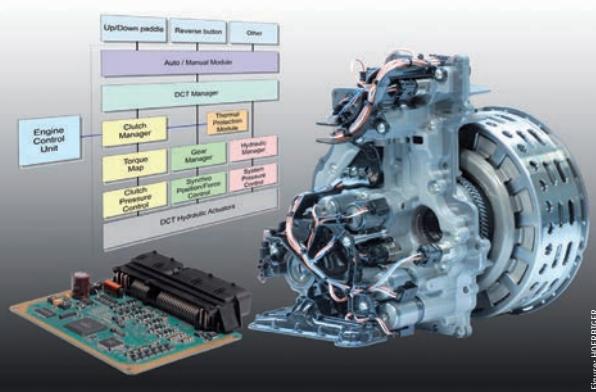


Figure: HOERBIGER

Case Study

Model-based development and ECU calibration with XCP and CANape



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

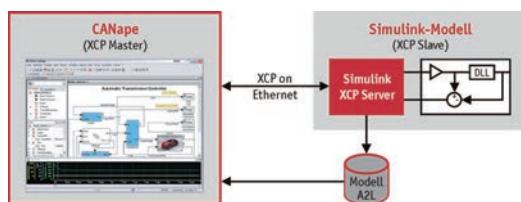
The simplest way to interface CANape with the model in Simulink is with the Simulink XCP Server. Users have the same options available here as in connecting to an ECU: drag & drop selection of measurement and calibration parameters from the description file and visualization in display and calibration 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.



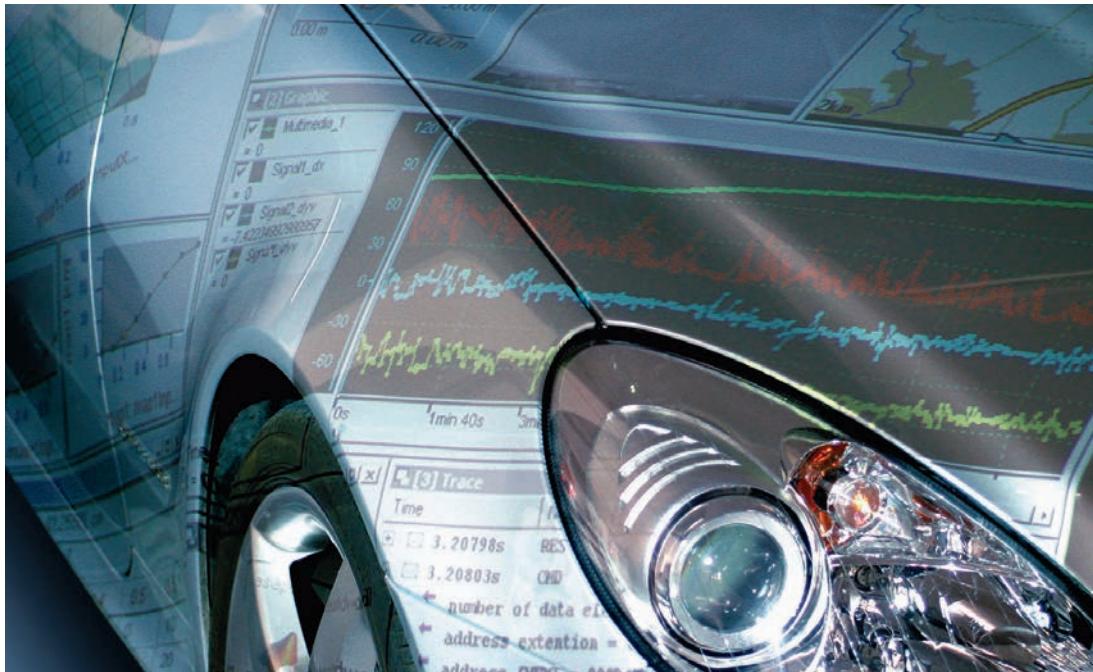
Your contact at Vector:

Andreas Patzer

andreas.patzer@vector.com

www.vector.com

XCP at the Focal Point of Measurement and Calibration Applications



More and more electronic functions for safety and convenience are finding their way into the modern automobile. Since the number of ECUs is being held in check, however, this means that the complexity of individual devices must grow to compensate. Making an important contribution toward rationalization of the development process for these distributed systems is the XCP communication protocol, whose main tasks include measurement and calibration of ECU-internal variables at runtime. A tremendous advantage of this successor protocol to CCP is its independence of the physical transport layer.

Today, control modules with more than 10,000 parameters are no longer uncommon. Since numerous dynamic processes need to be controlled in a vehicle, the typical tasks of ECU calibration include optimization of control algorithms.

In the case of a PID controller, there are almost limitless possible variations in calibrating the proportional, integral and derivative components. Therefore, it usually takes some effort until a sufficiently good compromise is found between stability, speed and transient behavior. This can be accomplished by reading and modifying parameters during runtime (**Figure 1**).

To keep the time and cost requirements for ECU calibration low, engineers and technicians depend on powerful tools and standards that enable flexible read and write access to variables or memory contents. For this purpose, the CAN Calibration Protocol (CCP) was created in the 1990s, a time in which CAN was the only dominant networking system in the automobile. CCP was slated to become a

cross-OEM standard. However, additional bus systems such as FlexRay, LIN, MOST, etc. came into play with the continued development of automotive electronics. Since CCP's use was restricted to CAN-networked applications, it increasingly encountered limitations in terms of its potential areas of use. This led to the development of its successor, XCP.

Universal standardized protocol

The “Universal Measurement and Calibration Protocol” (XCP), like CCP, also has its origins in the Association for Standardization of Automation and Measuring Systems (ASAM) [1] and was standardized in the year 2003. The “X” stands for the variable and interchangeable transport layer. XCP separates the protocol and transport layers completely by means of a two-layer protocol, and it takes a Single-Master/Multi-Slave approach. Depending on the

transport layer in question, the protocol might be referred to as XCP-on-CAN, XCP-on-Ethernet, XCP-on-UART/SPI or XCP-on-LIN, for example (**Figure 2**).

A XCP Master is capable of communicating with different XCP Slaves simultaneously. These include:

- > ECUs or ECU prototypes
- > Measurement and calibration hardware such as debugging interfaces or memory emulators
- > Rapid prototyping hardware
- > HIL/SIL systems

To meet the challenge of serving as a universal communication solution for a large variety of applications, the ASAM Working Group emphasized the following criteria in the design of XCP: Minimal resource usage in the ECU for RAM, ROM and required runtime, efficient communication, easy implementation of the XCP Slave, plug-and-play capability with low configuration effort, small number of parameters and scalability.

Interchangeable transport layer

XCP is capable of utilizing the same protocol layer on different transport layers. This protocol is a universal measurement and calibration protocol that operates independent of the network type used. Currently, ASAM has defined these transport layers in standards: XCP-on-CAN, XCP-on-SxI (SPI, SCI), XCP-on-Ethernet (TCP/IP and UDP/IP), XCP-on-USB and XCP-on-FlexRay. The last named variant is the latest member of the protocol line-up, and it has been specified since early 2006. One special technical feature of XCP-on-FlexRay is its dynamic bandwidth control. A measurement, calibration and diagnostic tool (MCD tool) such as CANape [2]

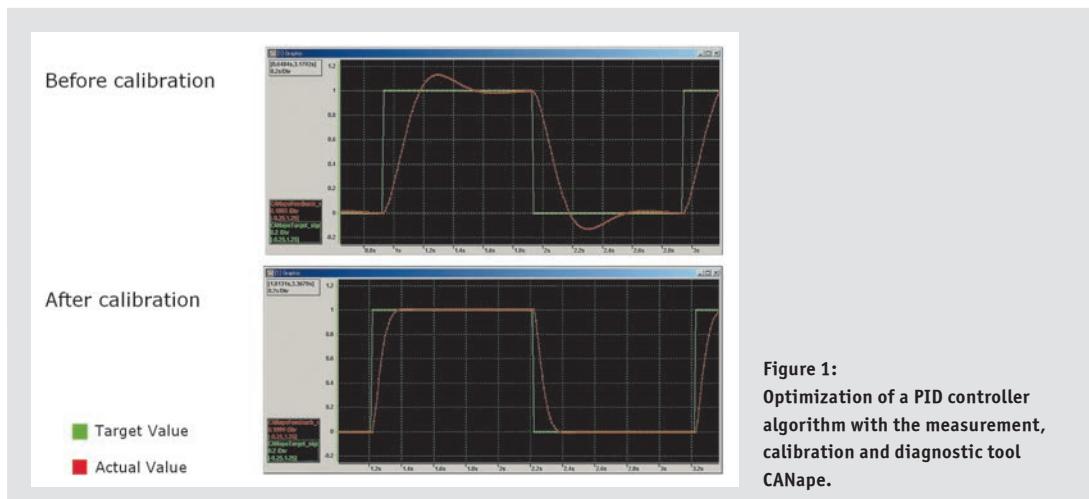
identifies the amount of bandwidth still available and dynamically allocates it to the current application data traffic very efficiently. The available bandwidth is thereby optimally exploited in XCP communication and hardly affects normal FlexRay communication at all.

Other options being considered for the future include XCP-on-LIN; also possible would be XCP-on-K-Line or XCP-on-MOST, if there is sufficient demand for them from user groups. The broad range of supported transport layers enables simple migration from broadband solutions such as Ethernet or USB in the development phase to a final CAN interface in series production.

Single-Master/Multi-Slave concept

The measurement and calibration system assumes the role of the XCP Master, while the ECU acts as a Slave. Master and Slave each communicate via the XCP drivers integrated in them. For each Slave there is an ECU description file; information specified in this file includes: Allocations between (symbolic) variable names and their address ranges, physical meanings of the data, and the checksum method being used. The XCP Master can read out all of the information it needs from these A2L description files.

In communication via XCP a distinction is made between the "Command Transfer Object" (CTO) and the "Data Transfer Object" (DTO). The Master might send a command via a Command Transfer Object over the bus to the ECU that acknowledges it over the same path after executing the requested service. CTOs provided are: CMD (Command), RES (Response), ERR (Error), EV (Event) and SERV (Service Request Processor). The DAQ (Data Acquisition) and STIM (Stimulation) Data Transfer Objects are used for event-driven reading of measurement variables from memory, or for writing of values to the memory of the XCP Slave (**Figure 3**).



From automotive bus to standard PC interface

PC platforms are used almost exclusively as the Master for measurement and calibration tasks. For direct connections to automotive bus systems such as CAN, LIN, FlexRay, MOST or K-Line, the PC is generally equipped with one or more hardware interfaces. Furthermore, the XCP Master is capable of utilizing standard PC interfaces such as Ethernet, USB and RS232. Of course, in such solutions no other costs are incurred for interface hardware. Measurement and calibration systems with debugging interfaces (JTAG, TRACE, etc.) and memory emulators can be implemented in this way. In principle, the standard PC interfaces are also well-suited for connecting gateways between different bus systems; FlexRay-on-Ethernet could handle this, for example. And finally, there are the conventional analog and digital I/O channels that are utilized in many development and test layouts involving especially time-critical measurements.

A significant advantage in the use of XCP lies in the fact that a single standardized protocol suffices for all of these applications. Without XCP it would be necessary to implement a proprietary driver for each communication channel. However, performance losses need to be taken into account when multiple drivers are used in parallel. Moreover, the risk of undesirable interactions and instabilities increases.

Versatile, scalable and resource-economizing

One and the same XCP driver code is used for all communication processes. It can serve to send just a few bytes that come from small controllers and interfaces, e.g. 8-Bit processors with integrated serial interface. Or the same code might be used to send

megabyte-size data volumes with 32-bit processors over high-speed networks such as Ethernet. Since an XCP driver is made up of mandatory and optional functions, driver size can be scaled to match the available ROM/Flash size. In the ECU, resource usage is characterized by whether the focus is on high data throughput or on low processor load and RAM size. With regard to bus load, the basic consideration is: Number of signal transmissions vs. bus bandwidth. Overall, the XCP drivers are easy to implement and require just a few parameters.

Event-driven periodic data acquisition

ECUs operate in discrete time intervals. The length of such a time interval can be defined in a fixed way (e.g. 10 milliseconds) or the length of a time interval may be event-dependent (e.g. one engine revolution). In the case of fixed defined time intervals, the end of the time slice is marked by the expiration of a timer. Expressed in generalized terms, such expiration of a timer is also an event. The task of the ECU is to complete all computational and control tasks within the specific time slice. To obtain correlated data from the XCP Slave, the DAQ mechanism of the XCP protocol is now used. In this mechanism, the XCP-Master informs the Slave, before the beginning of the measurement, which signals are to be measured when certain events elapse. If the event now occurs (e.g. the 10 millisecond timer expires), the XCP Slave reads the previously defined data from the memory, copies them to a protected RAM area and sends them to the Master in the form of messages. This assures that the values originate from the same event cycle and correlate.

The Master receives the data provided with a time stamp and saves them to a measurement file. The time stamp is either sent out by the

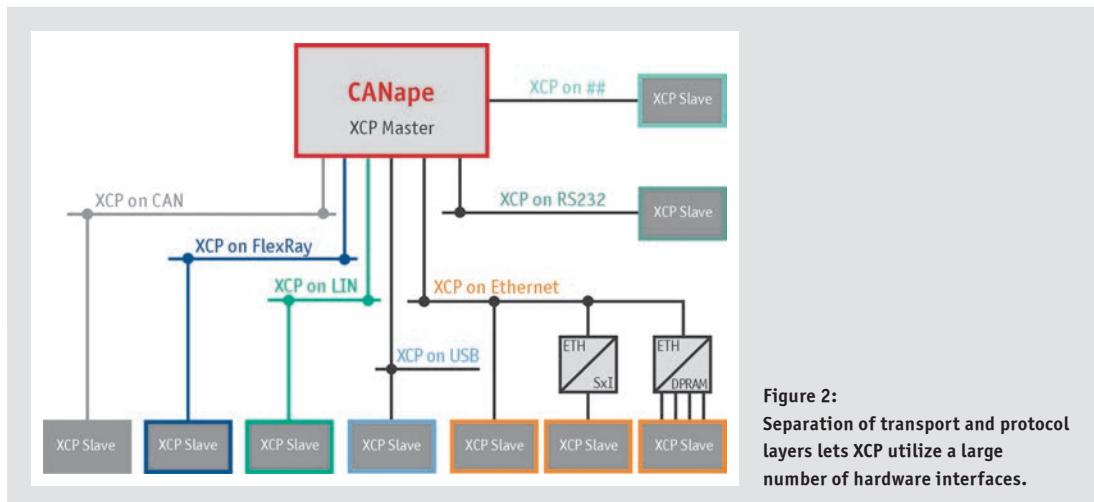


Figure 2:
Separation of transport and protocol layers lets XCP utilize a large number of hardware interfaces.

XCP Slave as a datum, or it is assigned to messages by the interface hardware being used, e.g. CANcardXL. In the measurement file, all data are synchronized in reference to the Master time base and are then further processed, e.g. by visualizing the measurement values on a global time axis. This allows multiple data channels of different XCP Slaves to be displayed consistently in one diagram.

Besides the advantages of XCP compared to CCP already been mentioned, XCP also offers support of so-called cold-start measurements and internal ECU time stamps for tasks in cyclic data acquisition. In cold-start measurement the ECU can be configured so that it begins to periodically send out data immediately after being activated, i.e. the Master does not need to initiate this explicitly. If an internal ECU time stamp is being used, it is not the time of receipt that is relevant for later evaluation in the measurement and calibration system, rather it is the time at which the data were created in the XCP Slave. This eliminates uncertainties attributable to possible delays in the transmission process, e.g. under conditions of insufficient bus bandwidth or high load.

Optimizing characteristic curves and maps

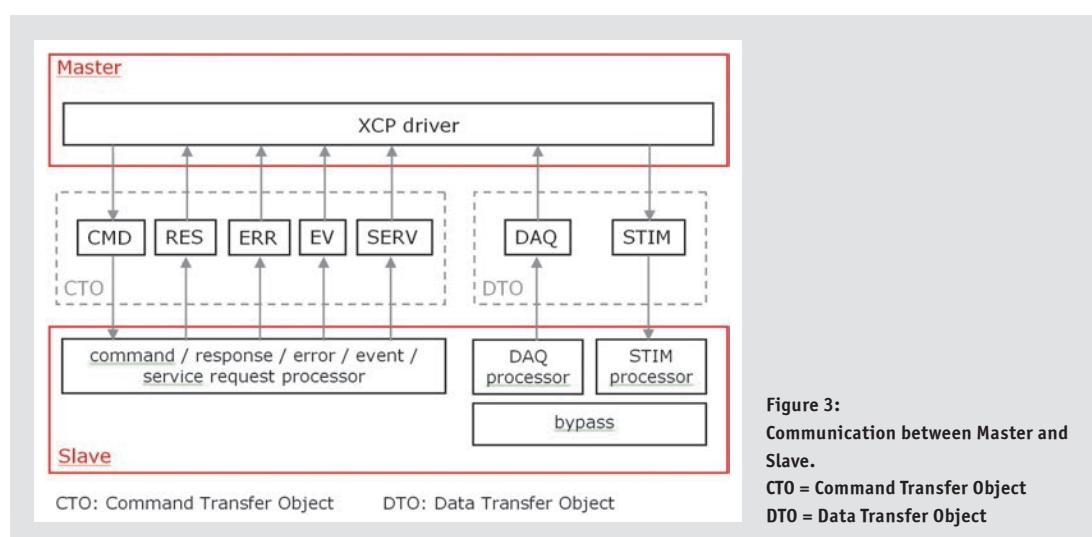
Besides control algorithms based on mathematical models, ECUs also utilize characteristic curves and maps composed of discrete interpolation points. To achieve desired system behavior, such characteristic value tables are usually set up and optimized experimentally, e.g. at the test bench. A2L files serve to describe the measurement variables and calibration parameters. The description options in the A2L files range from simple scalar parameters to complex tables. Among other things, the descriptions encompass the data types, conversion rules between raw and physical values,

storage schemes for characteristic maps, and much more. High-performance calibration tools such as CANape from the Vector display the characteristic curves and maps clearly on the screen in graphic diagrams or as numeric values in tables.

Rapid prototyping with CANape and XCP

During ECU development important functions are frequently swapped out to external simulation systems, so that they can be computed with less effort. It is not until the algorithms in the simulation model have reached a certain maturity level that the developer generates from them a source code that is compiled together with other ECU-relevant codes and flashed in the ECU. However, even before this occurs so-called "bypassing", a coupling between a real ECU and its model, is desirable so that tests and optimizations can be made independent of the hardware at an early point in development.

In bypassing with XCP, the Master reads data from the ECU by DAQ, passes the values to the model as input values and sends the model's results back to the ECU by STIM. Noteworthy in this context is that normal PC platforms running the MCD tool CANape are sufficient for bypassing and modeling. This is good news, since solutions based on special real-time hardware would be many times more expensive and would only be available in limited numbers in development departments. CANape, as a highly optimized XCP Master, simultaneously handles communication with the real ECU and with the model running on the PC (**Figure 4**). The ECU parameters and model parameters can both be calibrated via CANape and XCP.



Flashing via XCP

XCP also offers benefits to the user in programming ECUs. The data in the ECU's flash memory are only reprogrammable using specially prepared flash routines, which must also reside in the ECU. In principle, two approaches are conceivable: In solution 1, the flash routines are stored permanently in flash; first, this wastes memory, and second, it presents a security issue for delivered vehicles. In solution 2, a PC tool only loads a flash kernel to the microcontroller's RAM via XCP if it is needed for reprogramming. Besides containing the flash routines for erasing the flash memory and rewriting data, the flash kernel also contains its own bus and SCP drivers for communicating with the PC tool over the bus interface.

Summary

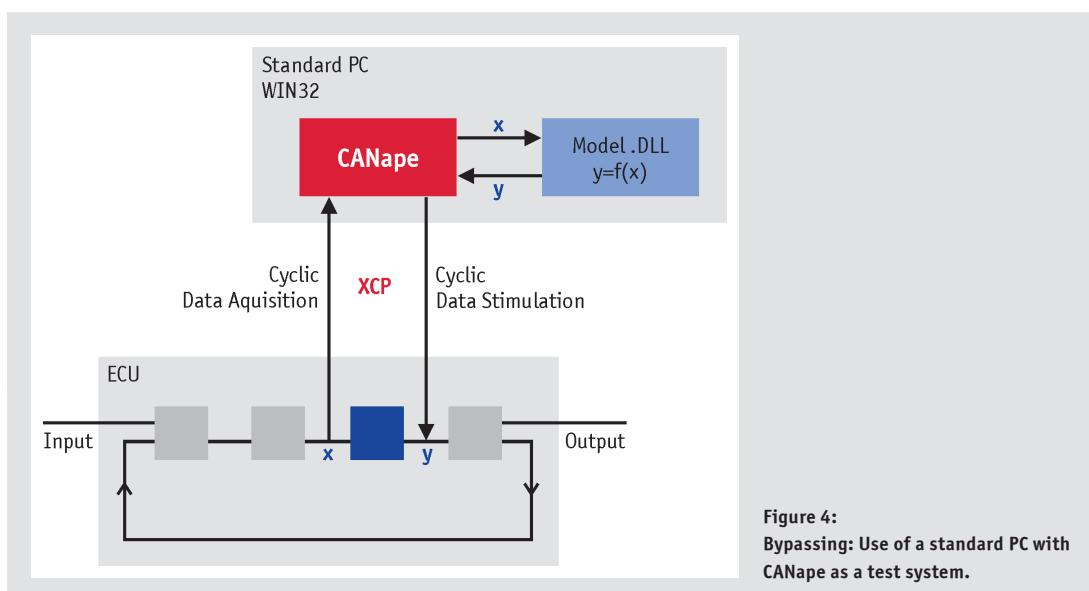
XCP is a standardized and universally applicable protocol with much rationalization potential. It is not only used in ECU development, calibration and programming; it is also used to integrate any desired measurement equipment for prototype development, functional development with bypassing and at SIL and HIL test stands. For quick access to internal data via microcontroller debugging interfaces such as NEXUS, communication occurs over dedicated hardware, trouble-free. This hardware converts communication from NEXUS to XCP-on-Ethernet. The benefits to the user are independence from tool producers with proprietary solutions, and reusability of components.

Vector provides a free driver for setting up a XCP Slave that can be downloaded at its web site [3]. The MCD tool CANape, in use as an ECU calibration tool since 1996, is continuously updated as an XCP Master to the latest level of XCP standardization based on Vector's active participation in ASAM working committees. CANape is the first tool on the market to have an XCP-on-FlexRay interface. In the development of the first production vehicle with FlexRay, the BMW X5, this was a crucial factor in the decision by BMW engineers to rely on Vector's XCP stack and CANape for calibrating the damper control system.

Translation of a German publication in
Elektronik automotive, 4/2007

Literature and Links:

- [1] www.asam.net
- [2] www.vector.com/canape
- [3] www.vector.com/downloads/drivers/xcp.exe





Andreas Patzer

After a practical education to become an IT technician, Andreas Patzer studied Electrical Engineering at the Technical University of Karlsruhe between 1986 and 1993. He specialized in measurement and control engineering as well as information and automation engineering. After receiving his degree, Mr. Patzer worked in the communications industry. In 2003 he joined Vector Informatik GmbH in Stuttgart where he handles the interface between customers, development and sales as Business Development Manager for the "Measurement and Calibration" product line.

Quantum Leap in Microcontroller Measurement Technology

Innovative ECU measurement concept for maximum data rates with minimal effects on execution time



In the development of complex ECU applications, there are greater and greater quantities of data to be processed, more signals to be measured, and a growing number of parameters to be optimized. Previous methods for measuring, calibrating and flashing are increasingly encountering limits with regard to the necessary data throughput. It was in this context that Robert Bosch GmbH initiated a search for a more powerful and future-robust measurement concept for the next generation of its ECUs, in particular for the development of a new long-range radar sensor.

The long-range radar sensor LRR3 (Long-Range Radar) from Bosch operating at 77 GHz is the “sensory input” for many safety-related and driver assistance systems. They include various versions of Predictive Safety Systems (PSS) and Adaptive Cruise Control (ACC). These smallest radar sensors in automotive industry, used in production vehicles since the beginning of 2009, are distinguished by their long acquisition range of 250 m and their wide aperture angle of up to 45°. Together with a favorable price the application area is broadened from luxury class to mid-class vehicles and commercial vehicles. This development posed enormous challenges for Bosch engineers when it came to performing measurement and calibration tasks. Along with options for measuring and logging data, it was essential to have efficient methods for calibrating and flashing as well as bypassing. All of these applications require extremely high transmission rates with low latency times.

From a technical perspective, it made sense to implement a modular layout of the measurement system and to make use of a standardized PC interface. Development of a near-production ECU enabled a simple transition from development to production later on. To acquire the large number of measurement signals, up to 100,000, without errors, a data rate of at least 4 MB/s is necessary while affecting the processor as little as possible.

Existing solutions: Low data rates and high CPU load

In solutions that utilize the standardized measurement and calibration protocols [1] CCP or XCP-on-CAN, FlexRay, JTAG or SPI, a driver integrated in the ECU is responsible for periodically reading, copying and sending out the desired signal values. Due to the large volume of data to be measured, the driver requires RAM memory

and processor resources that have limited availability. In addition, there is increased loading of the data bus, which impacts the ECU software in a negative way. Measurement data rates range from 50 KB/s for CAN up to maximum values of 400 KB/s for FlexRay, JTAG and SPI (**Table 1**).

A high-performance debug interface on the micro-controller opens up new possibilities.

Bosch decided to join together with specialists at Vector to conceptualize an entirely new measurement and calibration system. As a measurement interface it utilizes the Data Trace Interface, which an increasing number of modern microprocessors are equipped with for debugging purposes. More specifically, this is a standardized Nexus Class 3 Interface, which can communicate every change in the ECU's memory to the outside world with minimal processor load.

The fundamental idea of this approach is to acquire data from the ECU via the debug interface and route it to an external measurement adapter via a special high-speed cable. The data are transmitted serially by a dedicated protocol. The external measurement adapter is able to transmit the actual measurement data to the application on the PC – independent of the ECU – via the standardized XCP-on-Ethernet protocol.

In the project, the interface on the ECU was implemented by a Plug-on-Device (POD). This POD is especially compact, and it is easy to mount it on the ECU. The POD contains all of the electronics needed to acquire and send out measurement data. To assure error-free operation, the POD fulfills the same mechanical and electronic environmental requirements as the relevant ECU. This allows the POD to be installed in critical locations in the engine compartment, for example, which was a key requirement in the development project at Bosch.

Measurement adapter with mirror memory

A HSSL (High Speed Serial Link) cable up to 5 m in length connects the POD to the VX1100 base module (measurement adapter) of the

modular VX1000 system developed by Vector (**Figure 1**). The base module primarily consists of a FIFO buffer, Dual-Port RAM (DPRAM) and XCP Engine that also has dedicated RAM. Write accesses to data within the two user-definable memory areas are transferred to the FIFO buffer in the base module via the HSSL connection and the debug interface. The data are further processed and written to the DPRAM there. From a logical perspective, since this data is identical to the data stored in the ECU, the DPRAM always contains a current mapping of the data, mirroring memory areas in the ECU. The crucial aspect of this approach is that all measurement processes occur via the mirror memory. To initiate a measurement and thereby initiate data transfer, it is sufficient to have the ECU write an event number to a predefined memory address that is allocated to the measurement data. At precisely this time point, the connection between the FIFO and DPRAM is disconnected, “freezing” the memory map at the trigger time. This keeps the data to be measured constant over the measurement period. The XCP Engine now processes the data according to the protocol.

A transmission rate of up to 5 MB/s was achieved for the XCP-on-Ethernet connection between the VX1100 measurement adapter



Figure 1:
The VX1000 system is distinguished by very high measurement data rates and very low software modification requirements and effects on ECU execution time.

ECU interface	ECU software-modification	ECU RAM-reqmt.	Measurement data rate	ECU execution time effects	Bypass latency time
CCP/XCP on CAN	CCP/XCP driver software	1–2 KB	50 KB/s	Moderate	High
XCP on FlexRay	XCP driver software	2–16 KB	50–400 KB/s	Large	Moderate
XCP on JTAG/SPI	Tables for DAQ transfer via software	4–16 KB	200–400 KB/s	Large	Moderate
Data trace VX1000	Low	None	5,000 KB/s	Very low	Low

Table 1: Comparison of different methods of measurement data acquisition

and the measurement and calibration tool on the PC. A highly robust, temperature-resistant HSSL cable was used to ensure absolutely error-free data transmission in the engine compartment. In case of transmission errors, a retransmission protocol provides for quick repetition of data packets.

A look at the system's performance demonstrates that the effort was very worthwhile. The VX1000 measurement system impresses with a measurement data rate of up to 5 MB/s, it enables a write rate of about 1 MB/s and handles the 100,000 signals of the Bosch application effortlessly. The precision of the time stamps is 1 microsecond, while bypass turnaround times of 300 microseconds were attained.

From laboratory simulations to rapid prototyping

These properties make the system ideally suited for the two primary applications at Bosch. The first application is bit-precise simulation of real scenarios in the laboratory. This involved feeding certain scenarios into the simulation without requiring real vehicle drives. The second application, bypassing, is used to execute and test functions outside of the ECU.

The described measurement system fulfills all requirements necessary for the LRR development, and it is now being used in other projects at the Bosch company. Compared to classic measurement principles, the VX1000 solution offers performance increases by a factor of 10 to 100 in all disciplines. The effect of measurements on the ECU, with CPU loading of less than 1%, lies significantly below usual values. The modular layout of the VX1000 system assures cost-effective re-use as a measurement technology solution in subsequent projects, even with different microcontrollers.

The right solution for any required measurement data rate

The VX1000 system completes Vector's product line of measurement and calibration tools at the top end of performance. Because it could reach previously unattainable measurement data rates, it

fulfilled all expectations set in the Bosch project. Last but not least, along with good cooperation between the two companies, the CANape tool for measurement, calibration and diagnostics made an important contribution to successful project completion. CANape is primarily used to optimally parameterize ECUs. In the development and optimization of driver assistance systems like ACC, the CANape Option Advanced Multimedia offers specialized capabilities. Among other things, it lets users display objects detected by the system in a perspective view in time-synchronously logged video images, which gives them a reliable means for verifying object detection algorithms.

Other application options and outlook

The standardized XCP-on-Ethernet protocol can be used with both the VX1000 product line and other measurement and calibration tools. In the case of measurement and calibration tasks in the engine compartment, the VX1000 is not really an off-the-shelf product, since the harsh environmental conditions and limited installation space generally require custom modification of the ECU connection. In the framework of project work, Vector can work out individual solutions in close dialog with its customers. Devices currently supported, besides the Freescale PowerPC, are the TMS570 from Texas Instruments and the Infineon TriCore processors with DAP interface which are widely used in engine controllers (**Figure 2**). DAP enables a cost-effective solution via a plug connector on the mini-PC-board connected to the ECU. Cycle times of 50 microseconds are possible with the measurement and calibration system. These requirements are relevant in the development of vehicles with electric/hybrid drives, for example.

Based on acceptance of the VX1000 system among automotive OEMs and suppliers, all sorts of extensions and new features will be offered in the near future. They include plans to support additional processors. Well-known semi-conductor manufacturers have approached Vector seeking recommendations on how to adapt their processor architectures in the direction of optimal measurement functionality.

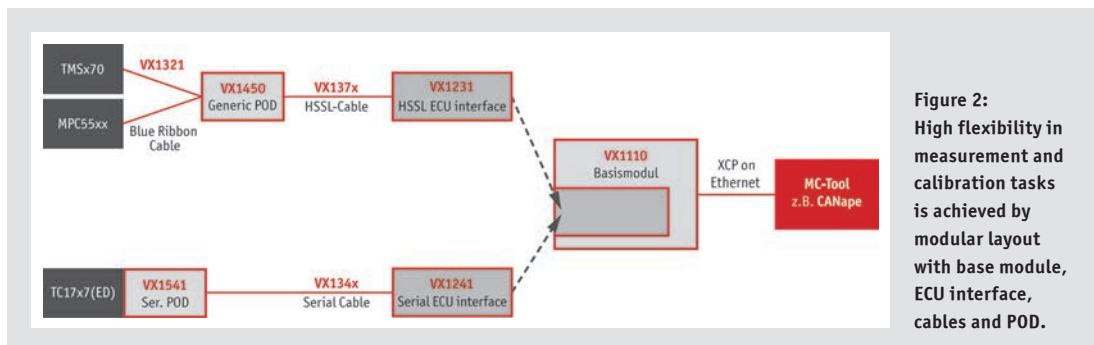


Figure 2:
High flexibility in
measurement and
calibration tasks
is achieved by
modular layout
with base module,
ECU interface,
cables and POD.

Translation of a German publication in
Hanser Automotive, 9/2009

Literature references:

- [1] XCP protocol: www.asam.net
- [2] Presentations by Riedl, A. and Kless, A. at the Vector Congress 2008.
Download at www.vector.com/veco08



Andreas Riedl, Bosch

is Project Leader for engine controllers in the international field at Robert Bosch GmbH.



Alfred Kless, Vector

a Business Development Manager at Vector Informatik GmbH since 2004, is responsible for the "Network Interfaces" and "Measurement and Calibration" product lines.

High-speed Measurements for Electric and Hybrid Vehicles

New measurement concepts enable high data rates and frequent sampling times



In the development of electric and hybrid vehicles, in particular, the requirements for instrumentation used to measure internal ECU signals are very high. Nonetheless, measurement data rates of up to 30 Mbyte/s and the necessary sampling rates of 100 kHz can be achieved with the latest generations of microcontrollers and an intelligent measuring instrumentation solution. The ECU's CPU is not loaded here.

The drives of electric or hybrid vehicles are generally controlled by pulse-width modulation (PWM) signals. The advantage of PWM technology is that it incurs very low power losses at power switches, because they only need to be operated in two operating states: fully conducting or fully blocking. The frequency of the PWM signals typically lies in the 10 – 20 kHz range, and in exceptional cases up to 100 kHz. Maximum sampling rates of only 1 kHz are achievable for internal ECU signals when XCP – a widely used standardized measurement and calibration protocol for vehicle development – is used together with communication over the CAN or FlexRay bus system. PWM signals cannot be acquired in this method.

That is why the debug and data trace interfaces are used for fast access to ECU variables. These interfaces can vary significantly depending on the type of microcontroller that is implemented. The measurement hardware is interfaced to the ECU over a "Plug-On Device" (POD). The maximum allowable distance between the microcontroller's debug pins and the POD is 10 cm. Communication between the measuring instrumentation module and the test PC is

over XCP on Ethernet in accordance with the MCD-1 XCP standard from ASAM. The physical connection is made by a standard CAT-5 Ethernet cable. Essentially, two different measurement methods are distinguished: the "RAM copy method" and the "data trace method." They are presented in this article, together with their advantages and disadvantages, based on current microcontrollers and new microcontrollers that will be available soon. The different data trace methods refer to two types of 32-bit microcontrollers that are primarily used in powertrain ECUs and their successors: Freescale PowerPC (primary market: USA) and Infineon TriCore (primary market: Europe).

RAM copy method

The RAM copy method is a generic method, and can be used for current and future generations of 32-bit microcontrollers from various manufacturers. For the Infineon TriCore or XC2000, access is via the Device Access Port (DAP) interface; for the PowerPC devices

from Freescale or V850 E2 processors from Renesas, access is via the Nexus Class 2+ interface. In this method, the ECU software initiates a RAM copy function according to the cycle time of the various ECU tasks. The measurement signals must be preconfigured over XCP on Ethernet. The mapping of signal names and RAM addresses is described in an A2L file (ASAM standardized ECU description file for signal-oriented RAM accesses). Once all measurement signals have been copied, the signals are transmitted to the base module for measurement data according to the existing debug interfaces (**Figure 1**). This concept is referred to as "Online Data Acquisition" (ODA).

Compared to CAN, the measurement data rate and sampling rate are improved by a factor of 20, i.e. 0.5 to 1 Mbyte/s of measurement data can be acquired with a sampling rate of 10 – 20 kHz. The copying operation loads the CPU approx. 4% at 1 Mbyte/s.

Data trace concept for Nexus Class 3 – current Freescale PowerPC

Most devices of the current Freescale PowerPC series support the data trace method of Nexus Class 3. In this case, the developer configures one or two monitoring windows with a maximum total size of 512 kByte in the ECU RAM. Any changes within these monitoring windows are transmitted to the POD via Nexus Class 3 without any additional CPU load. Transmission rates for raw data of up to 100 MByte/s are possible over the High Speed Serial Link cable. The advantage of this concept is that the base module for measurement data always contains a consistent mirrored RAM of the ECU's RAM. An ECU software trigger interrupts the data flow within the

measurement data base module, where new changes are saved in a First In, First Out (FIFO) buffer in RAM. The measurement is initiated by one of up to 256 different software triggers, and the contents of the mirrored RAM are "frozen". Based on the measurement configuration, the signals are read out from the mirrored RAM in the base module for measurement data and are sent to the measurement and calibration tool over XCP on Ethernet (**Figure 2**).

Advantages of the Nexus Class 3 solution:

- > The maximum measurement data rate of 30 Mbyte/s is a factor of 30 times larger than with Nexus Class 2+ and 600 times larger than with XCP on CAN.
- > The CPU is typically not loaded by the measurement.
- > All PWM drive signals can be measured at the 100 kHz sampling rate without any problems.

The disadvantage of this solution lies in the fact that significant effort is involved in connecting the POD with its 25 pins to the microcontroller, and it must process a very large raw data stream of 100 Mbyte/s.

Data trace concept for next generation microcontrollers

The main disadvantage of the Nexus Class 3 solution will be eliminated in next generation microcontrollers, because the pin count has been reduced from 25 to 5. However, the measurement data rate and sampling rate will remain at the same unchanged high level. This data trace solution will also be supported by future processors from the Infineon and Freescale companies. The raw data stream von 100 Mbyte/s must still be processed.

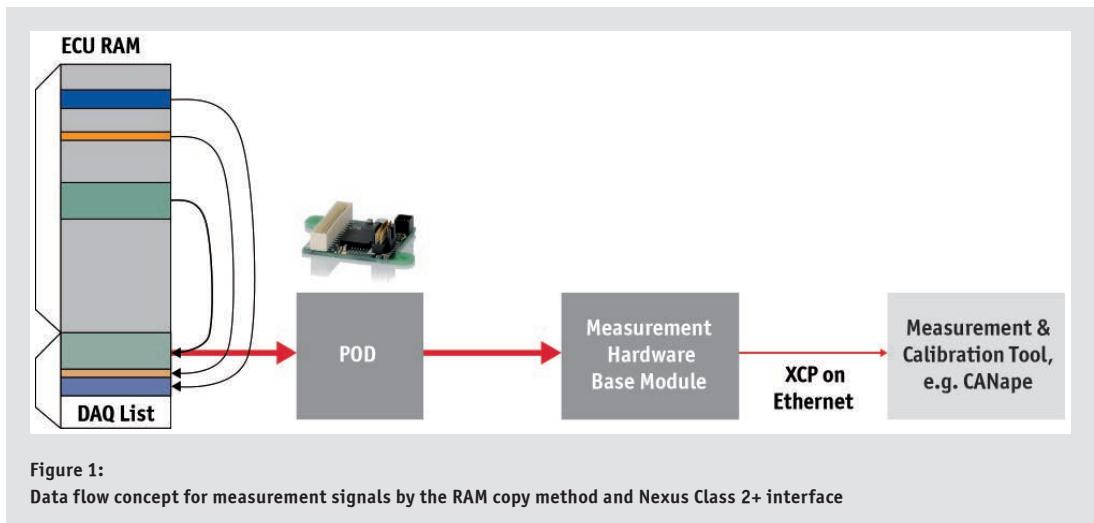


Figure 1:

Data flow concept for measurement signals by the RAM copy method and Nexus Class 2+ interface

Data trace concept for the current Infineon TriCore

A concept comparable to Nexus Class 3 may also be used for DAP. This involves reserving a 256 kByte memory range of the ED-RAM (Emulation Device RAM) for measurement data acquisition. In contrast to the 100 MByte/s of the Nexus Class 3 concept, the trace transmission rate for raw data must be limited to 5 MByte/s; just 4 pins suffice instead of 25 pins. A maximum of four RAM monitoring windows may be configured. They must be configured so that there is no overrun of the trace data. Generally, this permits monitoring of just 10–20 kByte of memory instead of 512 kByte and measurement of signals in this memory without processor loading. Signals outside of these trace monitored memory areas can be measured by the RAM copy method.

Advantages of the Infineon DAP data trace solution:

- > The maximum measurement data rate of 3 Mbyte/s is a factor of 3 larger than in the RAM copy method.
- > The microcontroller is not loaded by the measurement.
- > All known PWM drive signals can be measured at a 100 kHz sampling rate without any problems.

Data trace concept for future Infineon controllers

In the next generation of microcontrollers, Infineon is also offering the latest generation Device Access Port (DAP). One advantage lies in its higher raw data transmission rate, which is now 20 MByte/s in contrast to the previous 5 MByte/s. This is attained by the higher frequency of 160 MHz at the DAP interface instead of the

previous 80 MHz and by a new type of three-line concept, which permits parallel transmission on two lines.

The greatest improvement to the DAP2 interface is that it now lets users set up hardware-based data trace filters with extremely fine granularity. This significantly reduces the transmission of unnecessary data trace information from the microcontroller to the POD. Despite the maximum measurement data rate of 10 Mbyte/s, it is only necessary to process 15 instead of 100 Mbyte/s of raw data (**Figure 3**). Due to the considerably reduced requirements for processing the measurement data, cost-optimized measuring instrumentation can be used for DAP2.

Summary

Many aspects of modern drive concepts for vehicles with pure or hybrid electric motors make it necessary to develop new strategies for measurement data acquisition. Existing measuring instrumentation concepts for internal ECU signals often reach their limits in terms of data rate or sampling rate. The sampling rates of up to 100 kHz that are necessary for electric drive systems can be implemented for existing and future microcontrollers using the VX1000 measurement and calibration hardware from Vector. Over the course of this year, new controller generations will be available from Freescale and Infineon, which can perform their tasks with a data trace that requires significantly fewer connection pins. In combination with the high-speed VX1131 measurement module from Vector – which will be available in the second half of 2012 – they will enable measurement data rates of 30 Mbyte/s without CPU loading.

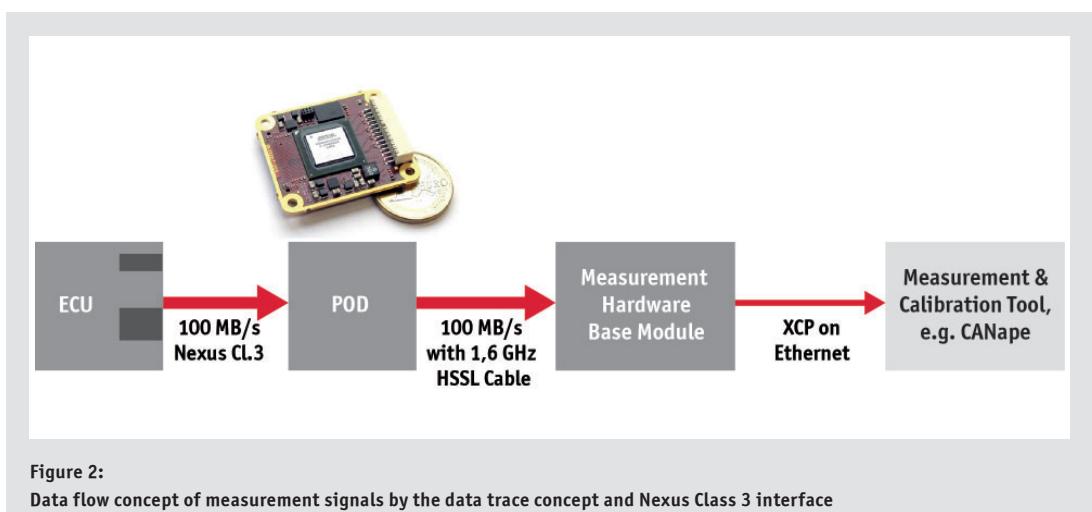


Figure 2:

Data flow concept of measurement signals by the data trace concept and Nexus Class 3 interface

In the case of Infineon, DAP2 with finely granulated signal filters in the microcontroller make it possible to reduce the raw data stream from 100 to 15 Mbyte/s, which permits the use of very cost-efficient measurement hardware to achieve high data rates. When used with the ASAM-standardized XCP on Ethernet as the PC interface, the measurement and calibration hardware is also ideal as a flexible and powerful bypass solution with short latency times.

**Translation of a German publication in
Hanser Automotive, 5-6/2012**

Links:

Vector's tools for ECU calibration:

www.vector.com/vi_calibration_solutions_en.html

Product information VX1060 Measurement and Calibration Hardware:

www.vector.com/vi_vx1060_en.html

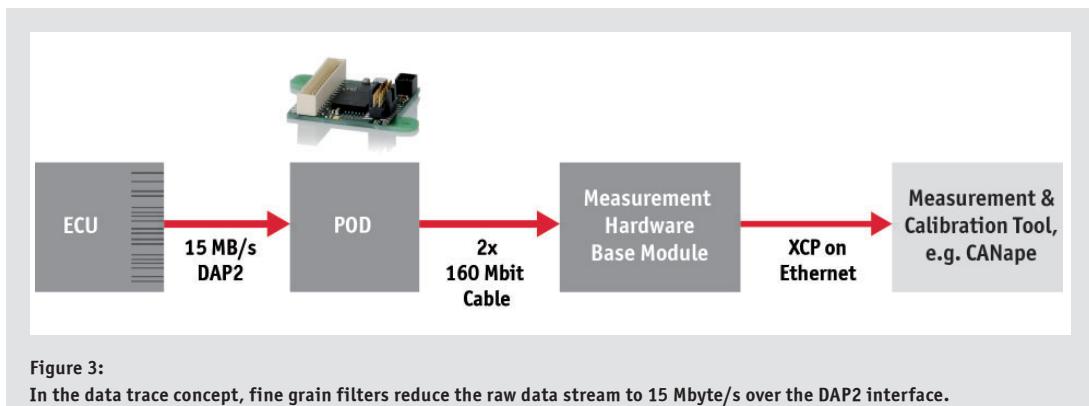
Product information CANape:

www.vector.com/vi_canape_en.html



Alfred Kless

After graduating from the Technical College of Esslingen with a degree in Electrical Engineering, Alfred Kless initially worked for ALCATEL where his roles included 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 product lines "Measurement & Calibration" and "Network Interfaces."



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 evaluations, e.g. upshifting or downshifting, and starts the analysis process (**Figure 3**). After completing the evaluation, statistics are

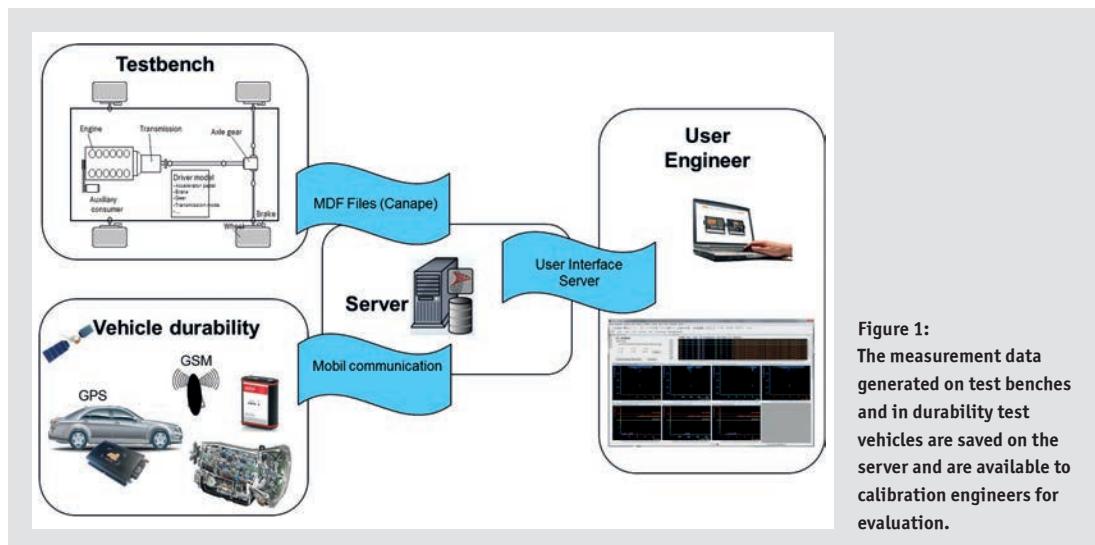


Figure 1:
The measurement data generated on test benches and in durability test vehicles are saved on the server and are available to calibration engineers for evaluation.

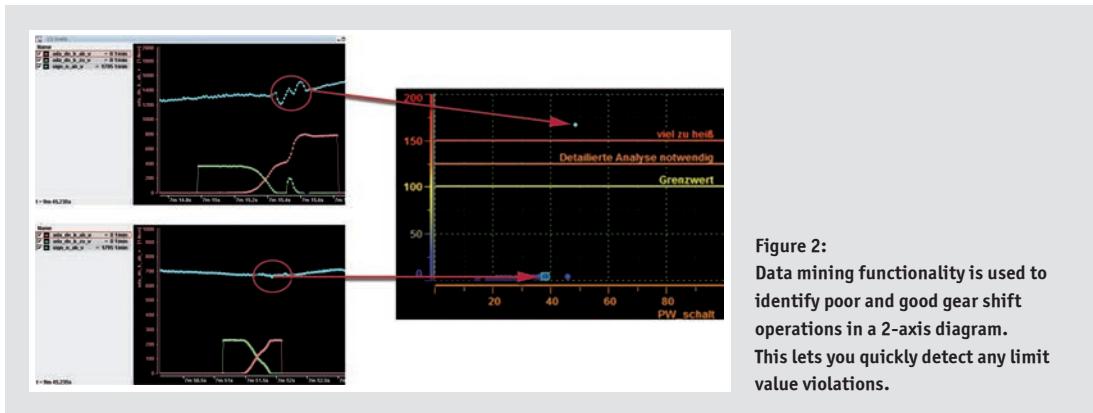


Figure 2:
Data mining functionality is used to identify poor and good gear shift operations in a 2-axis diagram. This lets you quickly detect any limit value violations.

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 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

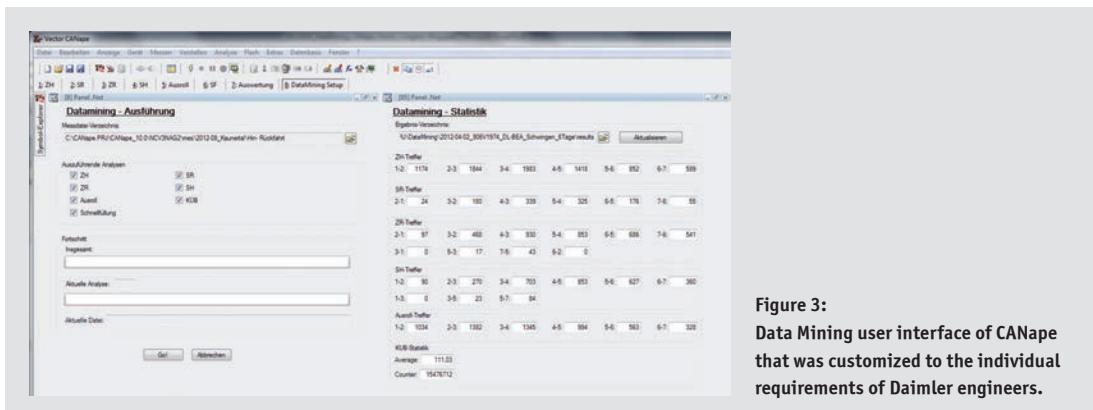


Figure 3:
Data Mining user interface of CANape that was customized to the individual requirements of Daimler engineers.

inadequate for any reason, other function libraries may be generated 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.

Translation of a German publication in Elektronik automotive, 10/2013

All figures: 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.

>> Contact information of the Vector Group:

www.vector.com/contact

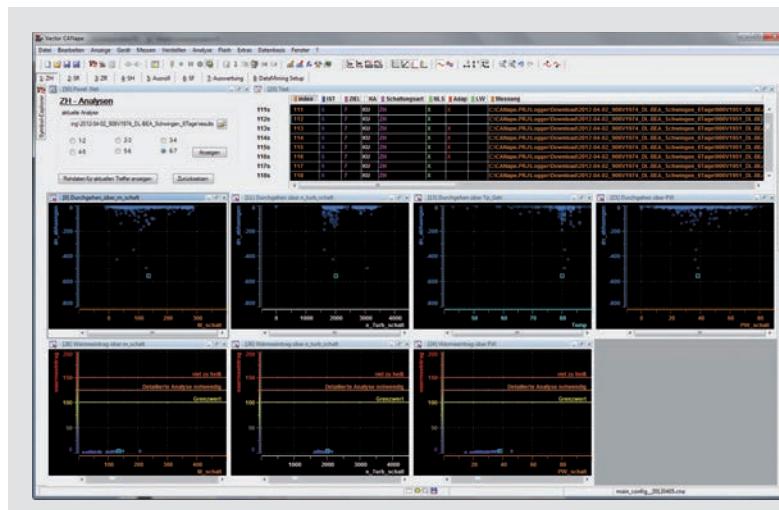
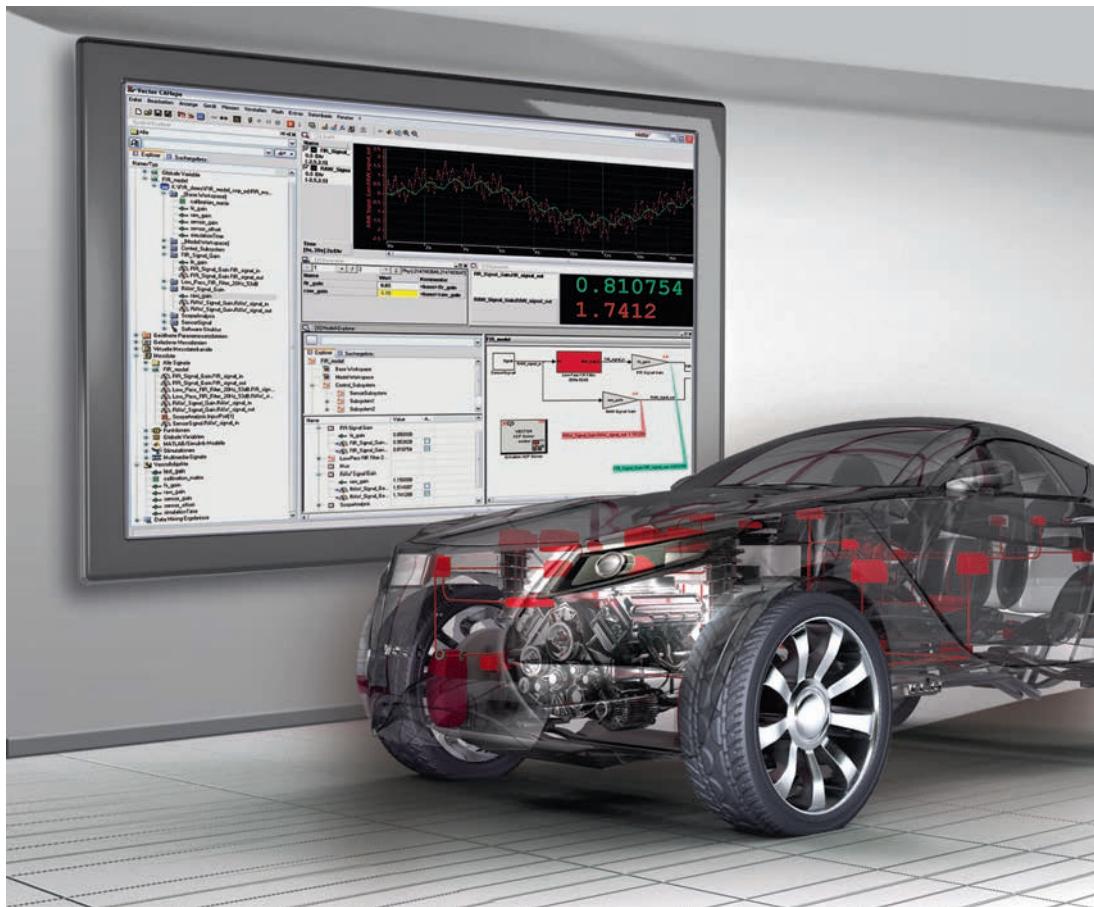


Figure 4:
Evaluation of traction upshift:
initial assessments of the dura-
bility run can already be made
based on information in the
predefined display windows.

Efficient Analysis of Model Behavior in ECU Function Development



The focus in ECU function development is always on finding the best possible control algorithms and parameter combinations. A new solution now offers users a single measurement and calibration tool with universal application – from initial model design to the production level ECU.

In the framework of model-based software development, application functions are checked in an iterative process. This involves repeated runs of the model in Simulink from The Mathworks. Depending on the results, the developer may add function blocks by drag-and-drop operation from libraries. These blocks are parameterized either directly in the function block with a numeric value or by defining a parameter and its value on the MATLAB console. To modify an existing parameterization, the same steps are executed again, either by looking for a block in the model and changing its value, setting the parameter and its new value on the

MATLAB console, or modifying an M-script and running it. To visualize the signal responses that occur when the model runs in Simulink, the user instruments the model by attaching scope blocks to each of the desired signals.

Use of the standardized XCP measurement and calibration protocol gives the developer a considerably more user-friendly way to efficiently manage parameters and measure signals from the Simulink model without instrumentation.

Communication via XCP-on-Ethernet

The ASAM protocol XCP (Universal Measurement and Calibration Protocol) has become established as the preferred solution for measuring and calibrating applications in ECUs. This approach gives application engineers convenient access to measurement data and parameters in the ECU via standard bus systems at runtime.

An ASAM standard A2L database (ECU description file) provides all the information needed to access parameters and signals in the ECU. It contains descriptions of relevant data objects within the ECU's software, such as characteristic values (parameters, characteristic curves, maps) and measurement variables. The database also connects the object names to their memory addresses in the ECU and provides conversion rules for physical interpretation of the raw data. Using such a database, measurement and calibration tools can be used to acquire signal data or tune parameters as desired by the user. Only a protocol driver is needed in the ECU; it enables memory access at the ECU's runtime.

In the "Simulink XCP Server" option for the CANape measurement and calibration tool from Vector Informatik GmbH, an XCP driver is integrated into the Simulink model. As a result, the model is treated like a virtual ECU. The effort required to integrate the XCP driver is minimal: a single block from the Simulink CANape library is dragged and dropped into the model. Settings of the XCP transport layer – such as the host name and server port – can be configured in the block's dialog. They are necessary, since the "XCP-on-Ethernet" protocol is used to interconnect the measurement and calibration tool with the Simulink model (**Figure 1**).

After this parameterization step, the XCP Server is ready to use. The model's A2L description file is generated from the block's dialog. A virtual address is assigned to each Simulink object there. This is how the Simulink XCP Server explicitly implements address-oriented XCP operation for the Simulink objects. The user then selects objects in CANape in the usual way – by their names. An object's address is read-in from the A2L and is sent as information to the XCP Server in the model. This means that for data objects in the XCP Server, the address in the A2L file is only leveraged for

identification because the application's data objects do not yet have a real ECU memory address. After instrumenting the model, using it with CANape is easy and efficient because it is possible to generate a CANape project directly from the configuration of the block's dialog in Simulink and start CANape with the created project.

Numerous measurement and calibration functions accelerate model optimization

The user visualizes the desired measurement parameters in the measurement and calibration tool CANape – independent of the hierarchical organization and without further instrumentation of the model. It is possible to display any of the input or output variables of the model blocks and have their time responses displayed in graphic form. Parameters and signals can also be conveniently displayed and calibrated right in the visualization of the model (**Figure 2**). Simulink users will feel right at home in the familiar model representation that does not require conversion. In the model, a modified parameter is passed directly to the XCP Server via XCP. It adjusts values in the Simulink blocks and in the base or model workspace; this is equivalent to manually setting the values via the MATLAB console.

The function developer can change parameters of a full Simulink model, or those of one or more subsystems, easily and conveniently. This provides a way to test and optimize a Simulink model with different parameter sets. CANape supports different file formats here. Once the model has attained the desired maturity level, the relevant parameters are saved to a parameter set file. The parameter set management feature of CANape's CDM Studio (Calibration Data Management) is used to compare individual versions created during model optimization and merge parameter subsets or work packets to obtain optimal settings for the entire model. These settings may be exported in the form of a MATLAB M-script so that they can be used directly as a new version level in the MATLAB/Simulink development environment (**Figure 3**).

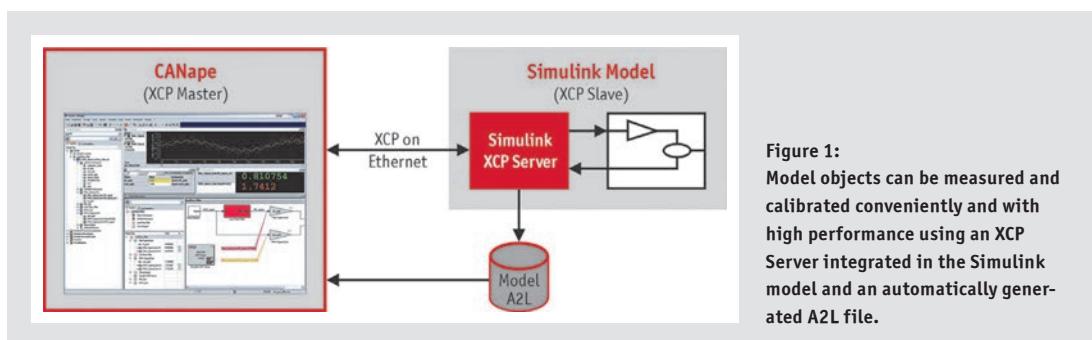


Figure 1:
Model objects can be measured and calibrated conveniently and with high performance using an XCP Server integrated in the Simulink model and an automatically generated A2L file.

MATLAB/Simulink as Time Master

Simulink models may run either slower or faster than real time, depending on their complexity and computer performance. This makes time stamps from Simulink indispensable. With each simulation step in Simulink, the associated time stamp is sent via XCP. Consequently, variations in the amount of the time needed for the simulation steps are irrelevant. Each simulation step corresponds to one time unit, regardless of how much real time is needed for it. In the process, Simulink acts as a time master, and the measured data sent out by the model can be visualized in CANape at simulation speed. Depending on the complexity of the model, sensor data from a one or two hour real test drive may be computed in just a few seconds or in minutes. If a user wishes to simulate especially large and complex models, standardized communication with XCP-on-Ethernet enables better computing performance, since two separate computers are used.

The simulation results may be analyzed either manually or through automation. In this process, an instance checks the received results and makes a parameterization decision. Serving as an instance is the CANape script language or an external software program that CANape controls via one of the existing automation interfaces.

Data from logged test drives may be fed into the model as an input vector at runtime to stimulate the model with real data. The function developer analyzes and optimizes system behavior under comparable constraints. This eliminates the need for real, cost-intensive test hardware entirely.

Summary

The implemented access to MATLAB/Simulink models via XCP in a measurement and calibration tool simplifies the work of function developers considerably. For example, the model is automatically instrumented via XCP, and this replaces the very tedious process of manual instrumentation. As a universal front-end for measurement and calibration tasks, CANape offers added convenience in the test phase of models in Simulink. When XCP is used as a universal protocol over the entire development process, this reduces overall process complexity. The function development process is simplified and accelerated, since just one protocol, one description file, and one tool are used for all measurement, calibration, and stimulation tasks.

Translation of a German publication in
Automobil Elektronik, 8/2009

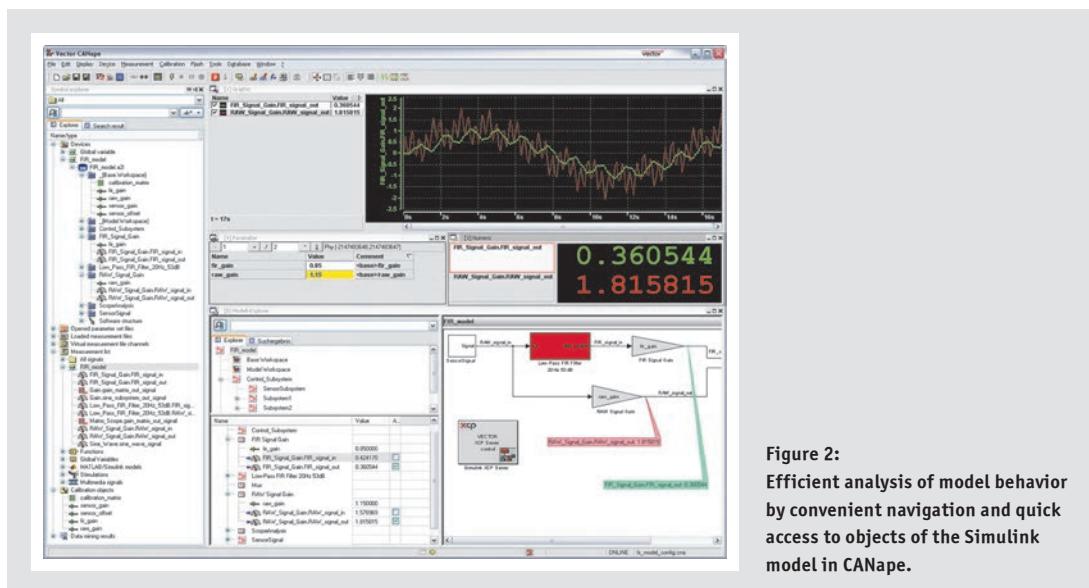


Figure 2:
Efficient analysis of model behavior
by convenient navigation and quick
access to objects of the Simulink
model in CANape.

André Ebner

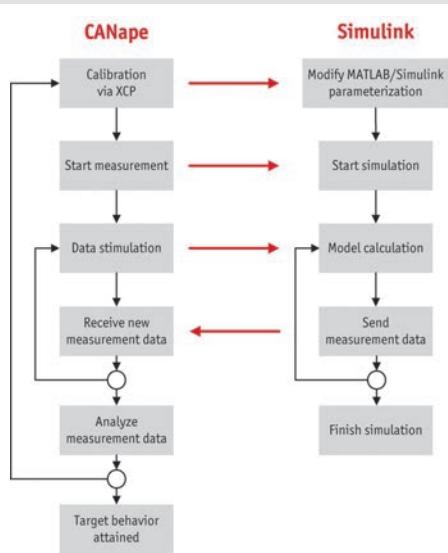
is employed at Vector Informatik GmbH as Technical Team Leader for the “Measurement and Calibration” product line.

Andreas Patzer

is employed at Vector Informatik GmbH as Business Development Manager for the “Measurement and Calibration” product line.

Wojciech Przystas

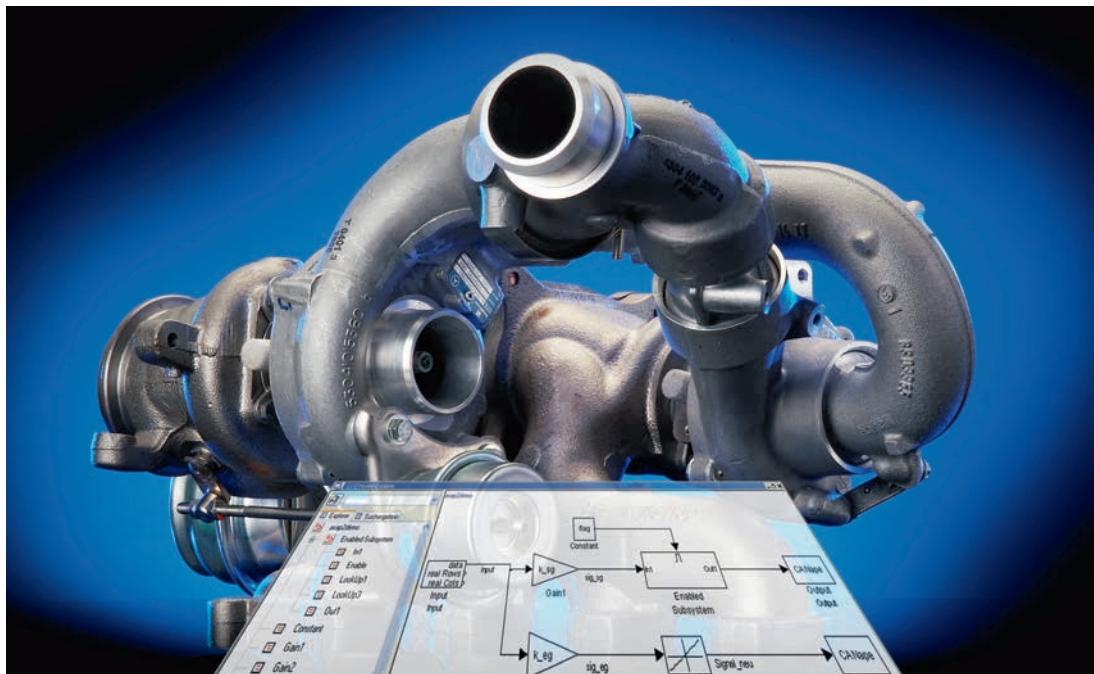
is employed at Vector Informatik GmbH as a Software Development Engineer for the “Measurement and Calibration” product line.

**Figure 3:**

Overview of actions in CANape and their effects on the model in Simulink

Accelerated Turbocharger Development

Efficiently developing control concepts with a cost-effective rapid prototyping solution



Turbochargers help engines, especially those with comparatively small displacements, to develop considerable torque and a high level of driving dynamics. Today's engine charging systems must flexibly adapt to engine speed and momentary power requirements; therefore, turbocharger control requires careful optimization. For the automotive supplier BorgWarner Turbo Systems, use of the CANape software tool has produced enormous streamlining potential in developing demo vehicles and hardware equipment for road durability tests.

"Charging" of combustion engines is a core technology when it comes to fulfilling requirements for low fuel consumption, low hazardous emissions and quality over long service life, while simultaneously improving driving dynamics. Today, 98% of diesel vehicles are equipped with turbochargers in the passenger car area, and on trucks approx. 80%. With the introduction of direct injection, turbocharging is also being used more frequently to improve the efficiency of gasoline engines, although the considerably higher exhaust temperatures make it essential to use higher grade materials that are more expensive.

Development competence and innovative force in demand

When a gasoline engine is driven at high power output, the turbine can become white-hot at exhaust temperatures of up to 1050°C. Simultaneously, charger speeds typically reach 220,000 rpm, and

on smaller turbochargers, e.g. on the Smart car, they may even reach 300,000 rpm. Accordingly, the challenges in the development of turbochargers lie in the areas of materials engineering, cooling, bearings and high-precision manufacturing/balancing of the rotating components.

The company BorgWarner Turbo Systems with headquarters in Kirchheimbolanden, in the German state of Rheinland-Pfalz, is one of the leading producers of turbochargers; it manufactures about 3.5 million charging systems annually in Germany and about 6 million worldwide. According to company information, BorgWarner in Kirchheimbolanden currently has the most advanced development center for turbochargers in the world. Besides various standard products for diesel and gasoline engines, its product line also includes advanced developments with multi-stage control of charging systems as well as the eBooster concept.

From the “turbo hole” to controlled charging

Due to its operating principle, high startup torque and high maximum power are mutually exclusive in simple turbocharger designs. That is, either a compact system is constructed for high charge pressure at low engine speeds or a large system optimized for high speeds is constructed that neglects the desired dynamics in the lower speed, which is then called a turbo hole.

Over the course of time, extended charging concepts have been developed to overcome this handicap, e.g. the wastegate charger equipped with bypass valve or the variable turbine geometry (VTG) that has become a standard today. Its adjustable vanes can be flexibly adapted to the exhaust gas flow. The latest developments include two-stage controlled charging with two charger systems in series and the eBooster, in which an electrically driven flow compressor supports the turbocharger.

Turbochargers increase complexity of engine control

The more refined the designs for charge pressure control, the greater the requirements for turbocharger control. In addition, it was necessary to acquire turbocharger speeds, exhaust gas backpressures and underlying parameters such as angles or positions of actuator elements by sensors and process them in the ECU. Actuators on the turbocharger consist of electrically or pneumatically actuated components for adjusting the vanes or actuating flaps and valves.

Control of the turbocharger is an integral component of engine control, and it therefore falls under the area of responsibility of the automotive OEM. In view of rising complexity, however, OEMs are relying on the support of turbocharger producers in implementing charge system control. In the development phase, BorgWarner uses the Matlab/Simulink program package to design the relevant

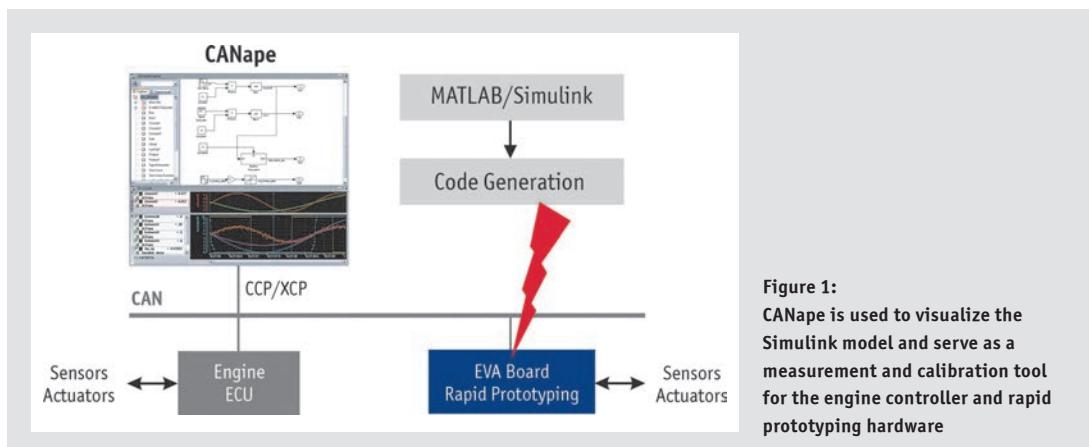
control concepts that are provided to the customer for use on test benches or in test vehicles.

Working efficiently on the visualized model

In calibrating prototypes in the engine or vehicle, Vector's CANape measurement, calibration and diagnostic tool performs valuable service. CANape is a powerful ASAM-conformant tool for all tasks related to ECU calibration. Via physical interfaces such as CAN, FlexRay or LIN and the standardized measurement and calibration protocols CCP (CAN Calibration Protocol) and XCP (Universal Measurement and Calibration Protocol), parameters can be measured from a PC at runtime, and they can also be calibrated at runtime. At BorgWarner the capability of visualizing the Matlab/Simulink models in CANape is particularly beneficial. Without requiring tedious searches through documentation, the user can recognize – directly from the visualized model – which parameters need to be adapted for the desired modifications. Search functions quickly lead to the desired parameters and support convenient navigation through levels of the model.

In the production vehicle, the turbocharger application is run in the engine controller. The sensor and actuator systems are also connected here. During development, the turbocharger application is swapped out to rapid prototyping hardware to enable flexible testing of different software levels. This hardware consists of evaluation boards with integrated power electronics for driving the actuators. This may involve use of an actuator/sensor combination that differs from the one in the production vehicle (**Figure 1**).

CANape's tasks here include calibration of the engine controller and the rapid prototyping hardware as well as visualization of the Simulink model. This is precisely how CANape closes the gap between the graphic representation of the model, which lets the user visualize the interrelationships between variables, and the A2L-based calibration of the application. The parameter files



created in optimization of the turbocharger application may of course be exported from CANape and read back into Matlab/Simulink. Since this layout is well-suited to both test stands and test vehicles, BorgWarner also implements it in road durability tests.

Outlook for the future

To achieve an even more efficient solution, BorgWarner is now testing CANape in conjunction with a PC as a rapid prototyping platform. In this case, the system makes use of a DLL generated from the Simulink model, which runs in the CANape context. The Matlab integration package supplied with CANape provides a CANape target with which the user generates CANape-specific code in the real-time workshop. The generation provides the DLL with an XCP interface, so that the user can access the DLL in measuring and calibrating as if it were running on a rapid prototyping platform (Figure 2).

Together with the PC that is present anyways, CANape replaces the prototyping hardware. If the XCP protocol is used in communication with the ECU, CANape can simultaneously be used as a bypassing coordinator. The ECU data is measured in real time via the tool, is passed to the compiled DLL model of the turbocharger control system, is processed there and written back to the engine controller ECU. The big advantage of this bypassing method is that the DLL can be calibrated exactly like a real ECU. Parameter

changes take effect immediately, without requiring the detour of making a modification in Simulink and then regenerating the code.

These applications point out the capabilities of high-performance development and diagnostic software in practice. CANape makes part of the hardware equipment superfluous, saves on licenses for modeling software and noticeably accelerates development progress due to fewer compiler runs. Calibration engineers at BorgWarner benefit from visualization of the Simulink model, including all of its key parameters. Even heightened real-time requirements do not pose any problem here, since bypassing cycle times as short as 2 ms are possible on PC platforms.

Translation of a German publication in Hanser Automotive, 11/2007

Links:

Homepage BorgWarner Inc.: www.borgwarner.com

Homepage Vector: www.vector.com

Product Information CANape: www.vector.com/canape

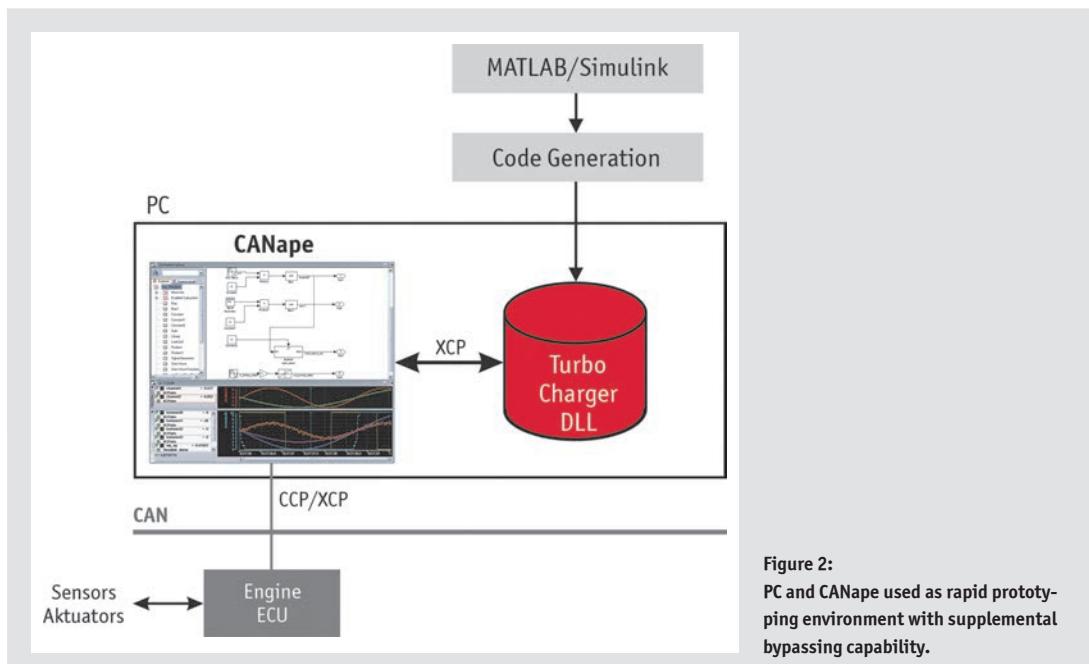


Figure 2:
PC and CANape used as rapid prototyping environment with supplemental bypassing capability.



Gerd Spinner, BorgWarner Turbo Systems
is employed in the Product Development area
at BorgWarner Turbo Systems Engineering
GmbH.



Andreas Patzer, Vector
is employed at Vector Informatik GmbH as
Business Development Manager for the
“Measurement and Calibration” product line.

Riding on the Razor's Edge

Optimal parameterization of an engine controller 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



Figure 1: To optimally calibrate the engine controller, calibration parameters are adjusted in real-time during the test run based on acquired measurement results.

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.

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.

Translation of a German publication in
Elektronik automotive, 4/2011

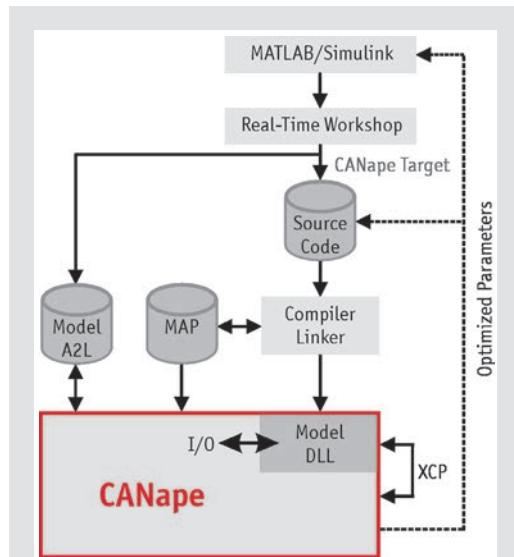


Figure 2: Besides being used to develop a control algorithm that is fed concrete data from ECUs, bus data, analog data, etc., CANape also covers other applications such as online computations during a measurement.

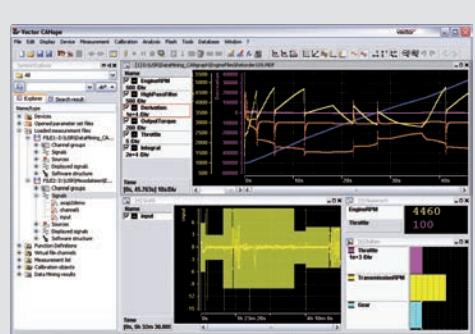


Figure 3: CANape visualizes the various parameters and offers convenient calibration options

Figures:

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.

Optimizing Driver Assistance Systems

Verification of Object Recognition Algorithms by Driver Assistance Systems



Driver assistance systems address the issue of growing traffic volume by offering significant relief to drivers. To obtain an objective assessment of control algorithms in the development of such systems, BMW is relying on the support of the CANape measurement and calibration tool. Many suggestions by Munich's leading car producer also flowed into the design of an extension that effectively handles the special requirements involved in calibrating driver assistance systems.

ACC (Adaptive Cruise Control) systems monitor the corridor in front of a vehicle, detecting vehicles ahead and maintaining the distance to these vehicles desired by the driver. ACC systems also adjust the car's current speed to the traffic situation by automatically reducing engine torque, braking, and accelerating again, if necessary. To maintain the correct distance to the vehicle ahead at any speed, ACC systems require very complex and precise computing algorithms.

What sounds relatively simple is in reality a great challenge for the development engineers, because a driving situation that a human driver is able to evaluate effortlessly is a nearly endless array of numbers in an ACC ECU. A forward-looking radar unit supplies coordinates of objects in cartesian form, i.e. in the X direction (car's longitudinal axis) and Y direction (car's transverse axis), or as polar coordinates with vehicle distance and azimuth angle. From these, the ACC ECU concludes whether the distance to the car ahead is sufficient, whether braking needs to be initiated or whether it is possible to accelerate.

The evaluation electronics must also decide whether the acquired object should even be considered as a relevant control object, because the aperture angle of the radar sensors also detects any objects adjacent to the roadway. While radar scanning initially finds many objects, only the data of vehicles in one's own lane may be utilized for adaptive distance control.

This is not a trivial task, since information from the radar sensor system is not always clear and unambiguous. Some of the radar reflections are bumps in the road or simply false reports. This indicates just how important it is to conduct a check of the acquired data (signals) based on visible evidence (the video image). The reliability and operating safety of this system, with its acceleration and braking maneuvers, is in the truest sense a matter of life or death. Faulty behavior could lead to vehicle reactions that are incomprehensible to the driver.

For this reason, additional data are utilized at BMW to determine the exact distance between vehicles and exclude irrelevant objects.

Besides dynamic driving data, data from GPS navigation are also used, for example.

Since there were no suitable products on the market, BMW initially supported the ACC development process by a tool it had written in-house, which helped engineers reach an objective assessment of the control algorithms. For production development, however, BMW is now increasingly relying on standard products that can also be used by its suppliers.

Tool-supported evaluation of sensor data and driving impressions

Ever since the CANape Option "Advanced Multimedia" (AM) became available, BMW has been using this tool more intensively on projects and in production development. The tool's standardized calibration protocols and flexible interfaces enable simple integration into an existing tool environment, leave room for engineering extensions and offer maximum future compatibility, e.g. for obtaining objective test results to evaluate the assistance systems of suppliers.

Even the base version of the measurement, calibration and diagnostic tool CANape from Vector is able to record all ECU-internal data time synchronously (**Figure 1**):

- > Signals from CAN, LIN and FlexRay bus with the CCP/XCP calibration protocols
- > Peripheral measurement technology
- > Video and audio signals, and
- > GPS signals (optional).

To achieve optimal control functionality in the ECU, it is necessary to make numerous parameter modifications in an iterative process that can be performed online or offline with CANape. BMW utilizes the CANape Option "Advanced Multimedia", an extension especially designed for developing driver assistance systems. A core element here is the Multimedia Engine, which displays ECU signals and information computed from them in 3D perspective in the video display. The relevant ACC coordinates can then be placed over the video image as defined bitmap information in 3D perspective (**Figure 2**). Only by means of such visual "matching" is it possible to objectively assess the original mass of numbers. The BMW developers no longer just get coordinate information on the positions of objects ahead of the driver; they can also immediately observe and verify them in a video image – from a bird's eye perspective or side view. Thanks to the saved information, it is possible to study real driving situations – which are normally never one-hundred percent reproducible – in the laboratory and efficiently adapt the control algorithms.

Environment detection with the camera

A coordinate transformation is necessary to represent object data from the ECU as geometric objects in the Video Window. To calibrate the connected camera, all the BMW developers need to do is click eight reference points whose coordinates are known. Based on stored information, the tool automatically computes a coordinate transformation for each detected object – from global coordinates to image coordinates – and then displays the objects accordingly in the video image.

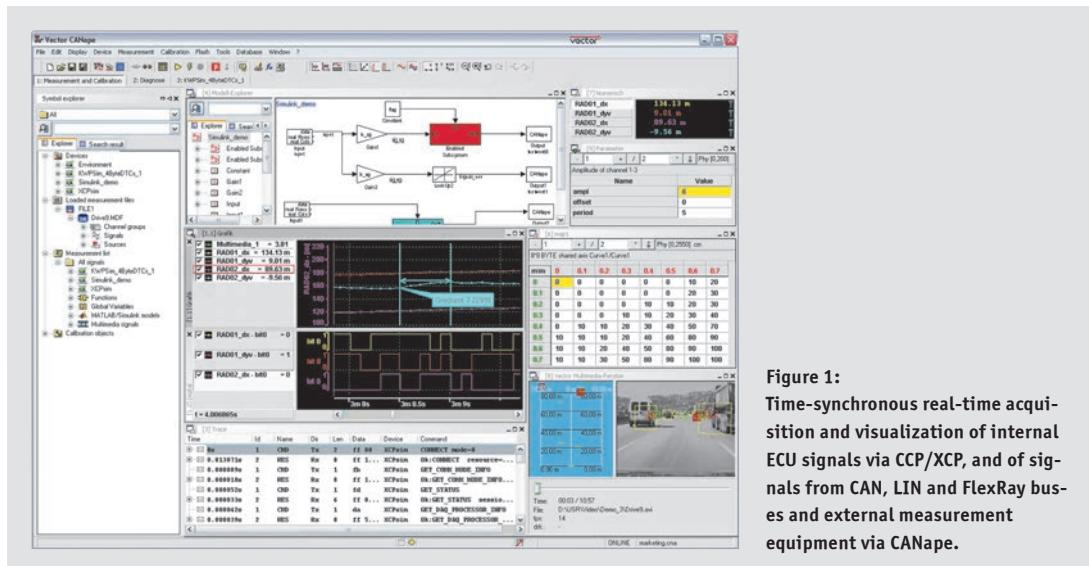


Figure 1:
Time-synchronous real-time acquisition and visualization of internal ECU signals via CCP/XCP, and of signals from CAN, LIN and FlexRay buses and external measurement equipment via CANape.

Vector also offers a suitable camera for the system, since BMW is not the only customer who values a universal and tested solution. ECU developers using CANape AM can use practically any desired camera, from a webcam to a professional high-tech device, provided that it has a USB or Firewire port and a DirectX interface. Optimal results are obtained with a camera that has a logarithmic characteristic and 120 dB dynamic range that further enhances image quality – both in tunnels and in direct sunlight. It is also possible to connect analog cameras via a Frame Grabber interface. Depending on the camera resolution, image refresh rate and number of cameras used, data might accumulate at rates of 20 MByte/sec or more. That is why developers work with reduced image resolution; data compression units are also used to reduce the data volume.

A number of standard pixel graphics are provided for object representation in the video image. For example, a detected vehicle is represented by a rectangular frame, and other objects might be shown as an "X" or a line. In the process, it does not matter whether the ACC information is obtained from radar, infrared or ultrasonic sensors. It is also easy to assign signals to an object with the user-friendly GFX Editor for graphics (**Figure 3**). In another dialog of the GFX Editor, the user defines the type of visualization for the specific object type. This involves defining any objects desired and linking them to the desired graphic symbols. In addition, users can also integrate their own graphics.

Joint advanced development work

For BMW, cooperative teamwork with the tool producer is a prerequisite for developing intelligent high-end systems. In this project, good cooperation has led to ideas that have spawned real func-

tional extensions. For example, today – at the request of BMW developers – a so-called "driving tube" is generated with data from the ACC ECU; it is then represented in the video image as either a bird's eye view or a 3D perspective. This driving tube corresponds to a virtual driving lane that demarcates the presumed future path. This defines the corridor ahead of the vehicle that is relevant for distance computation. Objects detected by the ACC system lying outside the driving tube do not need to be considered in distance control, and they are therefore represented by a different frame color. Also part of the evaluation is highlighting traffic signs and traffic lights. Theoretically, the tool can be used to represent up to 50 different objects simultaneously.

Similarly high requirements are placed on the hardware. The volume of accumulating data and enormous computational demands on the computing platform are still a great challenge. Previously, two PCs were needed to assure the specified performance. But this required manual data synchronization, since only one of the computers could access the internal ECU signals. Today, a dual-core computer platform is used for the ACC computations. Since parallel recording of multiple video sequences and processing of FlexRay data place increased demands on the CPU, BMW experts are seeking more balanced utilization of the two computing cores.

Outlook

By visually comparing those objects detected by the ECU with the real environment, BMW developers now have an easy and efficient way to verify the object recognition algorithms of their ACC ECU. Cooperation between BMW and Vector is bearing further fruit, e.g. improved processor loading of dual-core and quad-core computing platforms in future CANape versions and functional extensions for



developing parking assistance systems. In the next few years, safety systems will also be implemented based on environmental data acquisition. They require even higher levels of computing performance due to the need for comprehensive detection of the surroundings and networking of active safety systems to Adaptive Cruise Control sensors. CANape AM will also let BMW developers focus entirely on their core tasks in the future: considerable increases in driving convenience and further safety improvements in highway transportation.

**Translation of a German publication in
Hanser Automotive, 2/2005**

Lorenz Eisenknappl,

Vehicle Dynamics Development: Team Leader for Control System Measurement Technology, BMW AG

Walter Kagerer,

Vehicle Dynamics Development, Driver Assistance and Active Safety, ACCSnG, ACC and DCC Calibration, BMW AG

Harald Koppe,

Vehicle Dynamics Development: Measurement Technology for In-Vehicle Control Systems, BMW AG

Martin Lamprecht,

Vehicle Dynamics Development: Measurement Technology for In-Vehicle Control Systems, BMW AG

Alexander Meske,

Vehicle Dynamics Development: Driver Assistance and Active Safety, ACCSnG, ACC and DCC Calibration, BMW AG

Alfred Kless

is Business Development Manager for the "Measurement & Calibration" product line at Vector Informatik GmbH.

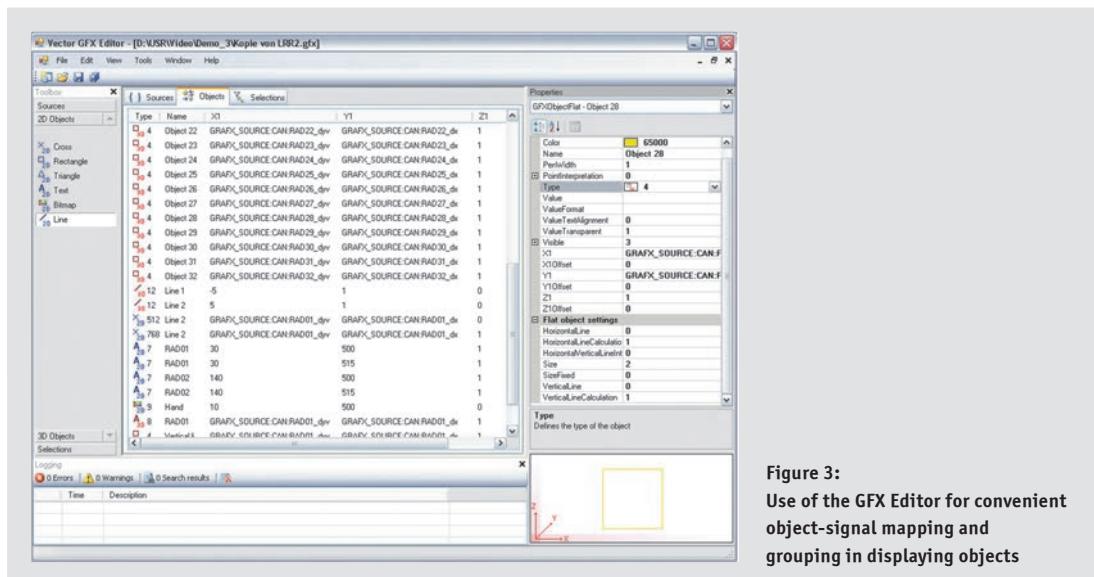
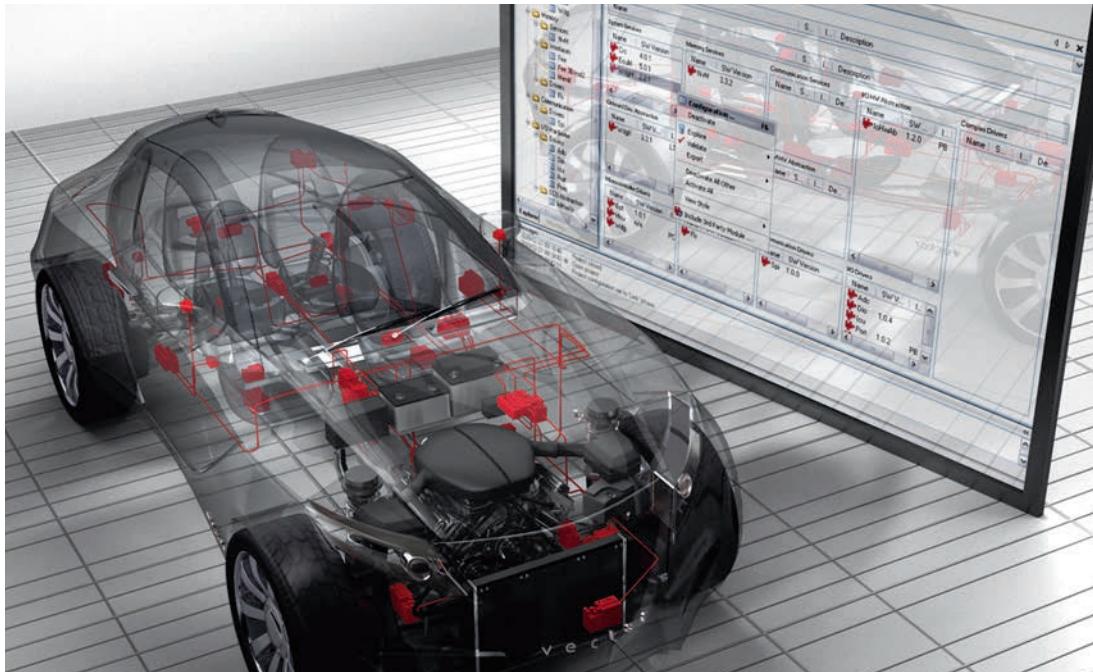


Figure 3:
Use of the GFX Editor for convenient object-signal mapping and grouping in displaying objects

AUTOSAR: New Paths to ECU Software – Part 1

Iterative collaboration between OEM, TIER1 and software supplier



A primary reason for introducing AUTOSAR, besides standardizing the basic software, is to increase reusability of the functional software. This affects the cooperation between the partners involved as well: OEM, TIER1, semiconductor manufacturer and software supplier. This first part of a two-part series describes a foundation for successful collaboration: AUTOSAR-specific exchange formats and tools. In the second part, you will learn about the significance of AUTOSAR for everyday work in developing ECU software for production projects.

Each OEM has its own functional requirements for the ECUs in its vehicles, especially when it comes to communication and diagnostics. These requirements are implemented in OEM-specific software. If a TIER1 supplies an ECU to several OEMs, it must manually modify the ECU software for each project. Even if the functional software is already decoupled quite well from the software, so that it can be adapted to the OEM-specific requirements, this modification effort is still work intensive and prone to errors. **Figure 1** shows how unmodified functional software is adapted to different vehicle projects without AUTOSAR.

One goal of AUTOSAR is to minimize these adaptation efforts in software integration. Therefore, AUTOSAR focuses on consistent abstraction of the software from the hardware and partitioning of the software into modules with defined functional scope and precise interfaces. These modules may be combined and, most significantly, they can be substantially configured to cover the

requirements of different OEMs. This eliminates manual modification of the software and facilitates its reuse. Defined interfaces make it possible to replace OEM-specific software components (e.g. for diagnostics) with just minimal effort.

AUTOSAR reference architecture

The AUTOSAR reference architecture is described in the document AUTOSAR Layered Software Architecture [1]. In this document, the ECU software is organized into the three parts shown in **Figure 2**:

- > The functional software consists of software components (SWCs). The SWCs are created, independent of the ECUs, based on a Virtual Function Bus (VFB), and they communicate with one another via interfaces.
- > The Runtime Environment (RTE) is used for executing the SWCs and it includes the technical implementation of the VFB in a real ECU.

- > The basic software (BSW) modules handle the basic functions of an ECU. They also offer higher-end standard services such as management of ECU states and diagnostic services.

The RTE is the layer between the functional software and the basic software modules. It provides all interfaces the SWCs need to access data and services of the BSW modules. Examples are signal values from the communication network (CAN, LIN or FlexRay), I/O signals and standard services of the BSW modules. The interfaces originate from the "SWC Description" files. Moreover, the RTE handles execution of the SWCs and communication among the SWCs with the help of the operating system.

The BSW modules are subdivided into three layers per the AUTOSAR architecture [1]:

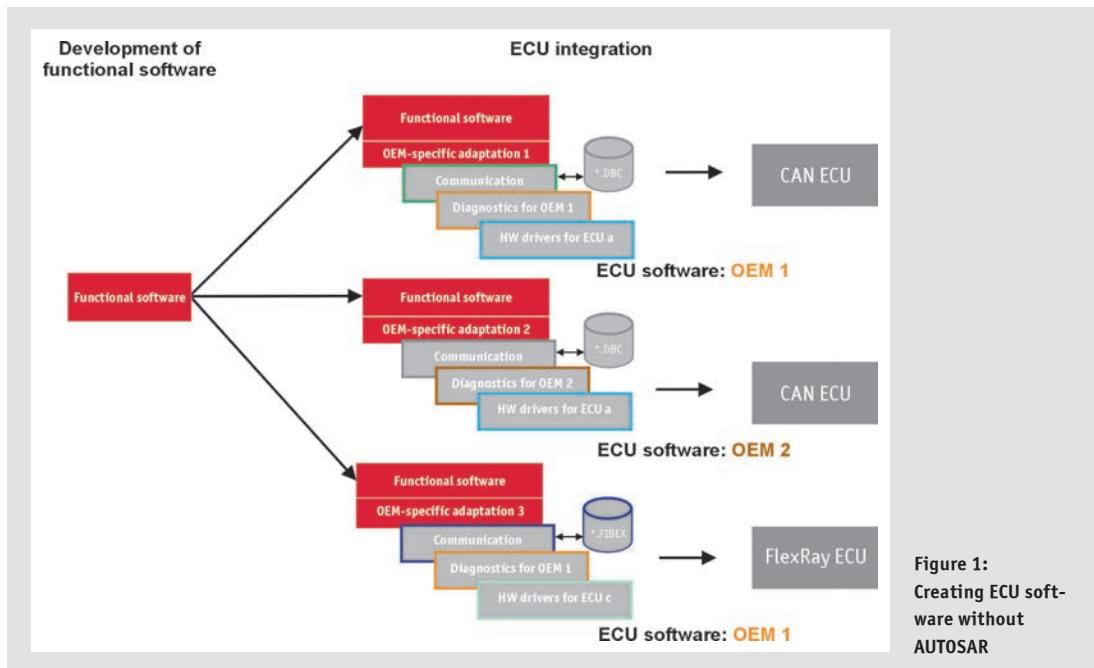
- > Service Layer
- > ECU Abstraction Layer
- > Microcontroller Abstraction Layer (MCAL)

The BSW modules of the Service Layer play a special role here, because they contain standard services for the functional software that are accessed via special interfaces within the RTE. The second part of this article, in the next issue, will describe the configuration of these services in greater detail.

The AUTOSAR Release 3.0 defines approx. 50 different configurable basic software modules; some of them are very complex. The

majority contains functions that were already usually present in previous software architectures, but now they are more precisely distinguished from one another. Consistent partitioning of functions into individual software modules is what assures the desired hardware abstraction and scalability for different types of ECUs. The use of such standard modules increases the quality of the ECU software. In most cases, this standardization covers the interfaces as well as functions of the BSW modules. Representing an exception here are the BSW modules for diagnostics. Since diagnostic processes are very dependent on manufacturing and after-sales processes at the OEMs, AUTOSAR only defines the interfaces of the diagnostic modules. This allows OEM-specific implementations of the diagnostic modules. Vector provides these modules for many OEMs, and it is the task of the TIER1 supplier to configure and integrate the specific variant.

Both the BSW modules and the RTE are available as software products from various software suppliers (TIER2), such as the MICROSAR products from Vector, which offer coverage of all BSW modules and the RTE per AUTOSAR Release 3.0. Although they are standard software products, the BSW modules and the RTE must be adapted to project-specific constraints (OEM, vehicle model, ECU variant). This involves use of relevant PC-based tools during the configuration process. For example, the RTE may be configured with DaVinci Developer and the BSW modules with DaVinci Configurator Pro from Vector.



AUTOSAR Methodology

The AUTOSAR Consortium has defined a method for developing ECU software, the AUTOSAR Methodology [2]. This document essentially subdivides the development process into three activities and standardizes data exchange between development partners with a set of XML files:

> Activity: "Component implementation"

The TIER1 or OEM defines the SWCs. For this purpose, it creates an XML file for each SWC, the so-called "SWC Description". This file describes the SWC's interfaces and resource requirements.

Afterwards, the TIER1 or OEM creates the related C files for the implementation of the SWC.

> Activity: "System configuration"

The OEM first defines the functional scope of the entire vehicle based on the SWCs, independent of the ECUs. The next step is to design the communication networks and distribute SWCs to the available ECUs. The result is saved in the "System Description".

For each ECU, the OEM reduces the "System Description" to an "ECU Extract of System Description" which the OEM can pass to the supplier (TIER1) of the relevant ECU. This file replaces the DBC, FIBEX or LDF files previously used to configure the BSW modules.

> Activity: "ECU design and configuration"

Starting with the "ECU Extract of System Description", the TIER1 integrates its own SWCs. This results in a complete and up-to-date "ECU Extract of System Description", which now contains the description of all SWCs (from OEM and TIER1) of an ECU.

Another prerequisite for ECU configuration is the existence of the "BSW Module Description" files, which contain the definition of the data structures and all configurable parameters of a BSW module. These files are implementation-specific and – along with the generators and the static code – are part of the BSW modules.

Afterwards, the TIER1 creates the initial "ECU Configuration Description" (activity 2 in **Figure 3**) based on the current "ECU Extract of System Description" and the "BSW Module Description" files. Then the TIER1 begins to configure the ECU and documents it in the "ECU Configuration Description". The TIER1 uses suitable tools for configuring and checking parameters of the BSW modules and the RTE for this purpose (activities 3 and 4 in **Figure 3**). The "ECU Configuration Description" is the foundation for ECU-specific generation of the RTE and the BSW modules by the relevant generators.

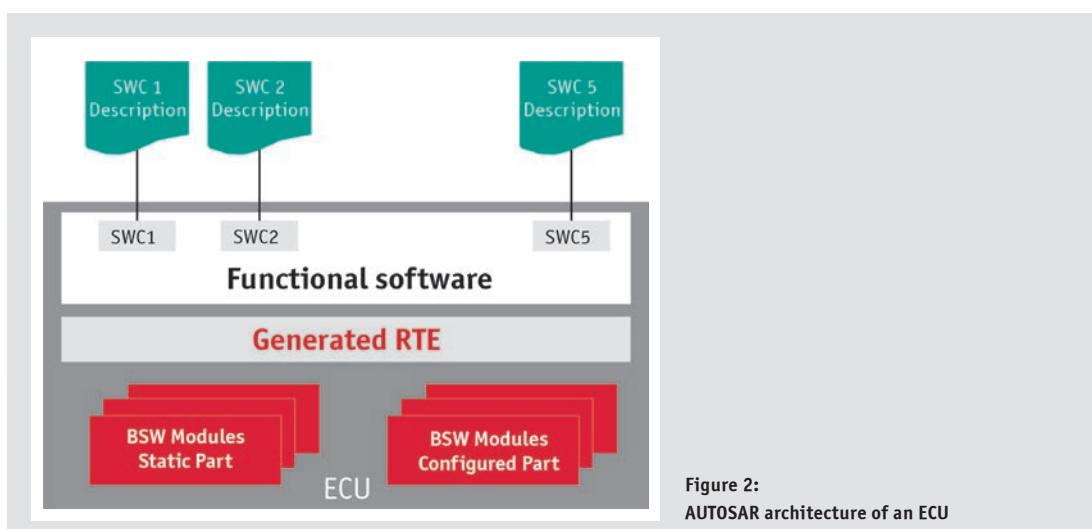


Figure 2:
AUTOSAR architecture of an ECU

The AUTOSAR method is flexible and suitable for satisfying the practical requirements of different projects or different OEMs. For example, use of SWCs in the "System Description" is optional.

Figure 3 shows – based on the example of the tools DaVinci Developer and DaVinci Configurator Pro from Vector – how the "ECU design and configuration" activity can be implemented with tool support.

Configuring and integrating all software components

During the configuration process defined in AUTOSAR, the TIER1 selects – from its component collection – those SWCs it needs for the ECU's functionality. Afterwards the TIER1 integrates them in its ECU, together with the BSW modules and RTE. This shifts the primary work of integrating the ECU software from manual code adaptation to tool supported configuration of the BSW modules and the RTE.

Since the current level of the AUTOSAR specifications still has some room for interpretation, from today's perspective it is advisable to procure either the entire BSW package, or at least defined BSW clusters, from a single source. The advantage is that the software supplier (TIER2) can already perform an integration test on these modules. However, it is also possible to procure individual modules from different TIER2 suppliers or use modified BSW modules. This increases integration effort, however, since both functional integration and integration in the configuration tools need to be performed.

Essentially, all MICROSAR BSW modules are tested within systematic integration tests. As an integration partner, Vector can extend its integration services to software modules from thirdparty producers, such as MCAL drivers, upon request.

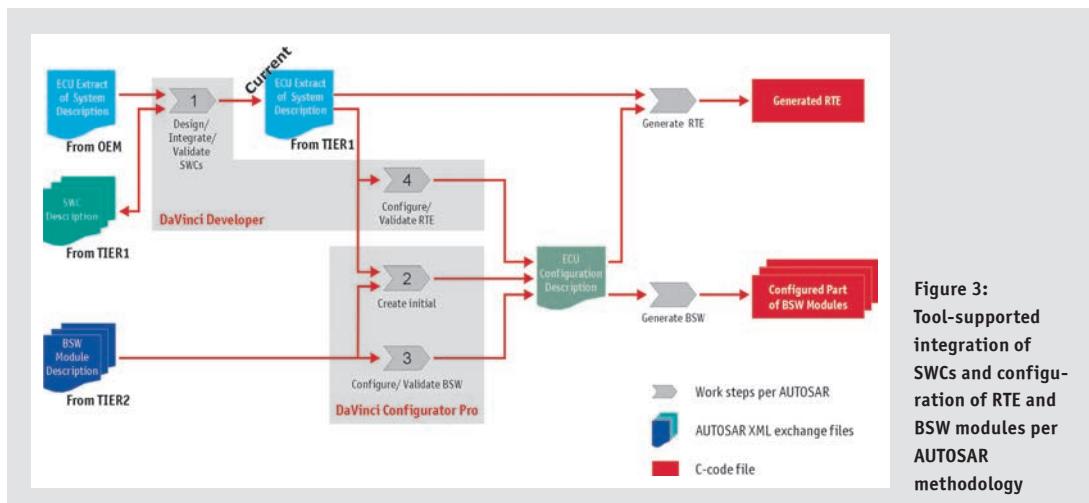
For configuration of the BSW modules, the TIER1 needs the support of a universal tool with user-friendly functions. That is why Vector developed DaVinci Configurator Pro. It supports three use cases:

- > Configuration of MICROSAR BSW modules from Vector
- > Configuration of AUTOSAR BSW modules from third-party manufacturers
- > Configuration of software modules you have created yourself

MICROSAR BSW modules are configured by using a graphical user interface that shows the complex interrelationships of the configuration parameters in simplified form. Furthermore, parameter selection is limited to valid input values, and this prevents setting implausible values.

The Generic Configuration Editor (GCE) defined in AUTOSAR is included with DaVinci Configurator Pro to configure the BSW modules from third-party producers. As an alternative, the TIER1 supplier may choose to develop a user-friendly and integrated configuration user interface for these modules itself. This may also be done with the newly developed DaVinci Configurator Kit. It is used to create "BSW Module Description" files for the software modules, define user-friendly user interfaces, establish validation rules and create code generators for generating the executable code. The TIER1 can also use this approach to configure its own BSW modules, e.g. complex device drivers.

Both DaVinci Configurator Pro and DaVinci Developer contain validation rules that supplement the AUTOSAR method. They ensure that individual parameters as well as complex parameter groups and their interdependencies are validated and that the "ECU Configuration Description" is generated consistently. This consistency is essential for subsequent generation of the RTE and the BSW modules.



In the second part of this article, you will learn – based on examples of selected use cases – how the exchange files and configuration tools are used in practice. The process of creating a complete set of AUTOSAR-conformant ECU software for a specific OEM is explained, and the article describes how to maintain the software over time or modify it for a different OEM.

**Translation of a German publication in
Elektronik automotive, 11/2009**

All Figures:
Vector Informatik GmbH

Literature:

[1] Layered Software Architecture:
http://www.autosar.org/download/specs_aktuell/AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] AUTOSAR Methodology:
http://www.autosar.org/download/specs_aktuell/AUTOSAR_Methodology.pdf

Links:

Homepage Vector: www.vector.com
Product information about AUTOSAR: www.autosar-solutions.com



Pascale Morizur (Dipl.-Ing.)

studied Physics-Electronics at the Grande Ecole ICPi in Lyon (France). After graduating in 1986, she worked for 10 years in advanced development for CAN, J1939 and diagnostics at MAN Commercial Vehicles. Since 2005, she has been employed at Vector as Product Manager in the area of Embedded Software Components.



Matthias Wernicke (Dipl.-Ing. (FH))

upon graduating in Industrial Electronics at the Polytechnical College at Ulm, was employed for four years at the Daimler Research Center in Ulm, Germany. Since early 2000 he has been working at Vector Informatik in Stuttgart, developing methods and tools for the design of distributed electronic functions in motor vehicles. Today he is responsible for product management of DaVinci AUTOSAR tools.



Justus Maier (Dipl.-Inf. (FH))

studied Computer Science in Regensburg. He began his professional career as a developer of standardized software in the insurance industry. For 8 years he was involved in design and advanced development of tools for ECU configuration in the AUTOSAR field. He has been employed at Vector since 2006 as technical Product Manager for DaVinci Configurator Pro.



►► Solutions for **AUTOSAR**

Successfully Developing AUTOSAR ECUs

Benefit from the Vector AUTOSAR solution for AUTOSAR 4.x and 3.2:

- > **PREEvision** and the **DaVinci tools** support the whole AUTOSAR process – from system design to the configuration of the basic software.
- > **MICROSAR** includes the complete basic software for a wide variety of microcontrollers.
- > Use **MICROSAR Safe** to develop safety-related ECUs according to ISO 26262-ASIL D.
- > The combination of **MICROSAR.AMD** and the test tool **CANoe.AMD** supports you in fast startup and testing of AUTOSAR ECUs.
- > Accelerate your ECU development with professional **AUTOSAR services** such as training, workshops and project work.

Experience, dedication and practice-proven products make Vector the smart choice in developing series production AUTOSAR ECUs.

- Information and downloads: www.autosar-solutions.com

AUTOSAR: New Paths to ECU Software – Part 2

AUTOSAR in Practice – Life Cycle of AUTOSAR ECU Software

The first part of the article covers the structure of AUTOSAR-conformant ECU software, the AUTOSAR development method and helpful auxiliary tool support. The second part of the article presents realistic scenarios illustrating how AUTOSAR ECU software is maintained over its life cycle.

During the development of an AUTOSAR ECU, code generators are used to adapt the basic software (BSW) and runtime environment (RTE) to specific ECU requirements. Generation is based on the following extensive description files mentioned in the first part of the article:

- > The “ECU Extract of System Description” contains the ECU-specific section of the overall system.
- > The “SWC Description” describes the interfaces and structure of the AUTOSAR software components (SWCs), and it may already be included in the “ECU Extract of System Description”.
- > The “ECU Configuration Description” contains the configuration of the BSW modules and RTE.
- > The “BSW Module Description” describes the configurable parameters of a BSW module.

The challenge for the developer is to keep these description files consistent while avoiding the need to recreate the whole configuration whenever a change is made. In performing this task, it is helpful to use configuration tools that support iterative work processes during an ECU development project. Typical scenarios and change cases are described as well as having work steps explained based on the Vector tools DaVinci Developer and DaVinci Configurator Pro.

New development of software components (SWCs)

Depending on the OEM, the “ECU Extract of System Description” might already contain the interfaces of SWCs. In some cases, these SWCs are still incomplete. For example, OEM-specified application interfaces may be included, but not the implementation description (runnable entities) or interfaces to the standard services of the BSW modules. The Tier1 developer can add this missing specification with the help of a tool such as DaVinci Developer. The Tier1 may also create its own SWCs (Step D1 in **Figure 4**).

There are two different ways to implement the SWCs: either manually based on a generated code template or with the help of model-based development (MBD) tools such as MATLAB®/Simulink® EmbeddedCoder® or TargetLink. The “SWC Description” is imported into the MBD tool where it serves as a basis for modeling the SWC. Code generators are used to automatically generate SWC implementations.

If SWCs are already available at the Tier1, they can also be integrated in the ECU. This involves importing the relevant “SWC Description” and linking the SWC to the other SWCs of the ECU (Step D2).

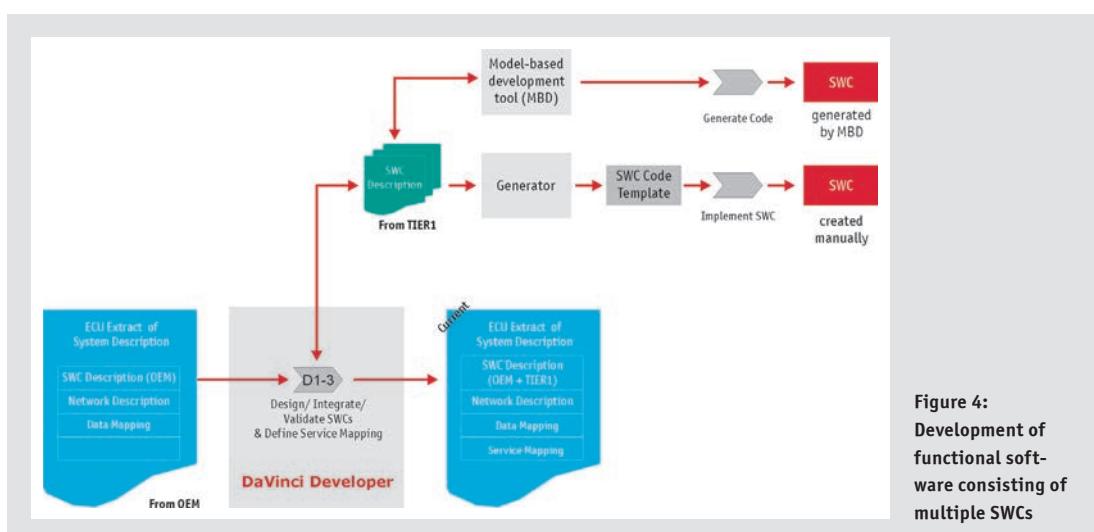


Figure 4:
Development of
functional soft-
ware consisting of
multiple SWCs

In a separate integration step, data elements of the SWC interfaces are assigned to bus signals (data mapping) – provided that the OEM has not already defined such a mapping in the “ECU Extract of System Description”.

Typically, the “ECU Extract of System Description” will change a number of times over the course of a project. When the Tier1 receives a new version, the SWCs it contains might also be modified. A special Update function makes it possible to accept changes in DaVinci Developer without losing any extensions previously made by the Tier1, such as implementation descriptions or interfaces to standard services.

Configuration of standard AUTOSAR services

One challenge arising in the configuration of AUTOSAR ECUs is how to configure the standard services of the BSW modules. Within the “ECU Extract of System Configuration”, the services are represented by special SWCs, the so-called service components. These service components are created and integrated with the SWCs by what is referred to as Service Mapping, which can be performed automatically with tool support (Step D3).

In the “top-down” approach, the need for services is determined from the entire functional software, and the services of the BSW modules are configured accordingly. This approach makes sense for those services that are not typically specified by the OEM in detail. Examples are services of the NVM (Non-Volatile Memory Manager) or the ECUM (ECU Manager).

In the “bottom-up” method, the service is first configured in the BSW module, e.g. based on OEM requirements. The SWCs are then extended to match the service configuration. An example of this is the DCM (Diagnostic Communication Manager).

Just like the services, the ECU’s I/O interfaces are also represented by a special SWC – the I/O Hardware Abstraction

Component. This SWC can also be created by a “top-down” or “bottom-up” approach.

At the end of the integration process, the Tier1 gets an updated “ECU Extract of System Description”. Besides the OEM’s original version, the file now also contains content defined by the Tier1 and is fully validated.

Modification of the RTE after changing SWCs

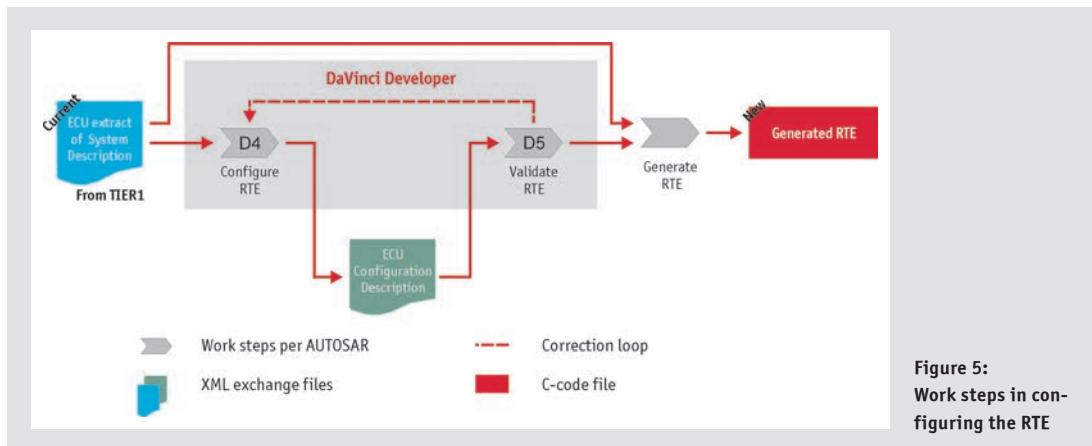
If only the implementation of a SWC changes without affecting the SWC’s interfaces or structure, neither the RTE nor the basic software would need to be reconfigured. However, if the structure of a SWC changes, e.g. due to the addition of a new runnable entity, the RTE configuration must be modified. This involves assigning the newly added runnable entity to a task. After this modification has been made (Step D4 in **Figure 5**), the RTE configuration is validated (Step D5).

The DaVinci Developer supports this process with detailed error messages and recommended corrections. For example, inconsistencies are reported so that the Tier1 can correct the “SWC Description” or RTE configuration.

Incorporating changed communication data from the OEM

A typical change scenario is updating the communication data of an ECU, e.g. because the distribution of signals to messages has changed. Such a change is only relevant to those BSW modules related to communication and does not require any changes to the RTE or SWCs.

Figure 6 shows the work steps that are performed to accept the changed communication data. The BSW modules can be adapted automatically: First, a new “ECU Configuration Description” is generated, and the contents of the old “ECU Configuration Description”



are converted to the new “ECU Configuration Description” (Step C1-2). All parameter values unaffected by the change are automatically accepted. Only the parameters of the new or modified signals or messages are then configured in Step C3. Finally, to ensure that all parameter values are consistent, the new “ECU Configuration Description” is validated in Step C4.

As a supplement to the AUTOSAR standard, Vector has implemented an “Update” function and validation in DaVinci Configurator Pro. If the OEM does not provide an AUTOSAR-conformant ECU Extract of the System Description, the Tier1 can instead use the familiar network description formats (DBC, LDF or FIBEX) as inputs to the DaVinci tools.

Switching over to a different processor or processor type

Thanks to the systematic hardware abstraction offered by AUTOSAR the TIER1 only needs to replace the affected MCAL modules when switching over to a different processor device or processor type. This transition is performed with tool support: The old MCAL modules are removed and the new MCAL modules are added to the project by selecting the new BSWMD files. The new platform-dependent parameter values that were added are checked in DaVinci Configurator Pro and reconfigured by the Tier1 as necessary (Step C3 for each changed MCAL module). Consistency of the “ECU Configuration Description” is restored by final validation (Step C4).

Especially useful to the Tier1 is the ability to extend the tool environment, e.g. to support the MCAL modules of future

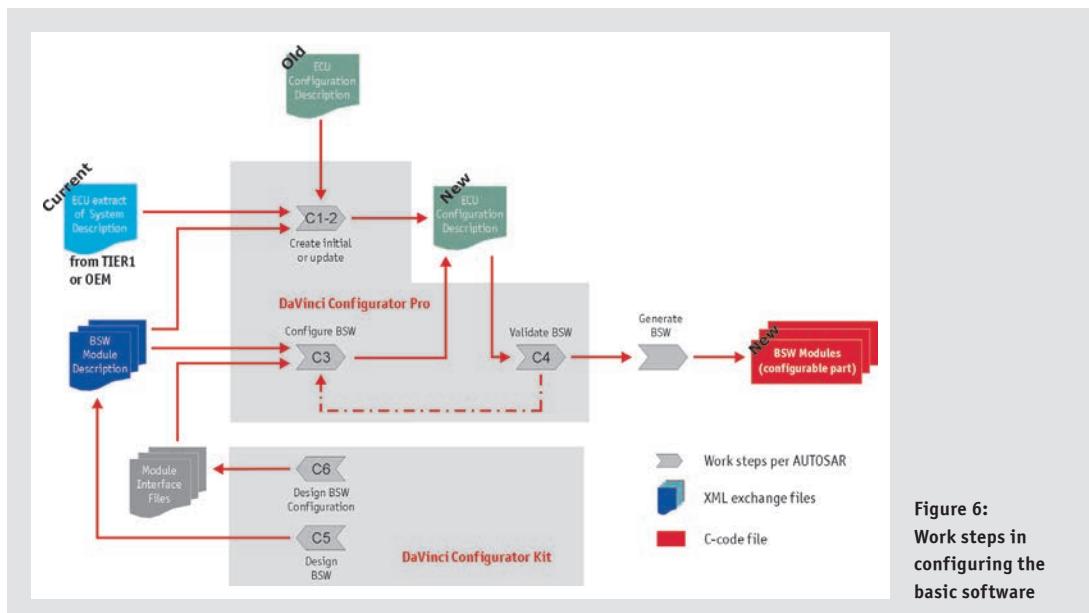
processors or own BSW modules. This is enabled by the DaVinci Configurator Kit, which is used to create the Tier1 BSWMD files and module interface files. They contain definitions of specific configuration interfaces, validation rules or generators for the BSW modules (Steps C5 and C6).

Replacing OEM-specific diagnostics

If an existing ECU configuration is to be used for a different OEM, the ECU’s diagnostic basic software must be replaced, because it is OEM-specific. This affects the DCM and DEM (Diagnostic Event Manager) BSW modules.

Figure 7 shows how this replacement is made. The Tier1 obtains the new CDD or ODX file from the OEM. These widely used formats contain the diagnostic description data for the ECU. They can be generated with tools such as CANdelaStudio from Vector. DaVinci Configurator Pro utilizes information from these files to automatically configure diagnostic BSW modules for the ECU (Step C7). Analogous to the modified diagnostic BSW modules, the DCM and DEM service components must also be modified and made known to the application SWCs in a “bottom-up” process. For this purpose, DaVinci Configurator Pro takes the “ECU Configuration Description” and generates the “SWC description” files which include service components for DCM and DEM (Step C8).

The Tier1 can now use DaVinci Developer to generate the service mapping for all diagnostic services of the new OEM. Validation of the SWCs ensures that no service is forgotten in the change



process. If the application SWCs does not yet offer the diagnostic services required by the OEM, they must be extended (Step D1-3), which in turn requires modification of the RTE (Step D4-5).

Changing I/O signals

In case a new sensor needs to be connected to the ECU the new I/O signal that it uses must be added to the "ECU Configuration Description". Therefore the Tier1 extends the configuration of the I/O hardware abstraction in DaVinci Configurator Pro by adding the new signal (Step C3 in Figure 6) and modifies the pin mapping in the I/O drivers in the MCAL layer. After successfully validating this configuration change, an updated SWC is generated for representation of the I/O Hardware Abstraction. By importing this SWC into DaVinci Developer, the Tier1 can extend the SWCs of the functional software so that they can access the new sensor.

Advantages of the AUTOSAR configuration process

The AUTOSAR principle "configuring instead of programming" enables early validation of the ECU software architecture. This prevents errors from occurring in a late phase. Nonetheless, the AUTOSAR formats are extraordinarily complex. Good tool support is essential to handle the changes that are typically required over the course of a development project.

**Translation of a German publication in
Elektronik Automotive, 12/2009**



Pascale Morizur (Dipl.-Ing.)

studied Physics-Electronics at the Grande Ecole ICIPI in Lyon (France). After graduating in 1986, she worked for 10 years in advanced development for CAN, J1939 and diagnostics at MAN Commercial Vehicles. Since 2005, she has been employed at Vector as Product Manager in the area of Embedded Software Components.



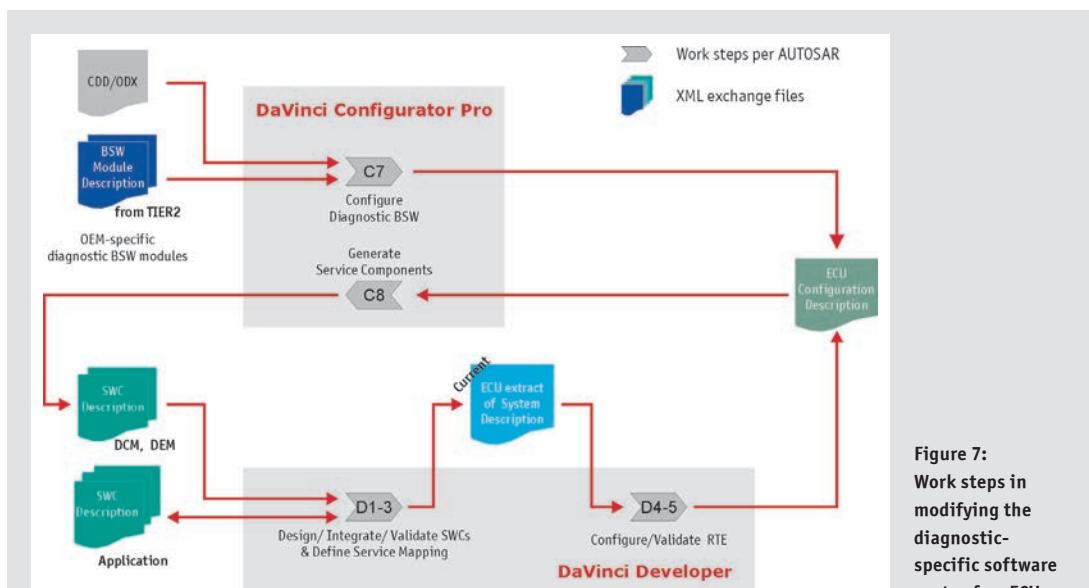
Matthias Wernicke (Dipl.-Ing. (FH))

upon graduating in Industrial Electronics at the Polytechnical College at Ulm, was employed for four years at the Daimler Research Center in Ulm, Germany. Since early 2000 he has been working at Vector Informatik in Stuttgart, developing methods and tools for the design of distributed electronic functions in motor vehicles. Today he is responsible for product management of DaVinci AUTOSAR tools.

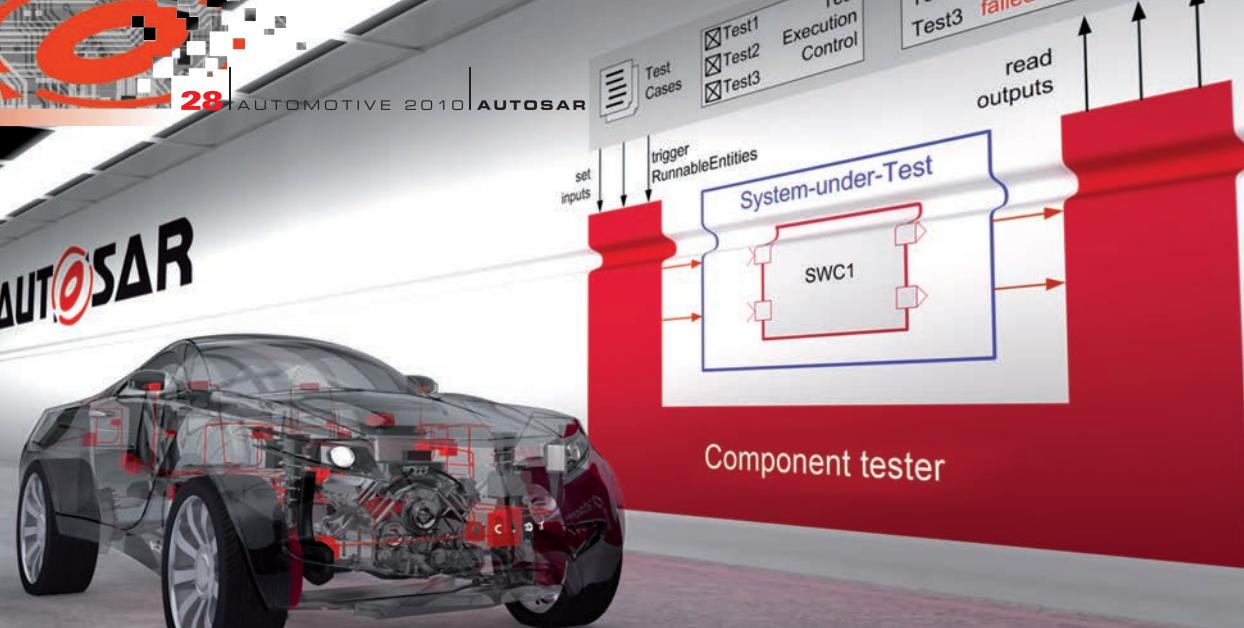


Justus Maier (Dipl.-Inf. (FH))

studied Computer Science in Regensburg. He began his professional career as a developer of standardized software in the insurance industry. For 8 years he was involved in design and advanced development of tools for ECU configuration in the AUTOSAR field. He has been employed at Vector since 2006 as technical Product Manager for DaVinci Configurator Pro.



**Figure 7:
Work steps in
modifying the
diagnostic-
specific software
parts of an ECU**



ANALYZING INDIVIDUAL
SOFTWARE COMPONENTS INSTEAD
OF ENTIRE ECUS

Early Testing of AUTOSAR Components

Software components and the Virtual Functional Bus of AUTOSAR define a new layer in the application software. This layer enables testing of individual software components and the communication between them. This article explains how you can test parts of the application even during early development phases and thereby simplifying the later integration of the software components.

AUTOSAR is becoming increasingly important to the entire ECU design process. The AUTOSAR methodology and its description formats have become the current state-of-the-art method for specifying the in-vehicle software architecture. Key AUTOSAR concepts are "Software Components" (SWCs), "Runtime Environment" (RTE) and the "Virtual Functional Bus" (VFB), as well as interfaces specified and standardized in AUTOSAR.

On the one hand test tools for analyzing and checking ECUs must understand the AUTOSAR interfaces and read-in the description formats. But, on the other hand the new component layer within the software architecture is increasing diversification and allows certain time independence in ECU development. This requires analyzing and testing of individual SWCs or groups of SWCs instead of entire ECUs. Thus module and integration tests on this component layer are necessary for an individual ECU.

Description of the Network Communication

In the AUTOSAR design methodology, bus communication is transparent as a central basic service for the application and is therefore also transparent to the SWCs; which is defined in the higher-level AUTOSAR system description. This description is derived from the distribution of SWCs to the ECUs and from the interface ports of the SWCs.

In testing an AUTOSAR ECU, the test tool must know the communication so that it can interpret the signals in the messages. Until now, communication databases have been read-in using the FIBEX or DBC exchange format. The process of converting the "AUTOSAR System Description" into these formats is time-consuming and error-prone. That is why the system description is read-in directly into CANoe since version 7.2, significantly improving the user friendliness of the analysis and testing tool from Vector.

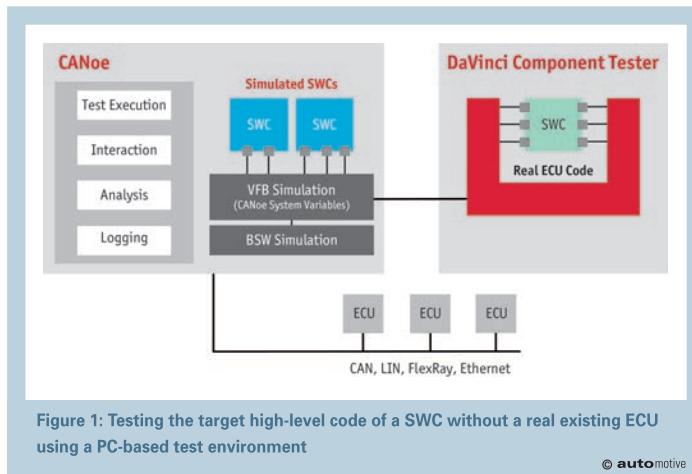


Figure 1: Testing the target high-level code of a SWC without a real existing ECU using a PC-based test environment

© automotive

Testing individual software components instead of entire ECUs

AUTOSAR methodology offers finer granularity of the software units under test. Instead of testing entire ECUs or parts of ECUs, in the future SWCs will be tested either individually or in combination. Real ECUs are tested using a remaining-bus simulation to simulate the non-real ECUs. Comparable to this method, in the future the SWCs will be executed in a runtime environment and tested with simulation of the remaining SWCs.

Use of the DaVinci Component Tester as a SWC runtime environment is advisable in early development phases, in which no real ECUs are available as a runtime environment. The DaVinci Component Tester provides a PC-based runtime environment for the real ECU code of one or more SWCs. CANoe uses an emulated VFB to interface the SWC to bus communication, inputs and outputs (I/O) and to a model-based simulation of other SWCs. The VFB in CANoe consists of system variables for each SWC port and, if applicable, routing of these variables to or from the bus or the I/O. The DaVinci Component Tester connects these CANoe system variables to the ports of the real SWC code (Figure 1) using an RTE emulation.

Familiar CANoe test functionality is used to test real ECUs, the ECU's target C-code of a SWC and completely simulated SWCs (MATLAB model / CAPL code), including automatic generation of a test report.

Communication analysis between software components

For integration tests of the SWCs on an ECU, the test tool requires a powerful interface to analyze internal ECU communication between the SWCs via the RTE. In the implementation planned for CANoe, no distinction is made

of whether the SWCs under test are SWCs in a real ECU or whether they are simulated SWCs on the PC. Just as the internally emulated VFB is mapped by system variables, RTE signals in the ECU are also accessed by system variables. In the future, these RTE signals might be acquired by a measurement over XCP-on-FlexRay. However, high-resolution traces require very high throughput in logging the internal ECU signals. That is why in the future the ECU and CANoe test environment will also be interconnected via a JTAG or Nexus Debug interface (Figure 2). This will let users read-in or adjust

internal signals of the RTE or basic software (BSW) in CANoe – quickly and with high data throughput.

Summary and Outlook

Today, it's just as simple and feasible to test and analyze AUTOSAR software components as it was to test and analyze an entire ECU previously and indeed in all phases of development – from fully simulated SWCs via partially simulated systems or the production code of an SWC to an entire AUTOSAR ECU. One future consequence is that interfaces to the test system will no longer be just real interfaces on the outside of the ECU; instead, they are more likely to be virtual interfaces within the ECU.



Dr. Carsten Böke is Senior Software Development Engineer for the „Tools for Networks and Distributed Systems“ product line, Vector Informatik GmbH.

Vector Informatik GmbH
www.vector.com

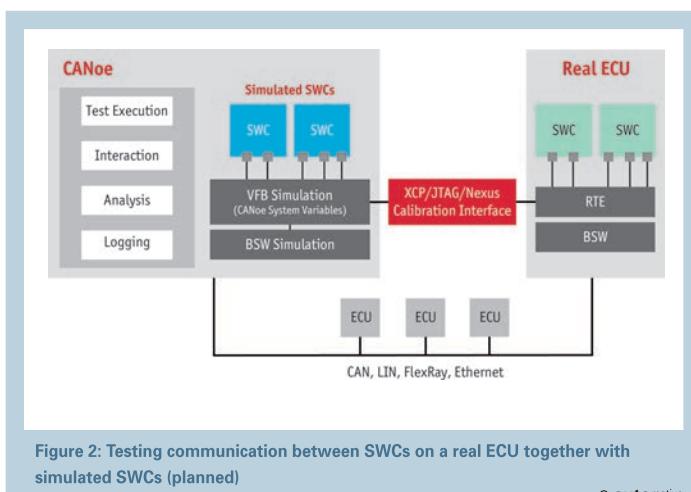


Figure 2: Testing communication between SWCs on a real ECU together with simulated SWCs (planned)

© automotive

Revealing Runtime Conflicts

AUTOSAR operating system directly measures task execution times

Software design is easy when the processor can provide far more computing time than the application needs. In practice, however, just the opposite is often the case. Good real-time embedded programming requires a lot of time and effort by the software engineer to steer clear of the cliffs of insufficient computing time. Now an embedded operating system offers welcome assistance; not only does it determine runtimes of tasks and interrupts directly, the “TimingAnalyzer” analysis tool also reveals runtime conflicts.

Modern embedded software is often complex and consists of multiple processes (tasks) running time-independently and interrupt service routines (ISRs). The use of a pre-emptive operating system like MICROSAR OS from Vector Informatik makes a significant contribution toward realizing a clear structure, shortened development times and better maintainability. The interactions of all implemented tasks must also be considered to obtain a stable system. Primary areas of focus are to achieve data consistency when accessing commonly used data areas and to optimize the runtime behavior of individual tasks.

Mechanisms provided by the operating system help to ensure data consistency. However, the developer is responsible for correct runtime behavior. The developer must ensure that all processes can be completed within their time limits.

Consider a case in which a user should be able to notice an effect within 10 ms after actuating a control; this is a typical example of a time limit for execution of a task. However, developers are usually confronted by periodic processes in which it is essential to

complete every operation before the beginning of the next cycle. Another aspect relevant to software design is that a running task may be interrupted at any time by an interrupt or a higher priority task.

Figure 1 illustrates what is required based on an application consisting of Task A and interrupts ISR 1 and ISR 2. The task is called periodically every 10 ms and requires an execution time of 5 ms; this results in 50% processor load. The execution time of the two interrupts is 3 ms each. Although interrupting the task with just one interrupt is noncritical (cycle 2), when both interrupts occur in the same cycle this leads to a computing time of 11 ms, which is unacceptable since it would extend beyond the time of the next starting point (cycle 5). Such time conflicts may lead to sporadically occurring and difficult to identify errors such as data losses, timeouts, perceptible delays, etc. Therefore, a thorough analysis is essential before delivery of the software.

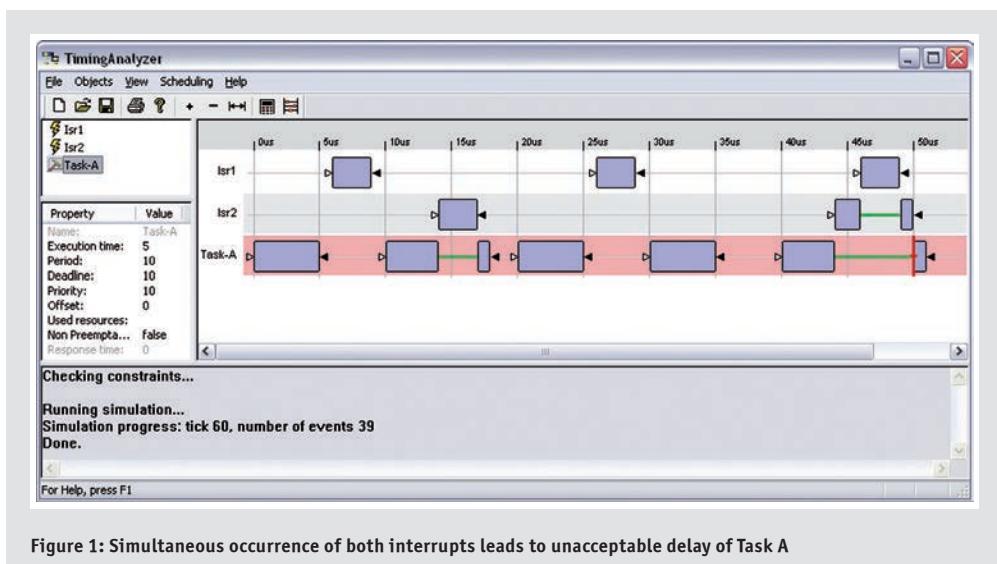


Figure 1: Simultaneous occurrence of both interrupts leads to unacceptable delay of Task A

The right way to measure execution time in the operating system

Each analysis is based on precise knowledge of task and interrupt execution times, which can be determined by various methods. A simple solution is to insert special test routines for toggling IO ports at the beginning and end of the tasks. The IO ports are then observed, e.g. via an external oscilloscope or logic analyzer. Another option is to trigger special measurement routines that generate time stamps via their own timers and pass their data to a PC via a (serial) data interface.

The disadvantages of these methods are evident:

- > At many points, the code needs to be modified manually: this involves substantial effort and there is high risk of errors.
- > The measured time interval, besides covering the execution time of the routine being analyzed, also includes the execution times of all interrupting tasks and ISRs.

Effective immediately, MICROSAR OS now offers a function for measuring the execution and blocking times of selected tasks and interrupts directly in the operating system, where interruptions are considered in calculating execution time. This simplifies the test engineer's work significantly, because now it is no longer necessary to modify the application code. All that is needed is a test routine for reading out the measured data. In principle, any supported serial or parallel interface of the processor or ECU hardware or a memory map in the emulator can be used for this. The user can

then conveniently switch the measurement function on or off in the operating system configuration. Figure 2 describes these simplifications.

Execution time analysis

After execution time measurements have been made in the operating system, the next step is for the software engineer to analyze the measured data and check whether each task can process its steps within the scheduled time slot. Vector Informatik supplies the "TimingAnalyzer" analysis tool with the operating system for this purpose.

The TimingAnalyzer applies the principle of Deadline Monotonic Scheduling. This approach is based on the assumption that for each task there is a time limit for processing its job. In the case of periodic tasks, this time limit must be less than the period. The system simulates aperiodic tasks as periodic tasks in which the period equals the minimum Inter-Arrival Time. The Inter-Arrival Time is the time between two task activations, e.g. two activations of the turn signal stalk.

The software engineer inputs the measured execution times of the tasks and ISRs into the TimingAnalyzer and supplements this data by indicating task priorities, deadlines to be met and typical cycle times. Based on these parameters, the TimingAnalyzer then checks to see whether all tasks always conform to their time limits, and it shows the computed time behavior graphically (Figures 1 and 3).

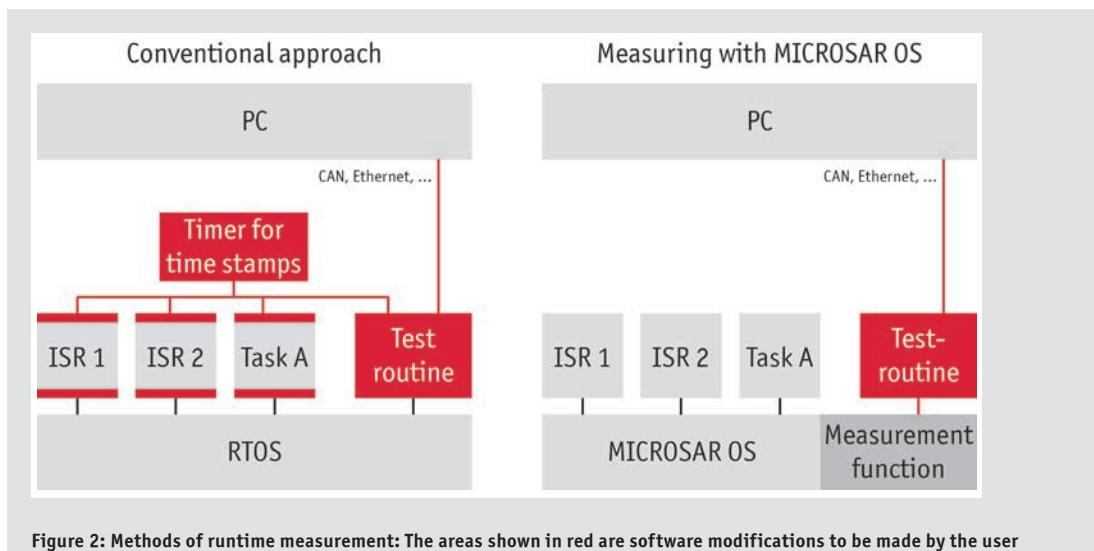


Figure 2: Methods of runtime measurement: The areas shown in red are software modifications to be made by the user

The analysis algorithm also takes into account the special dependencies of tasks and interrupts that share hardware or software resources.

Figure 3 shows a solution found with the help of the TimingAnalyzer for the example of Figure 1. ISR 2 is subdivided into a core function with an execution time of 1 ms on the interrupt level and a low-priority post-processing with an execution time of 4 ms on the task level. Such a scenario can be set up quickly in the TimingAnalyzer, and results are available within just a few seconds. This gives the developer a quick and easy way to check the effects of large changes made to the source code or new configurations.

Outlook

The extended capabilities of MICROSAR OS together with the TimingAnalyzer create an efficient duo for embedded software development. While the operating system simplifies the process of measuring execution times of tasks and interrupt service routines, the analysis tool lets users graphically display results and check for execution time conflicts. Currently, measured data are acquired by dedicated measurement routines and are transferred to the analysis tool manually. Future plans are automatically uploading the data to the PC and TimingAnalyzer using standardized protocols such as XCP.

Translation of a German publication in Automobil Elektronik, 1/2010

Literature:

- [1] www.asam.net
- [2] www.autosar.org

Links:

- Homepage Vector: www.vector.com
- Product Information MICROSAR OS: www.vector.com/autosar

All Figures:

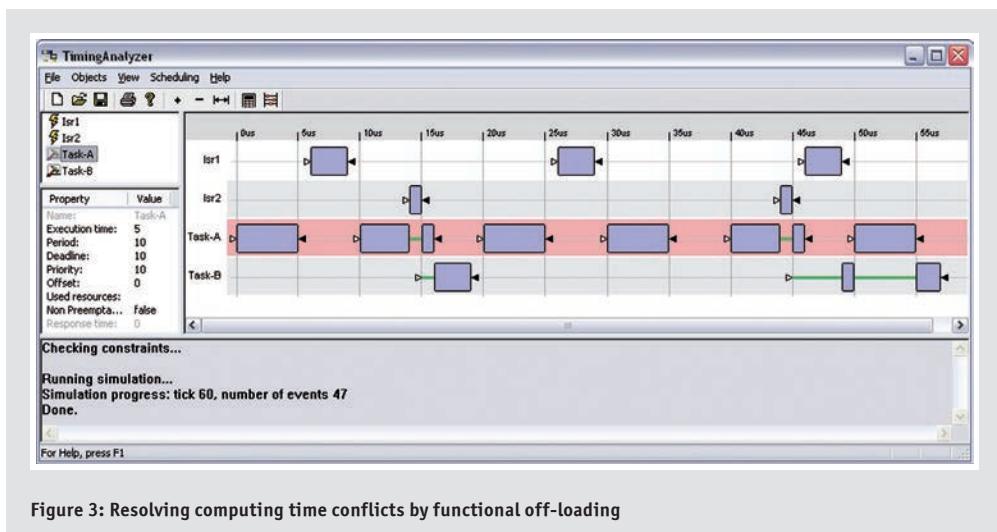
Vector Informatik GmbH



Dr. Helmut Brock

is product manager for the real time operating systems osCAN and MICROSAR OS in the productline "Embedded Software"

Vector Informatik GmbH
70499 Stuttgart
Germany





►► ECU Software

Accelerate Your Embedded Software Development

with standardized basic software. Benefit from scalable software modules for CAN, LIN, FlexRay, J1939 and CANopen used successfully by most OEMs worldwide.

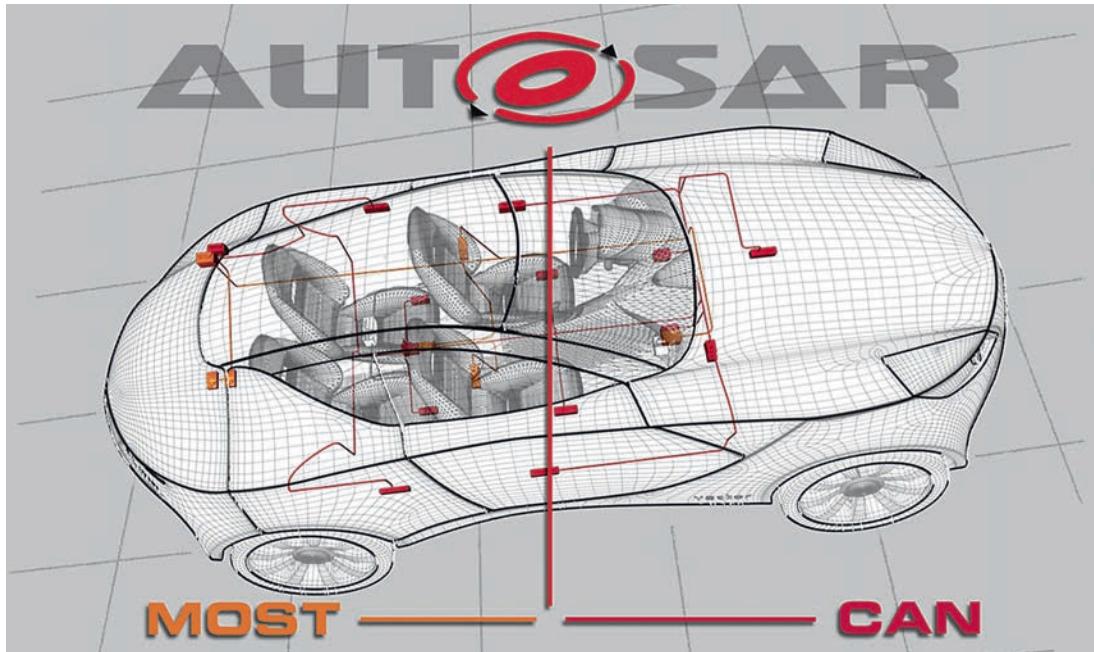
- > Build upon OSEK- or AUTOSAR-conformant operating systems which serve as the foundation for a stable implementation.
- > Create functionally reliable networks with the communication stacks that most developers rely on.
- > Use the Flash Bootloader to update your ECU software in a controlled process and protect it from third party access.
- > Focus on your application development tasks by using AUTOSAR-conformant basic software modules according to AUTOSAR 4.x and AUTOSAR 3.2.
- > Improve your efficiency with Vector's tailored consultation and development services.

Vector supports you with embedded software, configuration tools and services throughout the ECU development process.

► More Informationen: www.ecu-software.com



Implementing a CAN-MOST Gateway with AUTOSAR Basic Software



The gateway architecture defined in AUTOSAR makes it possible to interconnect different bus systems. However, at this time AUTOSAR only defines couplings of CAN, LIN, FlexRay and Ethernet buses. Despite this limitation, there are various solutions for implementing a CAN-MOST gateway based on AUTOSAR. But which is the right one? Vector Informatik presents three approaches and describes their respective advantages and disadvantages.

Fundamentals

Networking in modern vehicles is based on several different bus systems. CAN, LIN and FlexRay are used in the convenience and powertrain areas. Multimedia and infotainment ECUs are networked primarily over the MOST bus. The exchange of information between networks is the task of gateway ECUs. They exist in various forms – ranging from relatively simple gateways that only exchange minimal information between buses of the same kind to central gateways that interface to several buses of different types.

AUTOSAR offers ECU developers a solid software basis for interfacing their ECUs to CAN, LIN, FlexRay and Ethernet buses. Gateway tasks are handled by various software modules of the AUTOSAR architecture (**Figure 1**), depending on the specific functionality that is required.

PDU Gateway for Routing Entire PDUs

The PDU gateway is a part of the PDU Router (PDUR). This gateway routes entire data packets, Protocol Data Units (PDUs), from one

network to another. This type of routing assumes that the PDUs are defined identically on the source network and target network. This means that the PDUs must match in length and content. However, it must be noted that PDU routing is a spontaneous routing method: The PDU is immediately routed as soon as it is received. This means that it is not possible to convert the cycle time or send type. In some cases, however, such a conversion is necessary. Then routing must be performed by means of a signal gateway.

Signal Gateway

Often only individual signals of a PDU must be transmitted to another network, because only these signals are needed on the target bus. A second reason is that the source and target PDUs may be laid out differently, and a signal may be located at different positions. In signal routing, the received PDUs are first disassembled into their individual signals, so that they can then be copied to one or more target PDUs. Along with changing the composition of the PDUs, it is also possible to change the send type and cycle time. Routing parameters can also be modified flexibly. Signal routing is

performed by the Communication Module (COM). This module is located in the AUTOSAR Stack above the PDUR (**Figure 1**).

Transport Protocol Gateway

Another task of the PDUR is to route transport protocol data. This routing mechanism might be used, for example, to transmit an ECU's diagnostic data from one network to another. This involves receiving and transmitting the data over the transport protocol (TP). So, this type of routing is performed above Layer 4 of the ISO/OSI reference model, and it allows conversion to different addressing methods and various bus systems. TP data is routed by two mechanisms: One method uses the "store and forward" principle, the TP gateway receives the source data in its entirety before sending it to the target network. However, this mechanism is very memory-intensive and leads to longer latency times in routing. To keep delays and RAM requirements as low as possible in the gateway, the TP gateway also supports what is known as "on the fly routing". In this case, the gateway does not wait to receive all of the TP data first, but instead begins to send out the data to the target network at an early time point. So, it receives and transmits simultaneously.

Properties of the MOST Bus

The MOST bus has a ring topology. The ring consists of a Master ECU – also known as the timing master – that provides the system clock, and other ECUs known as timing slaves. Usually the timing master in the vehicle is contained in the Head Unit of the infotainment system. Currently, three baudrates are possible on the MOST bus:

25, 50 and 150 Mbit/s. MOST uses an optical physical layer. In addition, an electrical physical layer was specified for MOST50. An ECU's software is interfaced to MOST over the so-called Network Services. Network Services is a protocol stack standardized by the MOST specification.

On the MOST bus, data is transmitted over the synchronous and asynchronous channels, as well as the control channel. The synchronous channel is used to transmit video and audio data. Larger packet data whose transmission is not periodic, e.g. Ethernet tunneling or navigation data, is transmitted over the asynchronous channel. The control channel, whose bandwidth is vastly reduced, compared to the two other channels generally transports commands and status data.

The choice of the mechanism used to convert data from other networks to MOST depends on the data itself. Functional data (sensor data, states, etc.) is usually just a few bytes long; on the MOST bus, it is generally sent on the control channel. In the case of diagnostic data, which is also transmitted in larger blocks on CAN, it is advisable to send it on MOST using the MOST High Protocol (MHP). The MHP operates on the asynchronous channel, which has greater bandwidth than the control channel, which enables faster provision of the data packets to a specific receiver.

CAN-MOST Gateway as Hybrid Solution

A CAN-MOST gateway can be realized as a hybrid solution. In this context, a hybrid solution is a gateway architecture that uses a

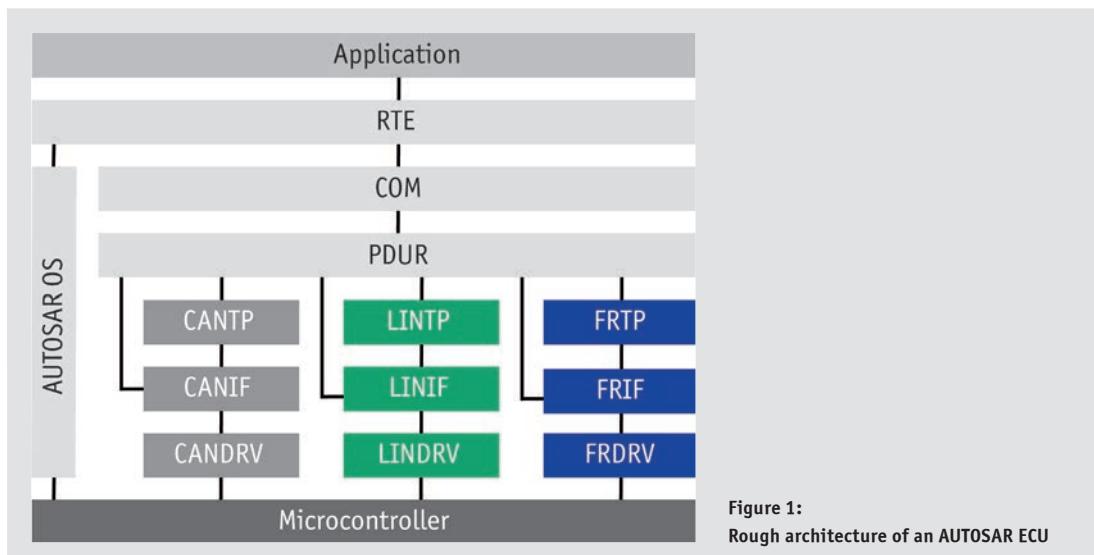


Figure 1:
Rough architecture of an AUTOSAR ECU

dedicated communication stack for each bus and implements the actual gateway by additional software layers. In this case, the specific gateway code must be contained in the application (**Figure 2**). Often, existing gateway software modules cannot be used in this approach, since interfacing to the other communication stacks is not supported.

To interface the different communications stacks, hybrid solutions frequently also require wrapper software, which leads to greater propagation time. Another disadvantage is the different configuration tools that need to be used for the communication stacks. Usually, the stacks can only be configured separately, so routing rules cannot be considered. This results in additional work effort whenever the communication descriptions are changed.

CAN-MOST Gateway with AUTOSAR Architecture

The Complex Device Driver (CDD) described in AUTOSAR represents a software module that allows direct access to the Basic Software Modules (BSW) and the microcontroller hardware. It also makes it possible to integrate microcontroller interfaces that are not defined in AUTOSAR. Implementation of a CAN-MOST gateway as a CDD (**Figure 3**), simplifies interfacing between the gateway and the drivers for CAN and MOST. As in the hybrid solution, any changes to routing rules must be handled afterwards at great effort. Another disadvantage of these two approaches is the redundant implementation of gateway functionality in the ECU and the additional memory required for this. Essentially, the CDD approach is comparable to the hybrid solution.

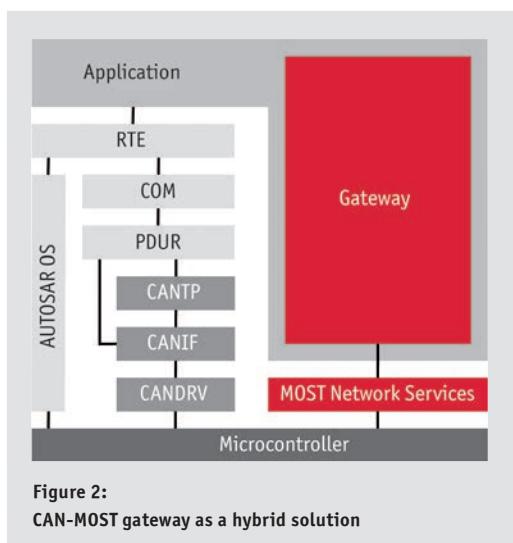


Figure 2:
CAN-MOST gateway as a hybrid solution

CAN-MOST Gateway with AUTOSAR Architecture

In its software architecture, AUTOSAR has solved the described problems in creating multibus gateways. So, it is natural to want to use this approach to develop a CAN-MOST gateway in the same way as a CAN-LIN or CAN-FlexRay gateway. To accomplish this, it is necessary to extend the AUTOSAR stack by adding a MOST bus interface.

According to the AUTOSAR specification, the higher layers to communication buses are interfaced via two BSW modules. The first is the hardware-specific driver, which controls accesses to the specific communication controller, and the second is an interface module, whose task it is to route the data of the underlying driver, independent of hardware, up to the higher layers. For example, in the CAN bus, these are the two BSW modules CAN Driver (CANDRV) and CAN Interface (CANIF). A similar driver and interface architecture is also used on the LIN bus and FlexRay bus. Above the interface layer, the bus systems are abstracted to the extent that the PDUR and COM modules can implement gateway functionalities hardware-independently.

Vector has chosen this architecture implementation as the best approach. Interfacing the AUTOSAR stack to the MOST bus requires a MOST Interface (MOSTIF) and a MOST Driver. Instead of implementing a MOST driver, existing MOST Network Services were accessed, similar to the way AUTOSAR 4.x provides for the Ethernet interface. No MOSTIP module was used, since this functionality is already included in MOST Network Services. In the framework of a project, Vector developed the MOSTIF module, which is part of the MICROSAR MOST package. The MOSTIF module is placed over the

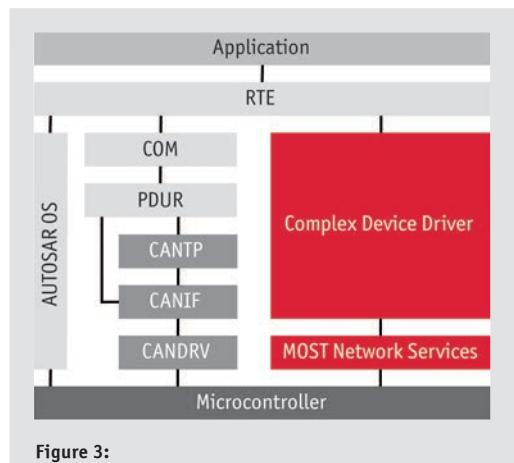


Figure 3:
CAN-MOST gateway as a Complex Device Driver

MOST Network Services, (Figure 4). The PDUR from Vector was extended to interface to the MOSTIF in addition to interfaces for CAN, LIN and FlexRay. The COM module – responsible for signal routing – was adopted without changes.

The architecture presented here allows the application to access MOST Network Services, so that existing MOST software modules can continue to be used. The Vector AUTOSAR stack has long supported the CAN, LIN and FlexRay bus systems and more recently Ethernet as well. The gateway mechanisms available for this are reused for the MOST bus. It is very easy, for example, to implement sending of data by the MOST notification mechanism in the AUTOSAR environment via the trigger-transmit interfaces. TP data received by the CAN bus can be sent on MOST over the MHP or over the control channel. The specific mechanism is selected at the time of configuration. Dynamic management of the TP buffers is handled by the PDUR.

AUTOSAR-Conformant Configuration

The AUTOSAR BSW is configured with tool support, and the configuration data – e.g. the communication description of the ECU – is saved in an ECU-specific ECU Configuration (ECUC) description file. In the case of a gateway, the routing relationships are also included in this file. Additional MOST-specific attributes are saved in extensions to the ECUC file. Describing the gateway configuration in a single file prevents errors to be introduced by any format conversions or even manual merging of parameters.

Advantages of Implementation by AUTOSAR Architecture

The AUTOSAR architecture makes it possible to develop universal gateways to interconnect CAN, LIN, FlexRay, Ethernet and MOST buses. When a MOST interface is integrated in the AUTOSAR architecture, wrapper layers are unnecessary, and this results in better propagation time and lower memory usage. The presented architecture also has a central routing unit (PDUR and COM), which does not require any additional routing code in the application. This central routing unit enables consistent configuration of all routing relationships with one tool, and the configuration data is saved within a single description file.

Another advantage of the AUTOSAR architecture is the post-build process. This offers a way to change the BSW configuration without re-compiling the application. In this process, the configuration data of the BSW is replaced (overwritten) in the ECU's flash memory. The ECU code is not changed by the post-build process. This provides a way to change the gateway's routing description during development or even in production use.

Outlook

The described implementation of a CAN-MOST gateway with an AUTOSAR architecture focuses on routing messages of constant length, which is the same on the CAN bus. It is conceivable that, in the future, the MOST stack might be extended to include additional functionalities. This includes routing data of variable size – e.g. a list with different numbers of telephone book entries. Similarly, MOST network management tasks could be handled by AUTOSAR modules.

With the Ethernet interfacing available in AUTOSAR, it will be possible to develop very powerful gateways with a MOST connection and thereby utilize the advantages of the AUTOSAR architecture.

Translation of a German publication in ATZ elektronik, 4/2010

Literature references:

- [1] Grzembka, A.: MOST – Das Multimedia-Bussystem für den Einsatz im Automobil (“MOST – The Multimedia Bus System for Use in the Automobile”)

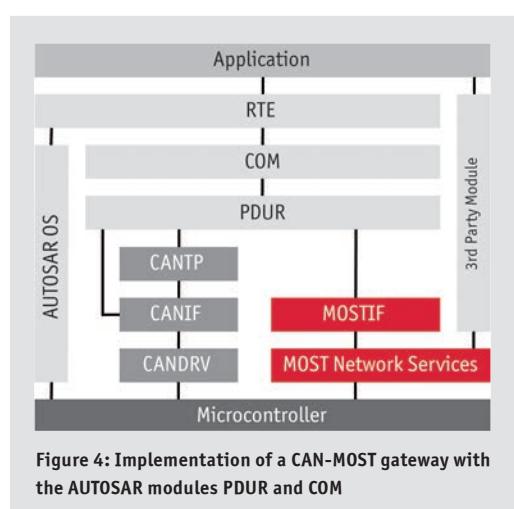


Figure 4: Implementation of a CAN-MOST gateway with the AUTOSAR modules PDUR and COM



Patrick Markl (Dipl.-Ing. (FH))
Senior Software Development Engineer
responsible for concept development in the
Embedded Software product line at Vector
Informatik GmbH in Stuttgart.

Analyzing AUTOSAR ECU Software with XCP

Convenient debugging by extending the AUTOSAR basic software



Many test tools available today support the approach of testing individual ECUs using a remaining-bus simulation. They let users verify required functions in an early stage and independent of the ECU network. This results in very high quality at an early stage of software development. However, testing options are reduced considerably when an ECU is installed in a test bench or in the test vehicle, or when a function is distributed over several ECUs via a network. It is no longer so easy to localize the errors that occur. The causes of this are:

- > An ECU is installed in such a way that it is difficult to access.
- > No more connection terminals are available for debugging.
- > The debuggers for the different ECUs do not condition the data uniformly.
- > The test tools cannot be housed in the vehicle.

Besides physical conditions that restrict debugging in the vehicle, there is another factor as well: some errors only occur sporadically. Determining the causes of sporadic errors is not always easy in the laboratory either.

Requirements for debugging

To find the cause of an error, the values of various software variables are typically checked. It is also important to simultaneously check the variables of multiple software modules in reference to a trigger point. This lets users monitor inter-module consistency conditions. Key examples of this are the "manager modules" whose state machines are interdependent, e.g. the AUTOSAR "Bus State

When debugging in an environment of networked ECUs, debuggers are often of limited use, especially if errors only occur sporadically or in the test vehicle. In such cases, the proven measurement and calibration protocol XCP offers useful services. Described in the following is a method for using XCP to monitor processes in the AUTOSAR basic software (BSW) and in the software components (SWCs). Certain features must be added to the basic software for these measurement purposes. Specialized extensions for the analysis tool allow for efficient debugging and easy evaluation.

Managers" and "Communication Manager" (ComM). Similarly, the test engineer must determine which application requests caused the ECU to enter the measured states. It is also necessary to observe causative chains beyond module boundaries, e.g. to trace a signal path from bus level to application level. Moreover, time stamps are necessary to reconstruct the error history in the analysis.

XCP in AUTOSAR 3.x projects

The AUTOSAR 3.x standard does not specify any mechanisms for remote debugging. For years now, however, the "Universal Measurement and Calibration Protocol" (XCP) has been successfully used to measure and calibrate ECUs in vehicle development, and it covers the requirements cited above very well. XCP provides mechanisms for reading and writing variables via bus systems. Based on DAQ (Data Acquisition) lists, multiple variables can be measured consistently and with a common time stamp. Dynamic DAQ lists make it possible to reconfigure the data records to be read out during the measurement. They offer a way to optimally utilize the available bandwidth of the communication bus.

Typically one or more A2L files are used in conjunction with XCP measurements. An A2L file contains information on the relevant measurement objects in the ECU. Required for each of these objects is information such as memory address, data type, symbolic interpretation and conversion rules. Along with the description of parameters for communication between the XCP Master (e.g. analysis tool) and the XCP Slave (ECU), an A2L file also contains a hierarchically structured representation of the measurement objects. XCP

is a powerful protocol for debugging with its many different ways to access internal ECU variables and convenient configuration via an A2L file.

The XCP protocol is implemented in a dedicated BSW module, which however is not defined in the AUTOSAR 3.x standard. It is linked to the relevant AUTOSAR communication drivers for the bus system being used (CAN, FlexRay, Ethernet, etc.) (**Figure 1**). The XCP driver is seamlessly integrated in the existing BSW configuration chain for convenient configuration of XCP and the remaining modules.

Creating an A2L file

It is a challenge to create the A2L file using the supplemental information noted above. To accomplish this task, the test engineer needs a defined process and suitable tools. Due to the many different configuration options of the AUTOSAR BSW, it is advantageous to generate the A2L file and therefore any changes made to the BSW configuration are accounted for in the A2L file.

The A2L file needed to measure the data flows between software components (SWCs) can be generated with the RTE generator from Vector (**Figure 2**). This enables monitoring of Intra- and Inter-ECU communication and measurement and stimulation of the connected sender/receiver ports. The A2L file is generated based on the RTE configuration which is created with DaVinci Developer. In addition, the RTE generator inserts changes needed for the measurement in the RTE code.

To track the data flow in the modules beneath the RTE as well (**Figure 2**), Vector also offers an A2L generator for measuring

parameters in the basic software. The A2L file is also generated based on the configuration. To make it easier to find the measurement objects, they are stored hierarchically in a directory structure within the A2L file.

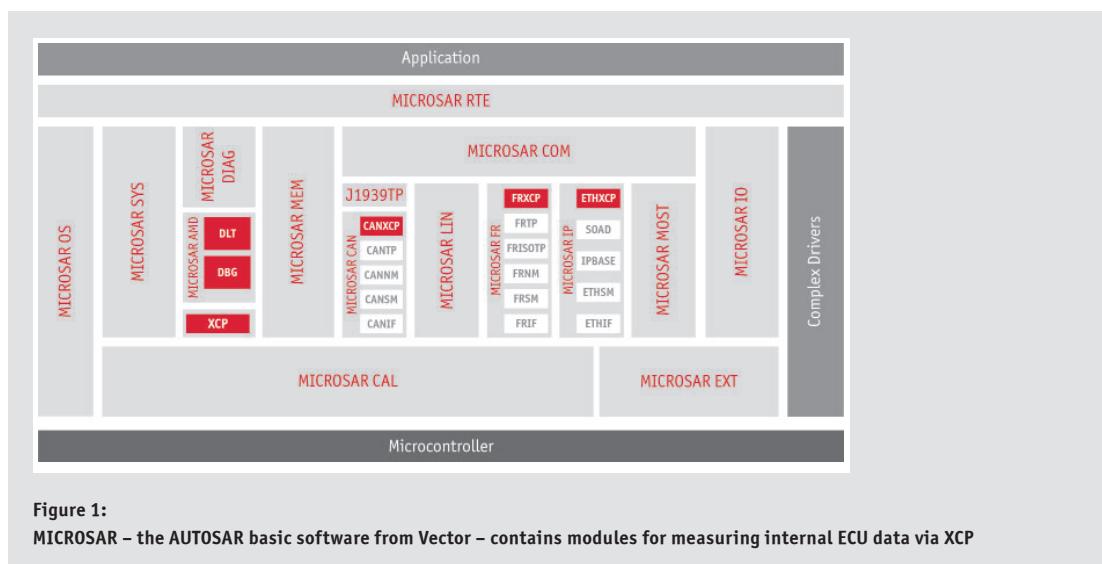
Acquiring measurement data

Often, the test engineer can already localize the modules responsible for the error and define the data for analysis based on the error pattern. First, the XCP Master establishes a connection to the XCP Slave. After the connection has been established, the first data can be read out from the ECU. This is generally information such as the version numbers, version identifiers, etc. Based on this information, the test engineer selects the A2L file matching the ECU's version level and starts the data measurement.

In case of errors that are difficult to reproduce, a quick analysis of the data might not be possible while the measurement is running. In these cases, data loggers with integrated XCP functionality or software tools with an XCP Master store measurement values in the vehicle for longer time periods. In subsequent analysis of the data, XCP Master tools are used such as CANoe – the widely used simulation, analysis and test tool from Vector.

Analyzing measurement data

To test ECUs independent of the communication bus that is used, the XCP Master must support various bus systems such as CAN, FlexRay or Ethernet. Furthermore, in debugging it may be necessary to measure and evaluate the data of multiple ECUs in parallel. Therefore, it is



advantageous to have an analysis tool that is also multi-master capable.

The Option AMD (AUTOSAR Monitoring and Debugging) extends the functionality of CANoe with an XCP Master, in order to dynamically address multiple Slaves. Users can then change the XCP configuration during a running measurement. With Option AMD, the State Tracker (**Figure 3**), Graphic and Data analysis windows are also available for variables measured over XCP. Similarly, the XCP variables can be accessed from the integrated CAPL programming language or from test automation, enabling automated analysis.

The measurement values acquired from the ECU can be analyzed directly during the measurement. The XCP Master must offer suitable mechanisms for this purpose. Therefore CANoe includes definable trigger conditions in the State Tracker Window, e.g. freezing the window contents for quick analysis when a certain measurement value is reached. In addition, CANoe enables complex evaluations of measurement values via a CAPL program.

If saved measurement values are to be evaluated in detail, it is easier for the developer to work with symbolic representation of values rather than with raw values. To allow this, the A2L file must contain assignments of raw values to symbolic names – similar to Enums familiar from the C programming language. CANoe handles the display of symbolic values. This type of representation is excellently suited for observing state machines, whose states are often represented in code using Enums. This method can be applied to any desired variable, e.g. to symbolically indicate whether it is loaded with a substitute value or initialization value.

Debugging in AUTOSAR 4.x projects

The AUTOSAR 4.x standard satisfies the requirement for debugging options in the software by implementing the modules DBG (Debug) and DLT (Diagnostic Log and Trace). These two modules utilize

mechanisms that are similar to those provided by XCP. The main task of the DBG module is to read ECU data, which is buffered inside the ECU and sent to a PC for further evaluation by the developer. Communication with the PC is done by a dedicated communication protocol. Similar to the A2L file for XCP, in the DBG approach a description of the data to be measured is generated. This generation is based on the module descriptions of all BSW modules to be measured.

The DLT module has the task of routing the error messages and warnings generated at runtime to a PC. These messages are generated by the BSW modules DET (Development Error Trace), DEM (Development Event Manager) and by software components of the application. DLT also has its own communication protocol, which is not XCP-compatible. The functions of the two modules DBG and DLT can, however, also be implemented with XCP. Vector already offers this in its AUTOSAR 3.x solution (**Figure 1**).

High-performance access to measurement data

The described mechanisms provide a powerful debugging tool for any ECU – regardless of the AUTOSAR version. The developer can track and analyze processes in the ECU very conveniently using proven XCP tools such as CANoe and CANape from Vector. Use of the VX1000 measurement and calibration hardware is recommended for measuring large quantities of data at rates of up to 5 MByte/s with minimal effects on ECU execution time (**Figure 4**). In this case the ECU is accessed via the debug interfaces JTAG or NEXUS.

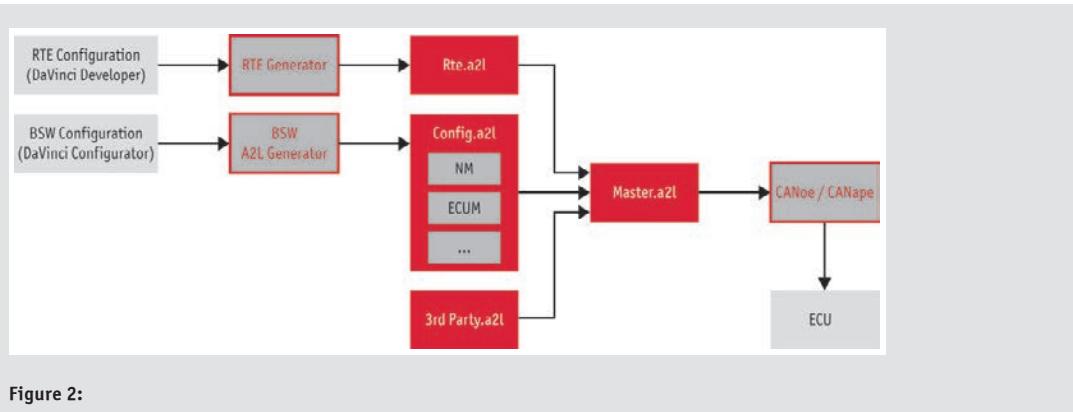


Figure 2:
Generating A2L files based on the ECU configuration

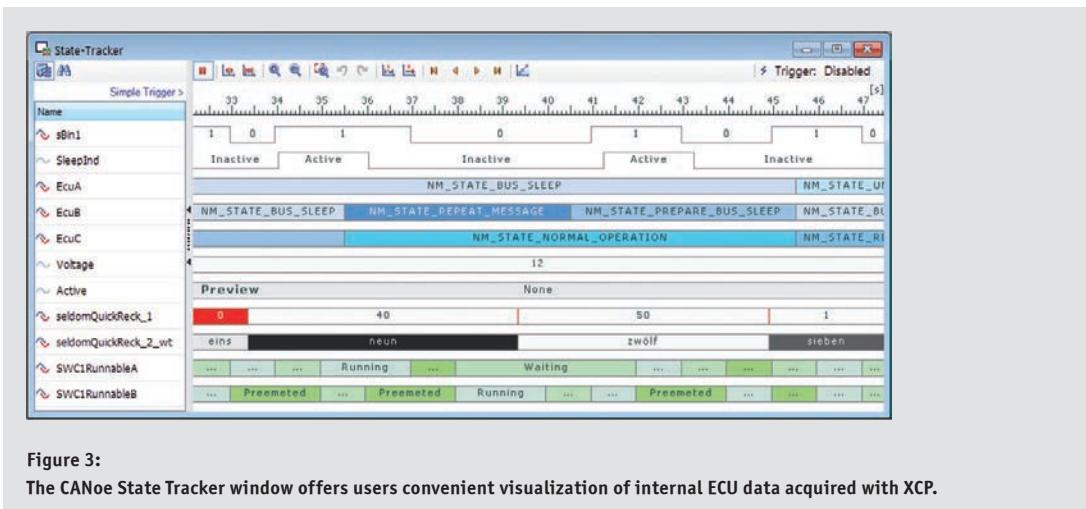


Figure 3:

The CANoe State Tracker window offers users convenient visualization of internal ECU data acquired with XCP.

How does XCP work?

For years now, the Universal Measurement and Calibration Protocol (XCP) has been successfully used to measure and calibrate ECUs in vehicle development. XCP is a global standard that was standardized by the Association for Standardization of Automation and Measuring Systems (ASAM) in 2003. While the CAN Calibration Protocol (CCP) is limited to the CAN bus, XCP enables the exchange of the Transport Layer. It supports buses such as CAN, FlexRay, Ethernet, USB, etc. and is based on a Master/Slave principle. Using a Command Transfer Object (CTO), the XCP Master sends commands to the XCP Slave over the bus; then the XCP Slave responds with a Data Transfer Object (DTO). XCP can be used to read and write values in the ECU's memory.

The XCP Master periodically requests measurement data from the XCP Slave (polling) or has the data sent to it when specific events occur (XCP Events). Which mechanism is used depends on many factors. Measurements with XCP Events utilize bus bandwidth better than with polling. However, it is necessary to modify the ECU code to trigger the XCP Events; this is not necessary with polling. Similarly, XCP Events can already be provided at certain points in the application code during development. Aside from a slight execution time overhead, the XCP code behaves passively. When XCP Events are needed for the measurement, the XCP Master activates them during the measurement. Another advantage of XCP Events is their event-driven output of multiple data with a common time stamp. This lets the developer precisely track certain processes in the software and check whether variables are consistent with one another.

With XCP Events, the test engineer can perform measurements independent of the state of the communication bus. In the case of polling, measurement commands must be received from the

Master, and so bus communication is absolutely necessary. This requirement is omitted with an Event. However, the XCP Slave must provide a buffer that stores the measurement data until communication with the XCP Master has been restored. Due to the severely limited resources within the ECU, continuous buffer storage is understandably impossible. Two buffer strategies are conceivable here. First, there is the linear buffer, which only accepts measurement values until it is full. The other version is a ring buffer, in which the oldest entries are overwritten by new values.

Before a measurement can be executed via XCP Events, the XCP Master must configure the Slave driver suitably. This is done by a series of commands. First, the XCP Master establishes a connection with the XCP Slave, then it transfers the necessary configuration. However, if the measurement should be started immediately after ECU initialization, the measurement configuration must be stored in the ECU's non-volatile memory. This so-called Resume Mode enables measurements immediately after the ECU start.

Published in the special edition of Hanser automotive in November 2011.

All figures: Vector Informatik GmbH

Your Contact

www.vector.com

E-mail contact

info@vector.com



Figure 4:

The VX1000 measurement and calibration hardware from Vector enables acquisition of large quantities of data with minimal effects on ECU execution time.

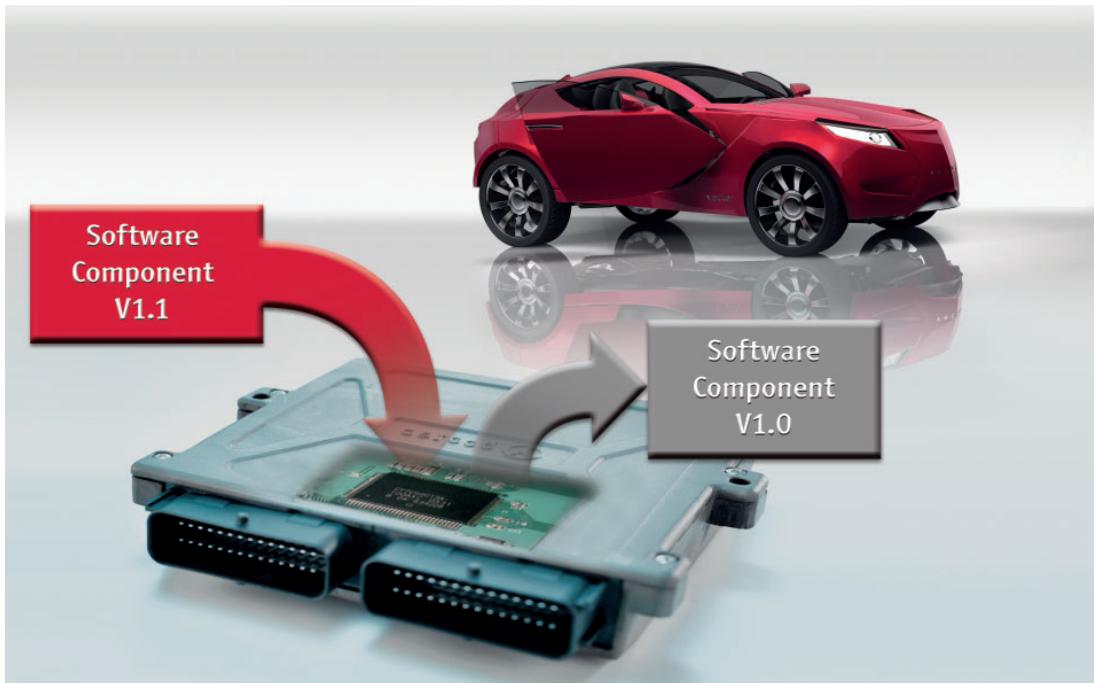


Dipl.-Ing. (FH) Patrick Markl
is teamleader of the Concept Development
team for the Embedded Software product
line at Vector.



Dipl.-Ing. (FH) Stefan Albrecht
is Senior Product Management Engineer for
CANoe/CANalyzer Option AMD.

Plug and Play Solution for AUTOSAR Software Components



The interfaces defined in the AUTOSAR standard enable an easier assembly of the ECU application out of components from different suppliers. However, the process of software development is delayed, because an additional step is required to combine the individual components into the overall software. The use of exchangeable software components reduces the number of these overall integrations, because it is possible to exchange individual software components in flash memory without linking the entire project. This accelerates software development at each of the individual suppliers.

The AUTOSAR ECU software is subdivided into three areas: basic software, runtime environment and application (**Figure 1**). The basic software (BSW) is the basic framework that provides the application with the resources of the microcontroller by means of drivers and abstraction layers. The Runtime Environment (RTE) serves as the interconnecting layer between the application and the basic software. The RTE abstracts the communication between application sections and thereby enables transparent data exchange beyond ECU boundaries as well. For this purpose the RTE provides interfaces to the application and the basic software. In contrast to the BSW, whose static modules can be used unchanged in different ECUs, the RTE is individually generated for each ECU and only supplies the interfaces that are actually needed. The application consists of software components (SWCs), which implement ECU functionality in the form of what are known as Runnables.

Distributed software development

The modular AUTOSAR architecture makes it easier for software suppliers to specialize in specific applications and components, e.g. basic software, vehicle dynamics, engine management, etc. As a result, it is not unusual for an ECU's software to consist of modules and components from different suppliers. The individual parts are supplied as either source code or pre-compiled binary files, and they are combined into complete, executable ECU software at the responsible ECU producer during an overall integration. In turn, all SWC suppliers get the resulting software image for their further development steps.

However, this method is very time-consuming, and it slows down software development. On the one hand, the ECU producer must perform a tedious overall integration for each new version of an SWC. On the other, the SWC supplier must wait for the

integration results to be able to test its application in interaction with the other ECU software. The frequent waiting periods lead to considerable extra effort, particularly in early development phases.

Post-build: software adaptations after linking

AUTOSAR defines a post-build method for the basic software that makes it possible to modify the behavior of certain BSW modules after compiling and linking. This involves storing the configuration data in flash memory, separate from the BSW module code. This means that the data can be replaced at runtime without having to recompile and link the ECU software.

The AUTOSAR standard does not define any comparable post-build method for the SWCs of the application software, and so every modification – even to just one SWC – requires compiling and linking of the entire ECU software. However, there is a way to reduce the number of complete integrations. In the following, we present how an SWC supplier can exchange individual SWCs of the application software directly in flash memory, which reduces integration overhead.

What is the procedure for exchanging SWCs?

When exchanging an SWC, the existing SWC is replaced by an updated version directly in the ECU's memory. This process does not impair the remaining ECU software. This is done either online on the ECU using a flash bootloader, or offline using a suitable HEX editor with subsequent transfer into the ECU (**Figure 2**). SWC exchanging is normally performed by the SWC supplier, but it can also be performed by the overall integrator. By saving in overall integration effort, suppliers can realize shorter development cycles: Coding – Exchanging – Testing – Optimizing (**Figure 3**).

The overall integrator initially generates a software image of the entire ECU and makes it available to the SWC suppliers. Then, the suppliers only need to create a new version of their SWC, place

the binary of it at the correct location in the overall software image, and they can then begin to test their SWC right away. To prevent the SWC versions from drifting apart excessively, the overall integrator generates new ECU software images at regular intervals with updated versions of all SWCs.

There are no limitations with regard to which SWCs may be exchanged. However, the decision about this must be made during the ECU planning phase, because special considerations must be observed in implementation and memory distribution.

Reserving memory for exchangeable SWCs

In today's software development, overall integration typically involves linking the SWC program code without specifying memory mapping parameters. In the case of exchangeable SWCs, the SWC supplier must perform linking based on specific memory mappings. During linking, both the RTE and the SWC expect one another's

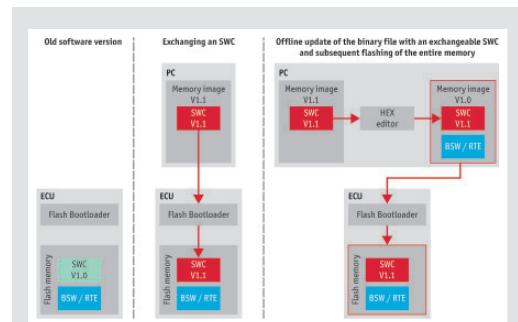


Figure 2:
The exchangeable SWC is released in the memory image (above) and is transferred to the ECU's flash memory.

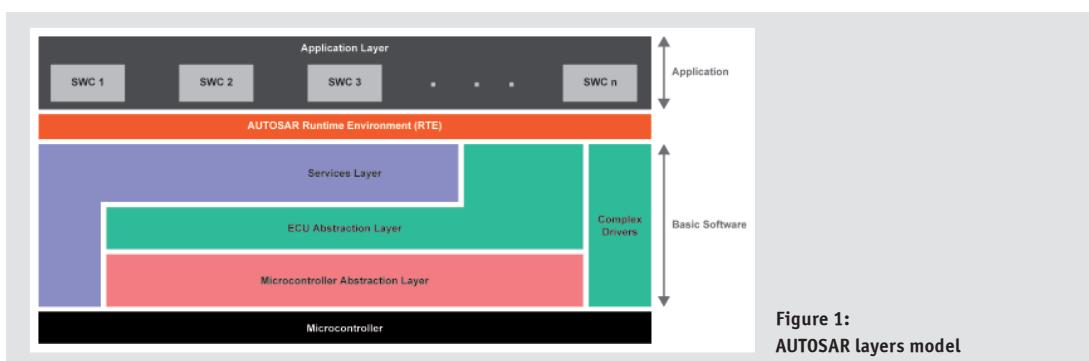


Figure 1:
AUTOSAR layers model

interfaces at the addresses stored in the ECU memory. To assure this, the ECU memory is subdivided into multiple partitions that are exclusively reserved to the individual SWCs. To determine the partition sizes, an estimate is made of how much memory the SWC's individual runnables are expected to require. Initially, this increases the memory requirements for the application. However, this only applies to the development period. Partitioning can be omitted in the final software for production use to reduce memory requirements.

Generally, only complete flash pages can be erased, in the simplest case one or more complete flash pages are reserved for each SWC. However, if the SWC is significantly smaller than one flash page, and the resulting fragmenting is unacceptable, multiple SWCs may be assigned to one flash page. In this case, however, it must be assured that the rest of the flash page content is not changed during the exchange of the SWC with the flash bootloader or HEX editor.

Software environment for compiling and linking an SWC

To compile and link their SWCs, suppliers need a software environment that resolves all SWC dependencies to the called interfaces. Ideally, this environment would be the entire, runnable ECU software – including the BSW and application – which is also used in the overall integration. In its basic methodology, this variant represents a complete integration; however, the integration takes place at the SWC supplier.

The set-up of such a software environment usually takes much effort, and is not always possible. Therefore, the SWC supplier can independently create a minimal environment as an alternative. This environment only needs to service the interfaces called by the SWC (Figure 4).

In both of these variants, after compiling and linking the SWC supplier obtains a binary file which, along with the SWC, also contains the code for the software environment. This additional code is irrelevant for the later exchange of an SWC and is removed. This

results in a binary file of the SWC to be tested, in which the jump targets are already stored at the correct memory addresses.

How does exchanging of SWCs work?

After updating an SWC in flash memory, it must still be possible for the RTE or other runnables to call the runnables contained in the SWC. During compiling in a complete integration, the linker replaces the symbolic jump targets stored in the RTE by the actual (real) addresses of the runnables within the SWC. Because these addresses remain unchanged within the RTE when an SWC is exchanged, it must be guaranteed that the runnables of an exchangeable SWC are always stored at these memory addresses for every compiling and linking operation. This condition can be satisfied by suitable compiler and linker instructions that allocate fixed, manually assigned addresses to the individual runnables, for example. However, it is very likely that the memory requirement of the runnables will grow over the course of ECU development. For this reason, a memory area must be reserved within the SWC. To set up this reserve, the SWC supplier uses compiler instructions to subdivide the SWC into a static area and a changeable area.

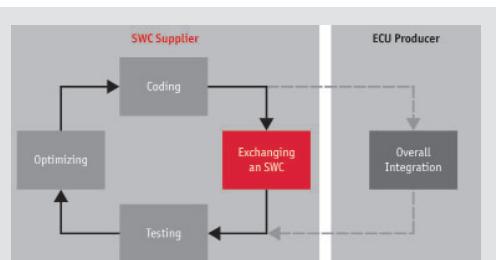
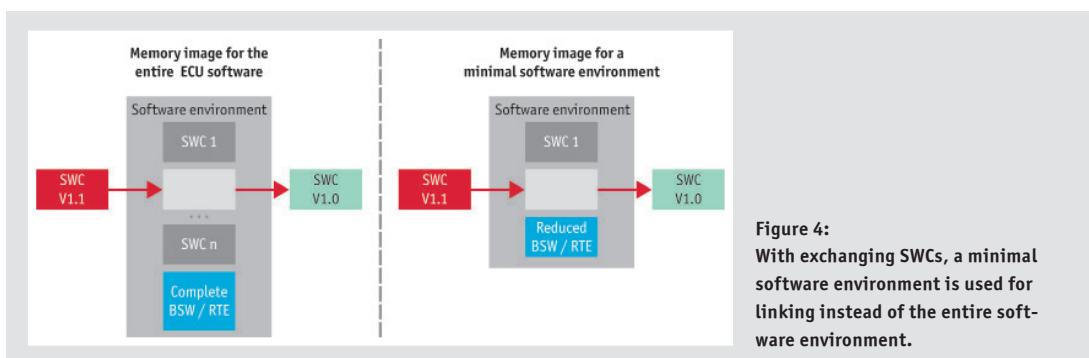


Figure 3:
Exchanging of SWCs accelerates an SWC supplier's software development.

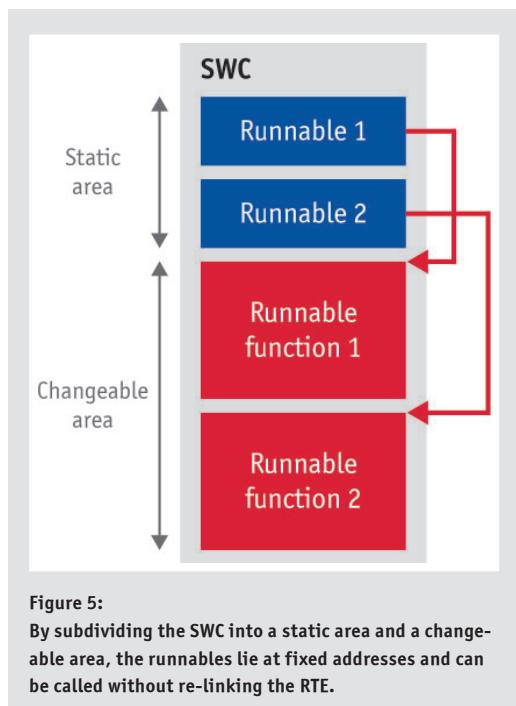


The static area contains the area for jumps into the SWC. This always remains the same, even if the implementation of the functionality changes. The SWC's runnables are modified to achieve this. They no longer contain the specific application code, rather just a function call. The so-called runnable functions that are called here are located in the SWC's changeable area and contain the actual functional code of the runnables (**Figure 5**). As a result, this code may be changed as desired, without shifting the addresses of the runnables in the static area.

Whenever an SWC is compiled and linked, the symbolic jump targets of the runnables are replaced by the actual addresses or offsets. Therefore, the RTE does not need to know the actual addresses of the runnable functions.

Calling port interfaces from an SWC

Analogous to the call of the runnables by the RTE, in the opposite direction the runnable functions must call the port interfaces provided by the RTE. If the SWC supplier uses a complete software environment with BSW and RTE from the complete integration (See section "Software environment for compiling and linking an SWC"), no further measures are required, because all jump targets are known at the linking time.



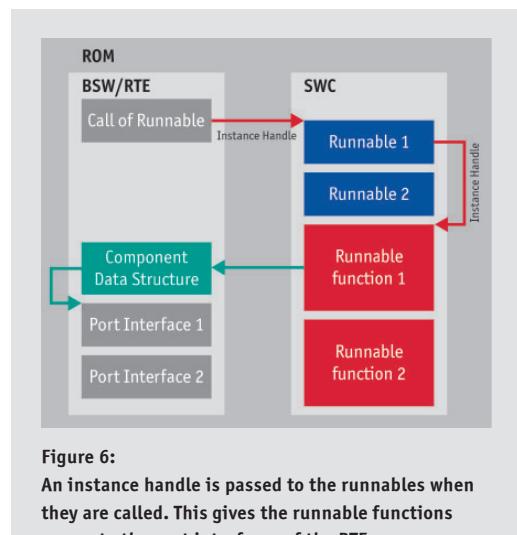
However, if a minimal software environment is used, the port interfaces of the RTE are located at other addresses than those of the overall integration when the SWC is created. This results in incorrect addresses being stored when linking the SWC.

To ensure that calling of port interfaces from the SWCs operates correctly, two AUTOSAR features are used with exchangeable SWCs: Object Code and Multiple Instantiation. With this combination, the RTE passes the so-called Instance Handle as a parameter to each runnable at runtime. This handle points to an indirection structure of the RTE, the so-called Component Data Structure (**Figure 6**). Stored in this structure are the addresses of all port interfaces of the SWC. Therefore, the SWC does not need to know the addresses of the RTE port interfaces. The Component Data Structure is described in the Contract Phase Header file of each SWC.

Constraints for exchanging an SWC

To make it possible to exchange SWCs as described here, certain constraints must be satisfied.

- > During the process of replacing an SWC, port interfaces must not be added or removed. However, additional port interfaces may be allocated in the RTE generation for future extensions.
- > The number of runnables must not change within an SWC, because the RTE can call all runnables known at the time of RTE generation. If runnables are removed during development, or if new ones are added, the RTE must be modified accordingly, and an overall integration must be executed.



- > In the planning phase, it is already necessary to define which requirements the individual SWCs will place on the ECU. This relates to hardware resources, on the one hand, but also to the additional mathematical libraries needed and scheduling of individual runnables.
- > The tool chains of individual suppliers must be coordinated to prevent incompatibilities due to different compiler versions and options.

Outlook

The application example presented here – with multiple SWC suppliers and one overall integrator – is not the only conceivable application case. Exchangeable SWCs can also simplify bypassing of SWCs, because the instrumentation necessary for bypassing can be conveniently applied to the ECU by a exchangeable SWC. If the SWC will then actually be tested on the ECU, the instrumentation can be replaced once again by the non-instrumented SWC.

Exchangeable SWCs represent a powerful tool that can support and accelerate the development of ECU software. So far, preparation of an SWC and memory mapping still needs to be performed manually. However, there are plans to automate much of this process by configuration.

Translation of a German publication in ATZ elektronik, 01/2012

Figures:

Figure 1:

AUTOSAR development cooperation with extensions by Vector Informatik GmbH

Figure 0 & 2-6: Vector Informatik GmbH



Alexander Zeeb, Vector

M.Sc. is Software Development Engineer in Concept Development for Embedded Software at Vector Informatik GmbH in Stuttgart

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.

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

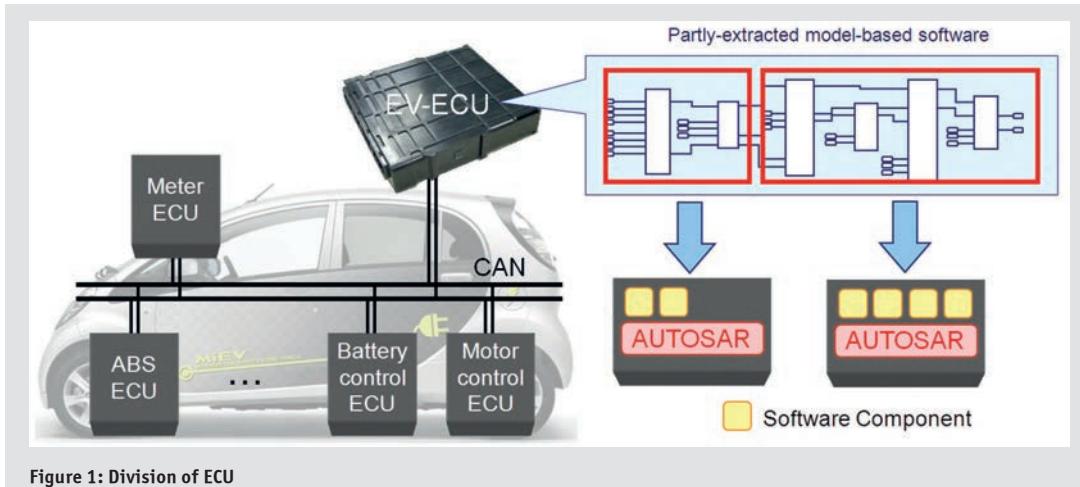


Figure 1: Division of ECU

- > Tool chains
- > Data definition file format

- > ECU software system design
- > ECU software detailed design
- > Coding/Implementation

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

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

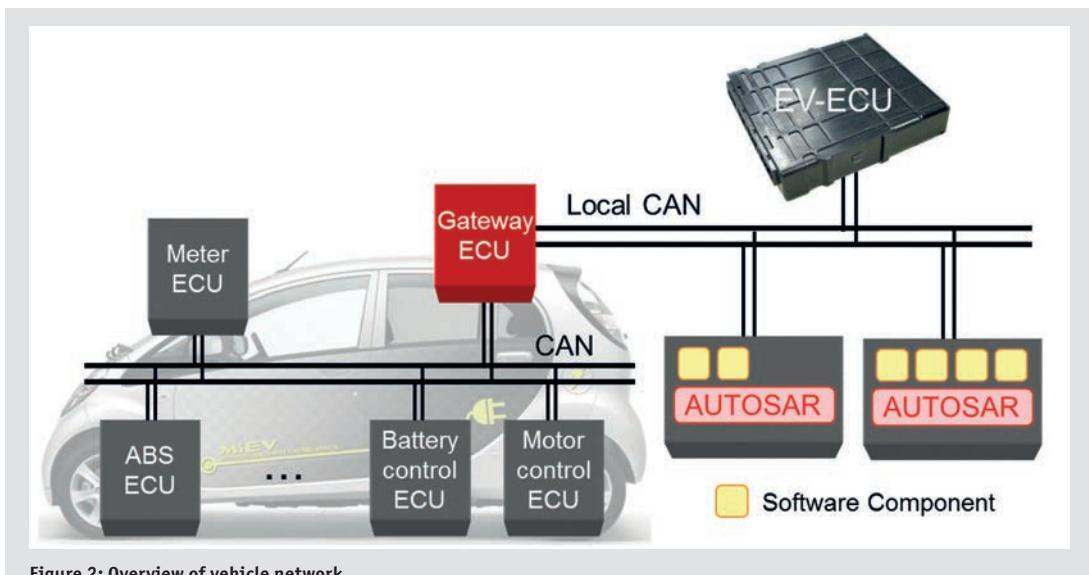


Figure 2: Overview of vehicle network

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

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.

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

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

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.

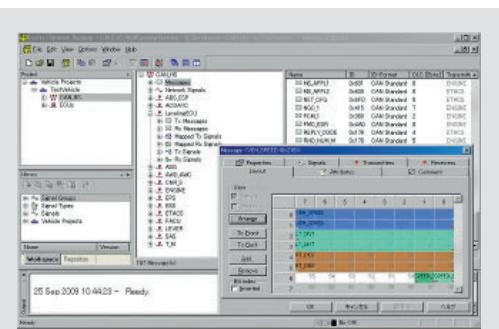


Figure 3: Network Designer™

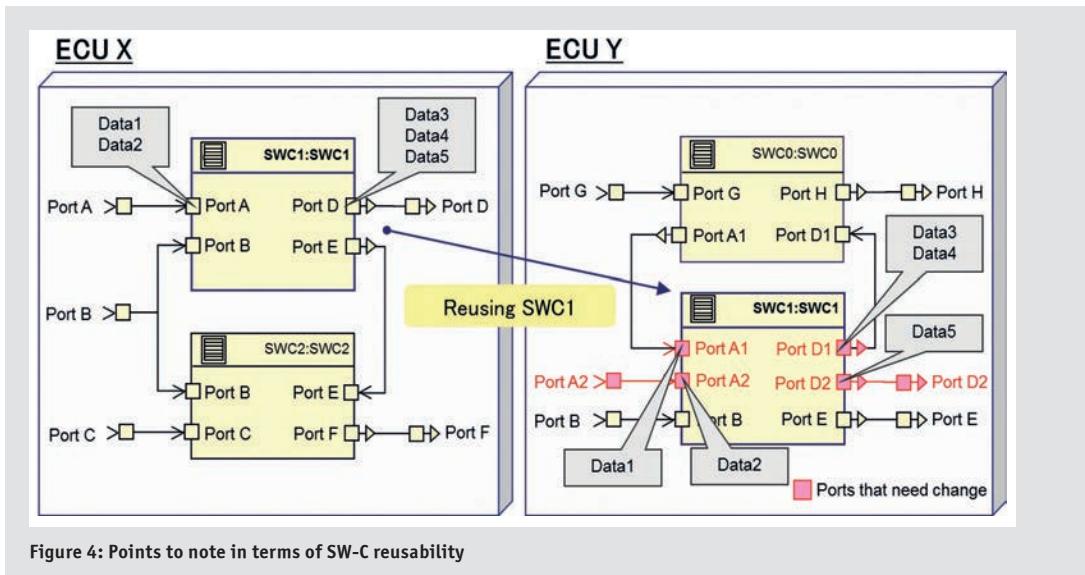


Figure 4: Points to note in terms of SW-C reusability

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.

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 generated above.

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

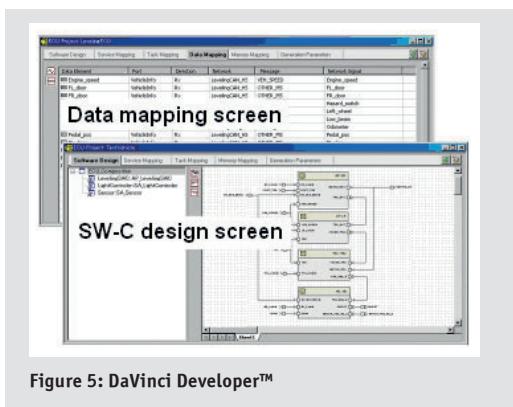


Figure 5: DaVinci Developer™



Figure 6: Vehicle testing

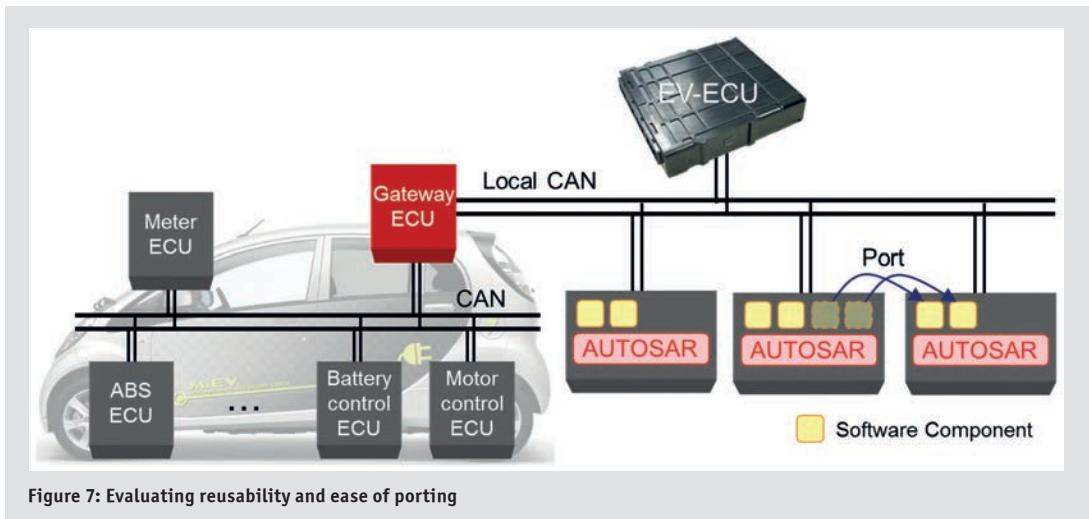


Figure 7: Evaluating reusability and ease of porting

"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 confirm that the car was controlled correctly by the AUTOSAR ECUs.

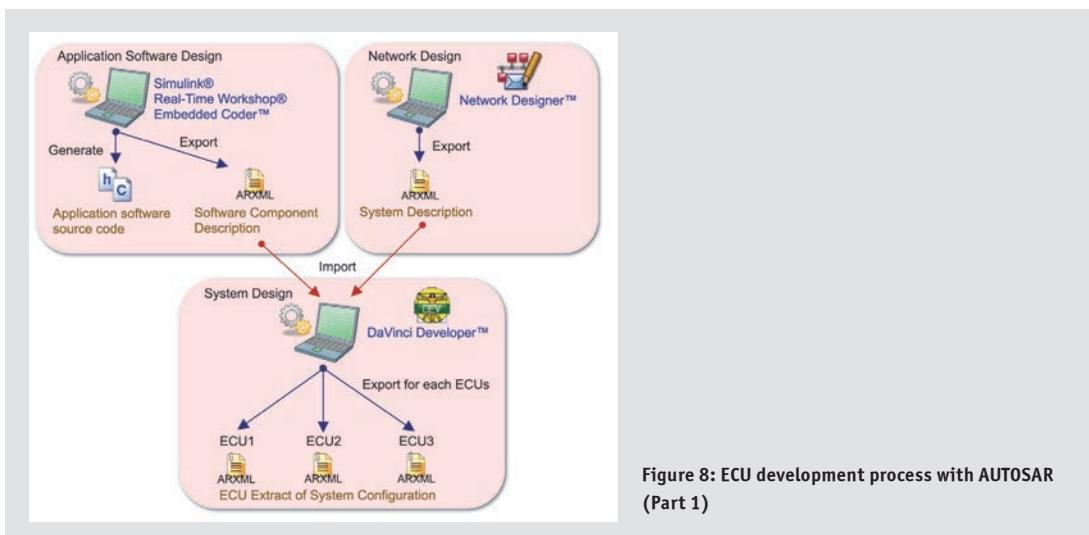


Figure 8: ECU development process with AUTOSAR (Part 1)

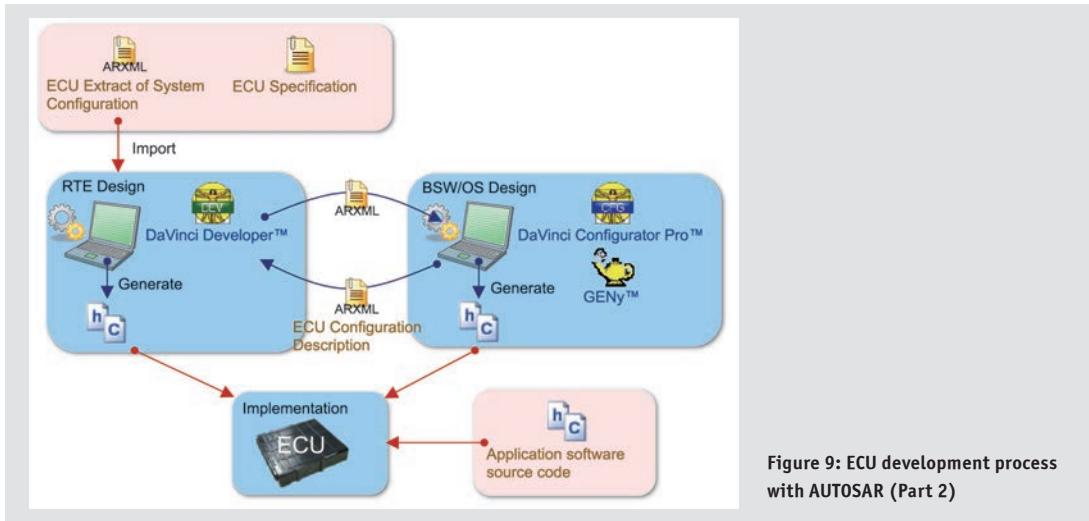


Figure 9: ECU development process with AUTOSAR (Part 2)

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.

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.

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,

³ In the latest generation of the tool chain, Vector provides according functions to compare the design data.

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,
9/2012**

Figures

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



Case Study

Developing a driver library for engine controllers with AUTOSAR Complex Device Drivers (CDD)



The Customer

The FAW Group Cooperation, the “First Automotive Works,” with its headquarters in Changchun, is the largest Chinese manufacturer of diesel engines, passenger cars and medium- to heavy-duty buses and trucks. FAW produces over 2.5 million vehicles annually and is one of the first Chinese OEMs to implement AUTOSAR.

The Challenge

Developing a driver library for engine controllers with AUTOSAR Complex Device Drivers (CDD)

FAW is developing a new generation of its engine controllers and is consistently implementing AUTOSAR basic software in this process. This software will be implemented as one platform which is capable of controlling both gasoline and diesel engines. Since AUTOSAR does not define any suitable drivers for engine controllers, FAW wants to extend their AUTOSAR driver library. They would like to be able to select the necessary sensors and actuators from a toolkit and configure them in the desired quantities and types with the help of a universal tool chain.

The Solution

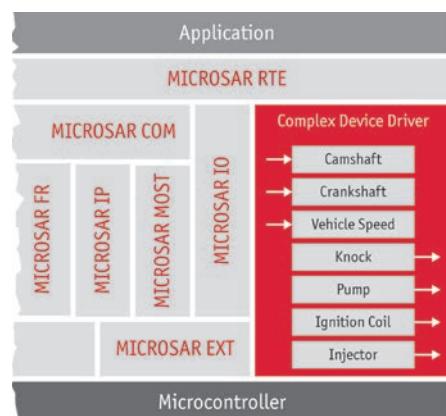
Configuration and code generation of the engine-specific drivers with the existing AUTOSAR tool chain from Vector

Vector implemented the drivers for controlling engine-specific sensors and actuators as what are known as Complex Device Drivers (CDD). To configure the drivers, the relevant Basic Software Module Description (BSWMD) files were generated with the DaVinci Configurator Kit. The code generators were also created with the DaVinci Configurator Kit. DaVinci Configurator Pro reads the BSWMD files and links the associated code generators. This lets it configure the drivers for the engine controller and the AUTOSAR basic software.

The Advantages

A domain-specific driver library that can be extended by FAW

- ▶ All basic software is configured and generated with a single tool: Both the AUTOSAR standard modules and the special drivers for engine control are configured with DaVinci Configurator Pro.
- ▶ The driver library can be extended or modified with the DaVinci Configurator Kit, e.g. when new sensors or actuators are introduced.
- ▶ Reduced administrative effort by having a central department that maintains and extends the library. Different vehicle projects have access to the central library and configure it for their specific engine controllers with DaVinci Configurator Pro.
- ▶ The process and interfaces conform to AUTOSAR specifications: The control algorithms for engine control are still developed with Matlab and Simulink and are implemented as AUTOSAR software components (SWC). They interface with the AUTOSAR basic software and engine-specific drivers via the RTE.





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, ①. 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.

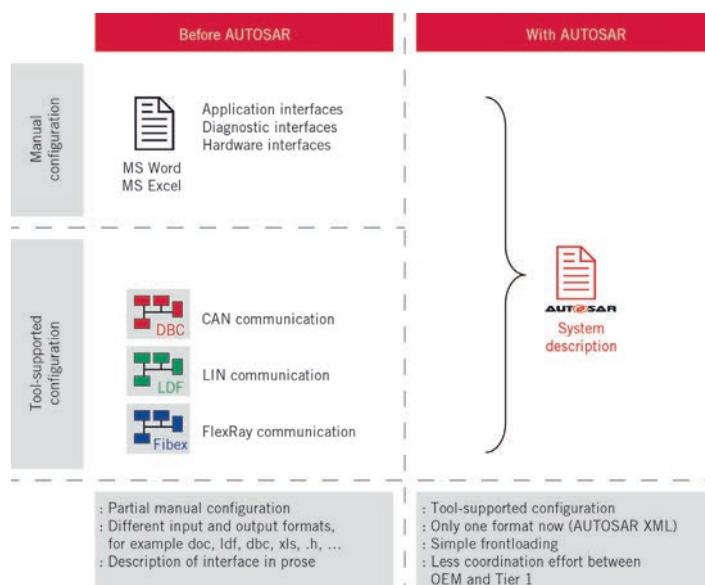
The SWC definition lets the OEM formally describe its own contents in the application software and transfer it to ECU suppliers (Tier 1) in a process-assured way. Daimler recognised this and still sees this as a significant step towards consistent implementation of a system- and function-oriented approach. 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, ②. 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.

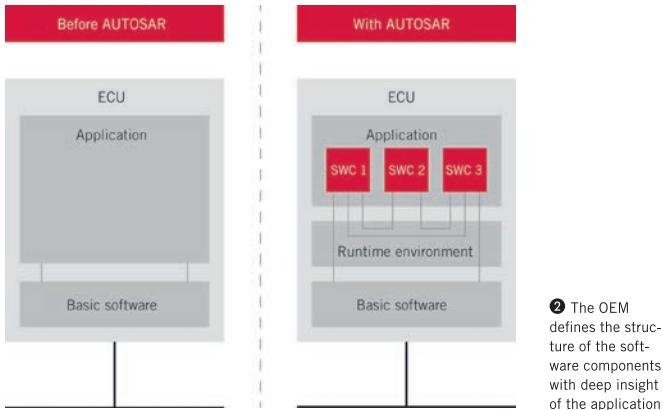
For the Tier 1, early definition means a loss of flexibility at first, since local changes to the interfaces can no longer be performed so simply. These changes must be exchanged between the Tier 1 and OEM, ③. This indicated the need for high-performance difference, merging and report algorithms. Vector 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.



① The AUTOSAR System Description contains descriptions of the communication and other information on the software components and the basic software



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

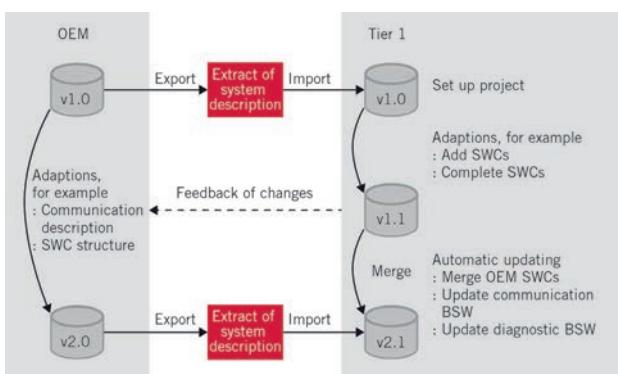
time, in-house tools that Daimler had used and extended for AUTOSAR were adapted to the tools from Vector. In addition, the application cases had to be considered from a Tier 1 perspective to permit an automatic transfer of the created configurations.

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.



- ③ 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

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.

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



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).

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, ①. 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

① The OBD-specific extension of the DCM module contains nine additional services



Case Study

Development of a reference system for AUTOSAR ECUs and system studies

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.

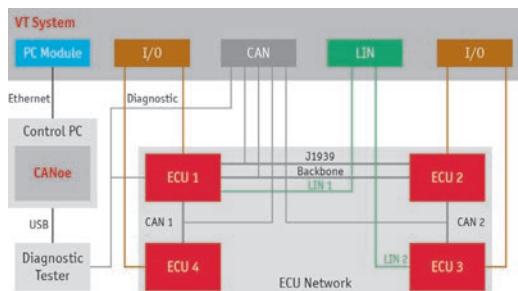
Examples of implemented AUTOSAR functions:

- ▶ 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



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 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.

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 to J1939-81.
- > The "J1939 Diagnostic Communication Manager" for diagnostics according to J1939-73.

The J1939 Request Manager of AUTOSAR 4.1 handles sending and receiving of request and acknowledgement messages. To this

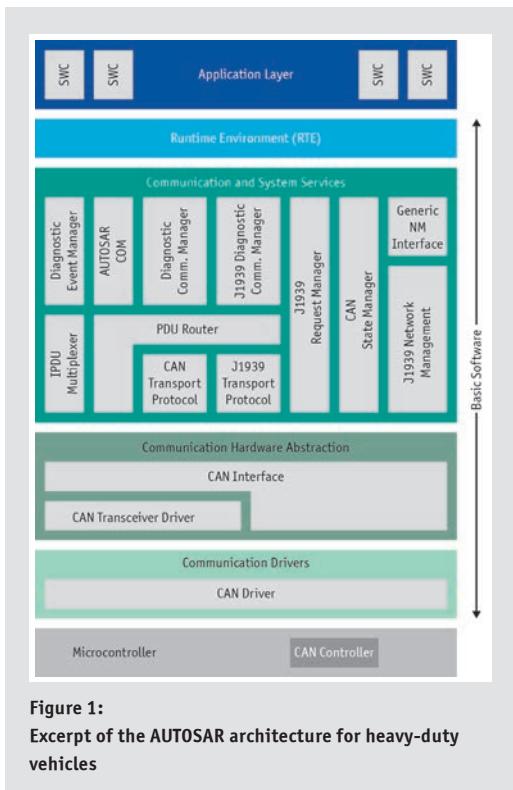


Figure 1:
Excerpt of the AUTOSAR architecture for heavy-duty vehicles

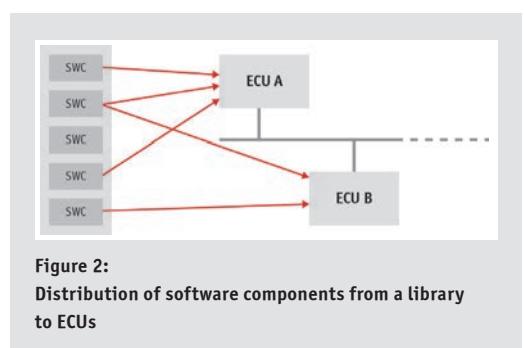


Figure 2:
Distribution of software components from a library to ECUs

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 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.

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

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

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.

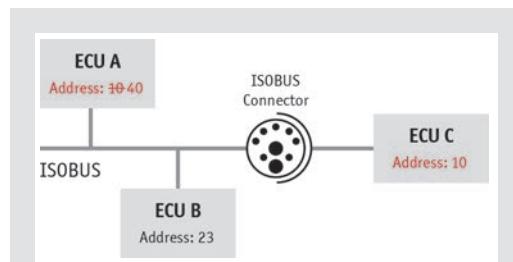


Figure 3:
Dynamic address allocation in ISOBUS: ECU A changes its address, because the just connected ECU C occupies address 10

Translation of a German publication in
Hanser automotive, November 2013

All figures: Vector

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

>> Contact information for all locations of the Vector Group:

www.vector.com

Model-based Functional Safety in E/E System Development



The introduction of the international standard ISO 26262 for Functional Safety of Electrical/Electronic Systems in the Automobile has significantly increased awareness of this topic in the industry. As a result, many OEMs and suppliers are seeking approaches that pragmatically fulfill requirements of the standard, while addressing the rising complexity of safety-related functions appropriately. It takes greater work effort to develop safety-critical systems compared to conventional systems. Although constraints remain the same, additional activities related to schedules, resources and costs are unavoidable. Existing development, analysis and test methods and their associated tools are often fragmented and can only be integrated in a uniform process with great effort.

Consequently, new approaches are needed to enable functional safety as an integral component in the development of E/E systems. It is important to consider all levels of system designs (**Figure 1**) and assure that safety goals of the systems are verifiably implemented according to the standard.

Comprehensive perspective of system architectures

A key requirement of all current safety standards in automotive and non-automotive industries (e.g. IEC 61511 for the process industry, IEC61513 for nuclear power plants, EN 50128 for railway systems) is that it must be verified that the developed system concept fulfills system safety goals. Safety goals are typically identified in hazard and risk analyses on the functional system level. The functional

and technical safety requirements derived from the safety goals are then allocated to system components. Correct implementation of these safety requirements must be assured by a suitable combination of reviews, analyses and tests.

The attainment of system safety goals depends on many different factors. One example: faulty programming of software functions or random hardware failures in critical components. As recommended in ISO 26262, such isolated failures can be avoided relatively easily, or can at least be detected and overcome by current development methods. It becomes more problematic when safety goals are affected by a combination of different system factors on different architecture levels. However, in the case of complex systems, such interdependencies can hardly be revealed by conventional, document-based design methods. Here are two

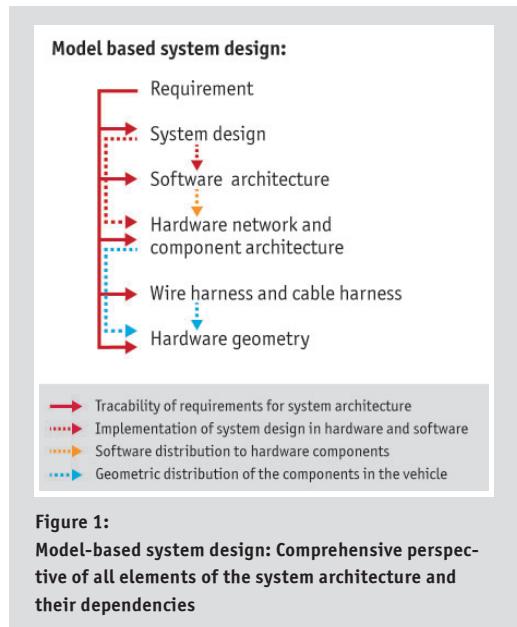


Figure 1:

Model-based system design: Comprehensive perspective of all elements of the system architecture and their dependencies

examples of typical design errors:

- > During software development, incorrect assumptions are made about the integrity of the communication medium between two functions. As a result, communication failures are inadequately considered. For example, later redistribution of software functions to different ECUs in different bus segments might violate system integrity due to the flawed original assumption – without the software developer being aware of it.
- > A hardware component exhibits atypically high failure rates, because it is located in a vehicle area that is exposed to extreme environmental conditions, such as temperature mechanical vibration or electromagnetic interference. Elevated failure rates were not considered in the safety concept.

A comprehensive approach to system modeling is necessary to give due consideration to such complex interrelationships in the design phase. This approach combines all functional and nonfunctional requirements in one data model as well as the logical system design, network architecture and software and hardware component architectures. Similarly, it is necessary to consider the wiring harness and the topological distribution of components and the wiring harness in the vehicle (Figure 1). In this process, it is important to be able to describe the interdependencies between different architecture levels as well as domain-specific attributes of the modeled artifacts (e.g. bus latency times, hardware failure rates or temperature ranges of installation location). Based on this data, analysis – which may be performed semi-automatically – can

then be used to formulate and answer questions such as: "Will wires transmitting signals relevant to the safety concept be routed in areas that are at risk of damage in a crash?"

Model-based safety analyses

ISO 26262 requires that safety analyses be performed on the system, software and hardware levels, so that design weaknesses can be detected and actions can be taken to improve the design. One such analytical method is Failure Modes and Effects Analysis (FMEA). It is used to identify potential failures in individual components and analyze their effects on system goals as well as their probability of occurrence. This type of analysis is well suited to identifying components that are critical from the perspective of safety goals. Nonetheless, the results depend very much on the quality and scope of the modeled system design upon which the analysis is based. When system dependencies are inadequately documented, the results often depend on the individual experience levels and knowledge of engineers, and this often leads to the problems illustrated in the examples above.

Comprehensive modeling of the system is an optimal precondition for performing safety analyses. In conducting these analyses, the safety expert accesses information directly from the model. Dependencies, especially between different architecture levels, may be automatically analyzed, and their effects on system safety evaluated. Actions derived from the analysis are implemented directly in the model, and they are linked to the relevant analysis. This produces traceability between the design, analysis and affected safety goals, which are absolutely essential to provide the required safety verification and conduct impact analyses in response to system changes. Close intermeshing of the system design, safety analysis and design of the safety concept makes it possible to immediately analyze and evaluate the effects of model changes. This eliminates the need for time-consuming re-designs, which would otherwise be unavoidable in the case of sequential handling of the design and then the safety analysis. In addition, multiple implementation variants of a system may be compared and evaluated. This makes it possible to perform system optimization that incorporates safety-technical perspectives in an early phase of system development (Figure 2).

Product line approach yields increased efficiency

From a system viewpoint, personalizing vehicles, i.e. offering different variants and configurations, leads to a number of variants that is simply unmanageable. The safety engineer is immediately faced with the challenge of providing the required safety validation for each system variant. One way to confront this challenge is to use a product line approach. This approach is based on the analysis (domain analysis) of variant-based systems (product lines) with

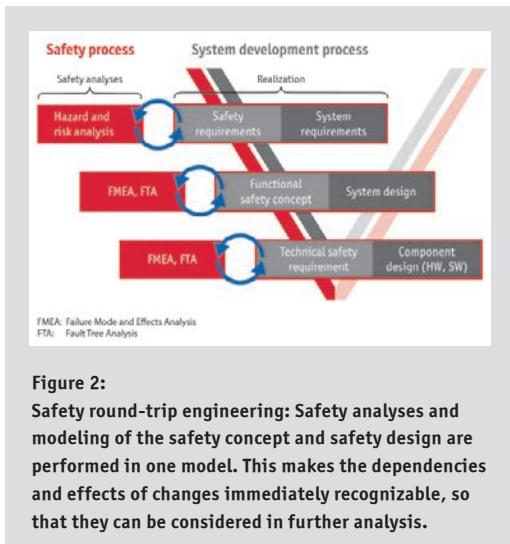


Figure 2:
Safety round-trip engineering: Safety analyses and modeling of the safety concept and safety design are performed in one model. This makes the dependencies and effects of changes immediately recognizable, so that they can be considered in further analysis.

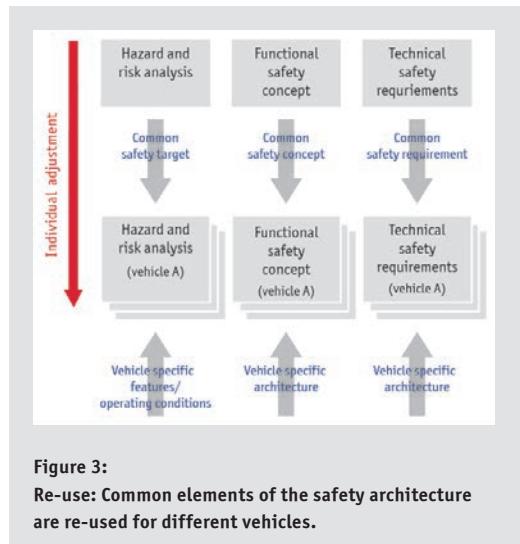


Figure 3:
Re-use: Common elements of the safety architecture are re-used for different vehicles.

regard to identical parts (commonalities) and differences (variations). Identifying commonalities provides a foundation for re-using them over a number of products (vehicles). Furthermore, the work only needs to be performed once to create requirements analyses, implementations or tests.

To make the reusability concept feasible for the development of safety-critical systems, the product line approach must be extended to hazard and risk analyses as well as to safety concepts. These analyses and concepts are available for re-use, provided that they relate to a common set of features. Technical safety requirements for specific vehicle projects are derived from the functional safety concept. For example, specific vehicles are characterized by specific operating states, system architectures, etc. (**Figure 3**). Applying domain analysis to the technical safety requirements can make some of these requirements available for re-use.

To maximize the added value of re-use, domain analysis can also be extended to system components (software components, hardware components). Performance of the safety validation is supported by re-use of a portion of the argumentation structure (safety case argumentation structure). Assumptions about the specific vehicle environment are formulated as requirements whose implementation or validity must then be analyzed for the specific context.

A model-based approach is a prerequisite for performing variant-specific analyses and consistency checks. This approach ensures, for example, that all actions listed in the safety validation and technical requirements derived from the re-used safety concept can actually be implemented in the specific vehicle. If a redundancy that is provided in the safety concept were to be lack-

ing, for example, this would be detected as an unfulfilled requirement in a variant-specific consistency check.

Summary

Model-based methods and a comprehensive approach (Systems Engineering) give the system developer a method for understanding and mastering complex systems, which in the end leads to safe systems. The model-based approach promotes the increased efficiency to compensate for the additional work effort caused by safety relevant activities (analyses, implementation of actions resulting from them and safety validations). The re-use of entire portions of the safety architecture avoids work steps that are repeated one or more times, and this leads to further increases in efficiency.

The all decisive factor for success is the availability of proven tool support for implementing the paradigm shift described here in daily practice: from a document-centered approach to a model-based approach. A key requirement for a tool is its suitability for describing complex systems by means of a related semantic data model. But domain-specific graphic notations and support in analyses as well as performance of safety validation are also absolutely essential. Collaboration on large teams results in further requirements. The provision of a collaboration platform for data storage (single source principle) and data exchange among all project participants. In this process, competencies and responsibilities must lend themselves to modeling by a rights and roles concept. Chronological traceability (history of versions) requires integrated version and configuration management.

In its PREEvision product, the company Vector is offering a proven tool for model-based systems engineering that has been on the market for over five years. PREEvision also provides assistance in the areas of testing (test data management), planning (product and release management) and change management. In its latest version, PREEvision supports an ISO 26262 conformant approach to modeling, analysis and testing of functional safety concepts.

Translation of a German publication in 'Automobil Elektronik', issue 2/2012

Figures:
Vector Group

Links:

Home page Vector: www.vector.com

Solutions for Functional Safety: www.vector.com/functional-safety

Product information PREEvision: www.vector.com/preevision_en

Authors:



Dr. Simon Burton
studied Computer Science at the University of York in England. He earned his doctorate degree in York as well with a thesis on Specification and Testing of Safety-Critical Systems. Focal points of his career include the introduction of systems engineering and model-based approaches in developing embedded, safety-critical systems. In 2006, he joined the Vector Group where he has served in the areas of safety consulting, product management and business management of the Automotive Solution.

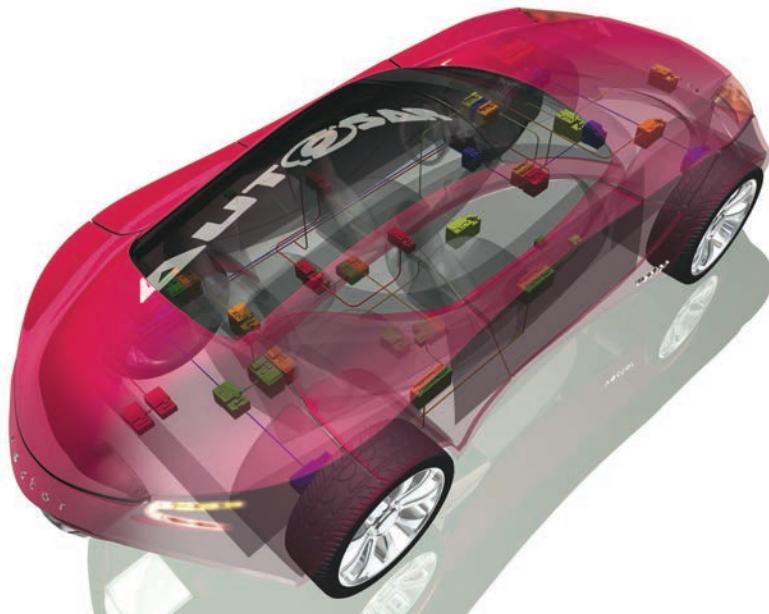


Albert Habermann
studied physics at the Ludwig-Maximilians University in Munich. Focal points of his career include the introduction of model-based methods to software development and testing. In 2006, he joined Vector where he is employed as a project manager in the Customer Service area for the PREEvision architecture and systems engineering tool.

Development of Safety-relevant ECU Software

Vector and TTTech to Partner

In the future, Vector Informatik (Stuttgart) and TTTech Automotive (Vienna) will work together to develop standard software modules for ECUs. While Vector brings its competence in AUTOSAR basic software to the joint venture, TTTech will focus on safety electronics based on ISO 26262 and FlexRay.



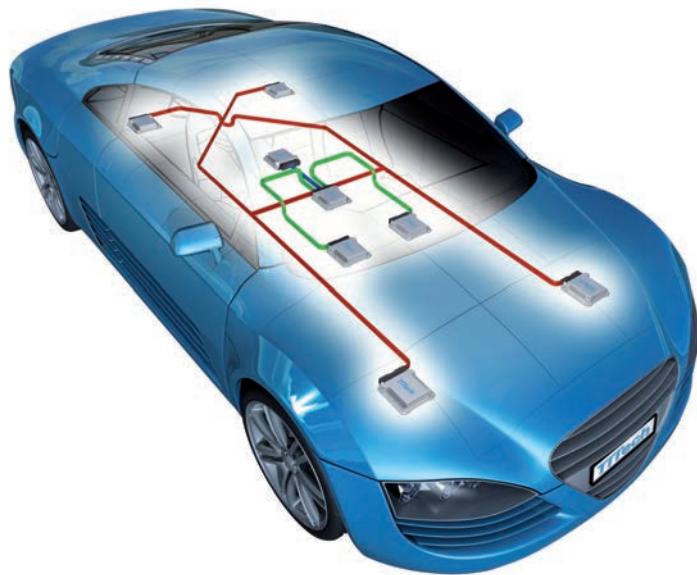
What was the motivation for this joint venture?

Dr. Helmut Schelling: Since 1992, we have been developing basic software for a large number of automotive OEMs. The effort required to supply mature and complete basic software has increased once again with AUTOSAR. Not only do we need to master AUTOSAR's large range of functional features and high complexity; we also need to support several AUTOSAR releases in parallel. Furthermore, in nearly all cases it is necessary to make OEM-specific modifications or additions. The market is also still very limited, so we think it makes sense to forge partnerships wherever the specific strengths of our strategic partners complement our own strengths.

The idea for the joint venture with TTTech came from our customers, who indicated that they would have an increasing need for safety-related ECU solutions over the near future. We already cover the entire bandwidth of basic software modules. Now, we can utilize TTTech's experience to extend them to the area of safety-related systems. So these are ideal conditions for a joint venture.

Dr. Stefan Poledna: TTTech's roots lie in the development of time-triggered communication. At the same time, the topic of safety was our key focus right from the start. Within the Automotive area we have successfully launched the FlexRay protocol and AUTOSAR into production of the new A8 together with our lead customer Audi. TTTech solutions can also be found in other areas with safety-relevant applications on such products as the new Boeing 787 and the Airbus A380.

In response to the growing acceptance of AUTOSAR among customers, it became necessary to obtain all AUTOSAR basic software from a single source. AUTOSAR solutions that can be used in safety-relevant applications to fulfill the upcoming ISO 26262 standard are also in demand. So, we were faced with the decision of whether to come up with a complete set of products or focus on our core competencies of safety and networking. After intensive deliberations, we decided on a partnership solution. In this area, we think that our solutions will offer an ideal supplement to AUTOSAR products from Vector.



What specific goals have you set for your collaborative work?

Schelling: The market requires AUTOSAR basic software to work seamlessly with safety-related ECUs. The goal of the joint venture is to now utilize our existing synergies to offer a solution that is powerful and simultaneously cost-effective. On the one hand, AUTOSAR requirements must be fulfilled, e.g. with regard to implementing an E2E Library to guarantee communication protection. On the other hand, we would like to provide certain functions needed in practice, such as algorithms for checking a controller's "health status."

Poledna: This solution will be available later this year, starting with implementation of end-to-end communication protection requirements per AUTOSAR 4.0. In addition, basic software products will be offered that enable the use of AUTOSAR in ASIL-D conformant ECUs. This solution also anticipates future issues such as memory and timing protection and program flow monitoring, which are not yet specified in AUTOSAR.

What benefits will customers have from this collaboration?

Schelling: The customer gets an integrated solution, in which the many years of experience of both TTTech and Vector complement one another ideally. For our customers, this collaboration means that component functionalities are perfectly in tune with one another - how safety components interact with the operating

system, for example -, and it means that these safety components can be configured with proven Vector tools. By bundling development capacities we will be able to offer a complete system for many hardware platforms very quickly.

Poledna: As a result, the Vector/TTTech solution will give customers an efficient way to use AUTOSAR-based ECUs over a full range of applications. A special highlight of the joint development is the ability to run safety-relevant functions together with non-safety-relevant functions in one ECU. This lets customers realize enormous cost benefits and time-to-market advantages. Only the small share of safety-relevant software needs to be developed to safety requirements; all other functions can be re-used or created in usual development processes. This eliminates high certification and integration overhead.

To what extent do AUTOSAR and ISO 26262 currently fit together, and what issues still need to be addressed?

Poledna: Version 4.0 of AUTOSAR addresses the topic of safety and defines safety mechanisms that simplify ISO 26262 certification for many ECUs. In this process, project-specific developments are replaced by standardized components. For example, AUTOSAR 4.0 specifies the aspect of end-to-end communication protection. Other concepts such as memory and timing protection and program flow monitoring are being evaluated in Phase III of AUTOSAR that is currently starting, and they will flow into upcoming releases.

TTTech has been actively involved in defining communication protection and will continue to assist in the standardization process.

AUTOSAR does not specify any requirements for the process needed to develop safety-relevant components, so ISO 26262 is relevant here. These two standards complement one another ideally.

TTTech can look back upon many years of experience on certification projects. It has already developed complete ECUs including software per the generic standard of ISO-26262, which is IEC 61508. This experience has also been applied to implementing end-to-end communication protection in the automotive industry, where it has already been successfully validated by TÜV (German certification authority).

What is your assessment of market acceptance for AUTOSAR and ISO 26262?

Schelling: All of our discussions in recent years show a clear trend in AUTOSAR: increasing numbers of automotive OEMs and suppliers are relying on this standard. Even though the initial work involved is quite high, over the long run this standardization can be expected to deliver substantial cost savings.

Poledna: That is also our experience; a standardized process in the area of standard software and technical safety processes will greatly benefit the industry. This has been shown in other industries, where standardization has resulted in optimized quality, safety and costs. A similar trend can be expected with ISO 26262, although compared to AUTOSAR we are clearly still in a significantly earlier phase.



Dr. Helmut Schelling, Managing Director of Vector Informatik GmbH

Schelling: The standard creates the assurance that new safety-related systems can be developed in state-of-the-art processes. On the other hand, solutions exist today that were not developed based on ISO 26262 yet still satisfy the most stringent safety requirements. They will certainly remain in use for some time.

Poledna: We assume that the ISO 26262 standard will define the state-of-the-art in safety-relevant automotive applications. The standard provides a set of guidelines for developing safety-relevant systems, and it serves as the foundation upon which TTTech and Vector will be offering their combined product line. The goal here is to help our customers reach their goals more quickly with high-quality, high-performance software – in both safety-relevant and non-safety-relevant applications.

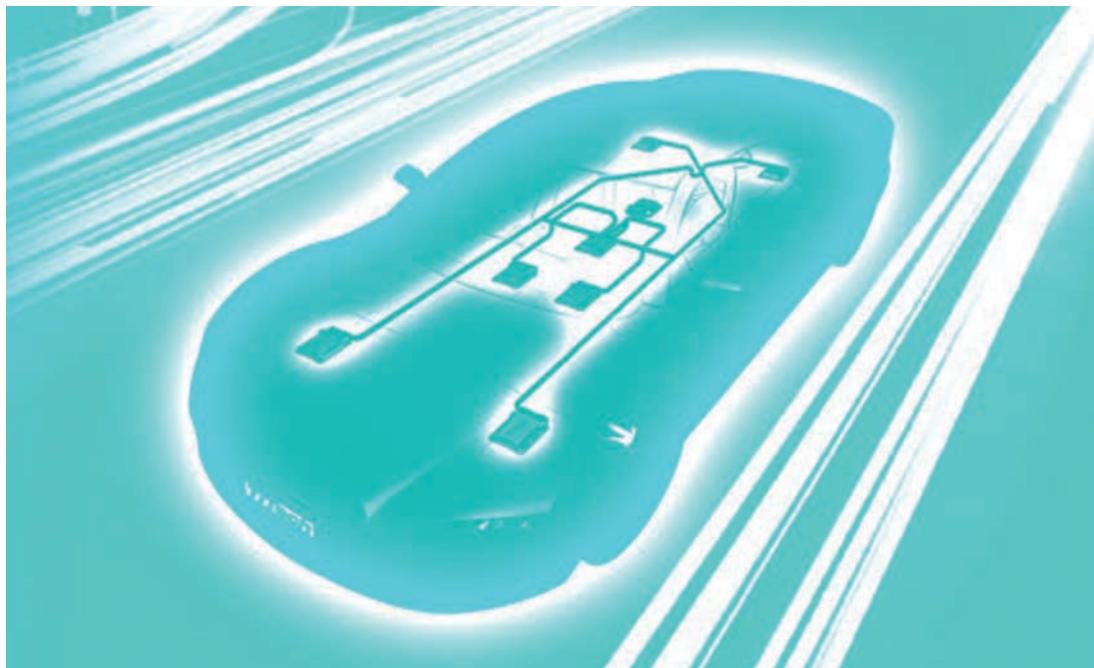
According to the interview of Stephan Janouch, published in the German magazine Elektronik automotive 1-2/2010.



Dr. Stefan Poledna, CEO of TTTech Computertechnik AG

Recipe for Safe Software

Development of ECU basic software according to ISO/DIS 26262



With the introduction of the new ISO 26262 standard, requirements for safety-related functions are becoming much more challenging than they used to be. At the same time, they are more precisely and clearly defined. Formally validated systems and the re-use of existing solutions do not need to contradict one another here. Generic safety modules in hardware and software can supplement proven components.

ISO 26262 is currently a draft standard in the voting phase but is expected to become a binding safety standard in the year 2011. It defines functional safety for electric/electronic (E/E) systems in an applicable manner. The future standard is based on the IEC 61508 standard that is used in a variety of industries, and it also addresses the fact that safety-related and non-safety-related functions are often interrelated in the automotive field, and therefore cannot always be clearly isolated from one another. The goal of ISO 26262 is to create a better understanding of safety-related functions and narrow their range of interpretation as much as possible.

One challenge in engineering development is to assess all potential hazards and risks in advance and take suitable measures to minimize these risks. To facilitate this, ISO prescribes that a "hazard and risk analysis" has to be performed at the beginning of development. All operating states and their associated failure types are analyzed here within the system under consideration, and various potentially hazardous situations are identified. Afterwards,

each hazard is assigned a safety requirement level (Automotive Safety Integrity Level, ASIL) from A to D, where A is the lowest and D the highest safety level.

As the ASIL increases, requirements increase for hardware metrics (FIT rates, test coverage, etc.) of the system as well as for the software development process (testing depth, requirements tracking, review documentation, etc.). The system supplier must fulfill these heightened requirements in addition to the high quality requirements which already exist.

Mastering rising complexity with standardization and re-use

In recent years, not only have the various in-vehicle functions grown in number and complexity; they are also more often being distributed throughout the vehicle. On the one hand, this trend was enabled by significant growth in computing power of the

processors used, and on the other by the larger bandwidth available in networking.

Meanwhile, the implemented functions have attained such complexity that conventional development methods can no longer meet the requirements of this architecture. This situation led OEMs to join together in the AUTOSAR development partnership in 2003; their common goal is to define a uniform software architecture for ECUs and decouple the hardware from the software.

Before AUTOSAR was defined, the communication stack and diagnostics were primarily the focus of automotive OEMs. With AUTOSAR, the basic software has undergone significant expansion. While it previously covered such areas as CAN, LIN, FlexRay and diagnostics, basic software now also includes the operating system, watchdog, memory stack and driver layer for the microcontroller.

AUTOSAR basic software attained a very high level of complexity when version 4.0 was released at the end of 2009. Over 80 software modules are defined via system description files (AUTOSAR System Configuration Description); they are also configured with powerful tools and their code is generated.

Requirements of AUTOSAR basic software used in safety-related ECUs

Aside from functional requirements, one key requirement remains: the basic software must not "disturb" the safety-related software. The meaning is two-fold: for one, the basic software must not violate the memory of the safety-related software, and for another, the basic software must not require more execution time than originally intended. Together, these conditions are also known as "freedom from interference".

The seemingly obvious approach is to develop the entire basic software based on standards for safety-related software grouped under ISO 26262. However, as already mentioned, the AUTOSAR basic software is very complex, has an enormous functional scope,

and its specifications are subject to short revision cycles. This would make development based on ISO 26262 very extensive.

Another approach is to reliably separate the basic software from the safety-related software without interference by implementing a protection layer (software partitioning). AUTOSAR already contains the functionality needed for this: memory protection and program flow monitoring. This functionality makes it possible to perform an ASIL decomposition (see following section) of the basic software according to QM and the protection layer of the desired ASIL according to ISO 26262. Therefore, it is sufficient to just develop the modules for memory protection and runtime behavior (program flow monitoring with watchdog) according to ISO 26262.

Use of ASIL decomposition according to ISO 26262

As already mentioned, ISO 26262 does not absolutely require re-development of all software mechanisms. To reach a certain ASIL, which is necessary to reach a defined safety goal, a combination of independent elements and suitable partitioning of safety requirements into redundant safety requirements can be used. For example, it is possible to attain ASIL C by combining two components with ASIL B and ASIL A while addressing specific requirements (**Figure 1**). ASIL decomposition can be performed on the system, hardware and software levels. In performing decomposition, care should be taken to ensure that the hardware architecture metrics remain unaffected and that there is no likelihood of violating the safety goal. In addition, ISO 26262 requires mandatory confirmation reviews that are based on the safety goals of the original ASIL (in our example: ASIL C).

To guarantee the necessary independence of the software components of different ASILs in an ECU, the standard defines rules for maintaining and verifying freedom from interference. These rules have to be kept and verified. That is because – in the special case of safety-related and non-safety-related elements – "coexistence" of the elements requires suitable measures and their verification.

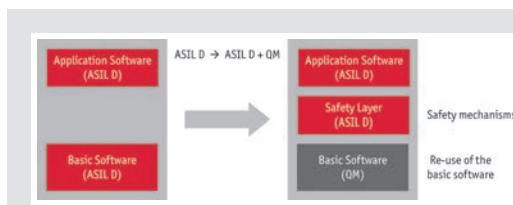


Figure 1: ASIL decomposition: A safety layer is added to the basic software to reach ASIL D.

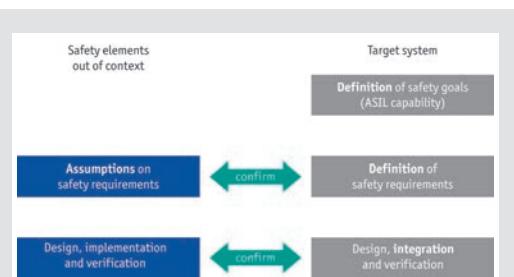


Figure 2: Prior to the development of a so-called "Safety Element out of Context" (SEooC) assumptions must be made and documented

Generic components (subsystems, hardware components, software components) that still do not have any concrete use case ("items") during development - and therefore no safety goals are defined for them - can be developed according to ISO 26262 as so-called "Safety Elements out of Context" (SEooC).

In this case, since the applicable "safety requirements" to be fulfilled with the help of the element are unknown, "assumptions" must be made and documented accordingly (**Figure 2**).

When the SEooC is first used, a supplied Safety Manual must be checked to determine whether the assumptions that were made agree with the safety goals and safety requirements defined in the use case, its conformance is supported and no contradictions result. Correct use according to the safety manual must then be assured and verified.

Generic Implementation of the Validation Layer

TTTech Automotive has developed a generic monitoring layer in its "SafeExecution" product that fulfills the cited requirements for "coexistence" and freedom from interference and contributes to cost reduction by efficient reuse of existing components. With module integration for memory protection and program flow monitoring, it is possible to reliably detect potential errors in QM developed portions of the basic or application software, and suitable reactions to errors can be initiated (**Figure 3**).

Users must integrate the SafeExecution modules based on their safety manuals to achieve a system that satisfies the requirements of ISO/DIS 26262.

Besides its proven solution for end-to-end communication validation "SafeCOM," SafeExecution is the second module from TTTech that is used to develop safety-related ECUs.

MICROSAR Safe as an Integrated Solution

To get validated basic software from a single source Vector and TTTech integrated the generic software modules SafeCOM and Safe-Execution into MICROSAR – the practice-proven AUTOSAR solution from Vector (**Figure 4**).

Re-use of certifiably developed central software components reduces costs for integration of the application. Instead of an application-specific solution, TTTech and Vector are together offering a generic standard solution that minimizes development costs for ISO 26262 conformant safety-related ECUs.

Translation of a German publication in
Elektronik automotive, 11/2010

All Figures: Vector Informatik and TTTech Automotive

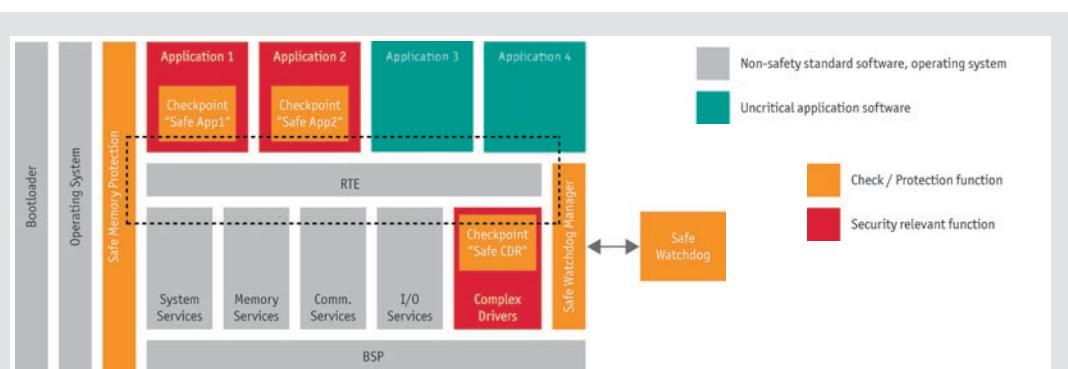


Figure 3: The SafeExecution modules must be integrated in an existing system according to the safety manual to fulfill the requirements of ISO 26262

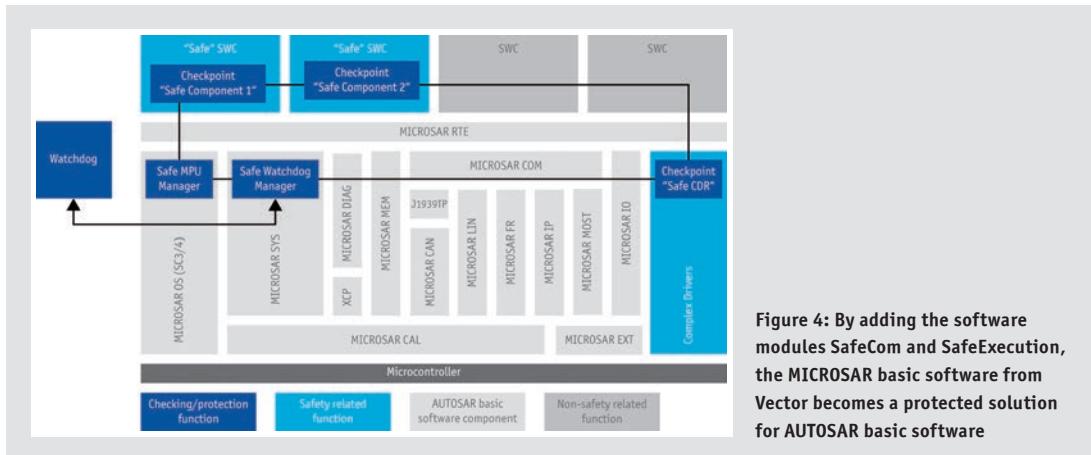


Figure 4: By adding the software modules SafeCom and SafeExecution, the MICROSTAR basic software from Vector becomes a protected solution for AUTOSAR basic software



Joachim Kalmbach (Dipl.-Ing. (FH))
studied Computer Science and Automation at the University of Applied Sciences at Reutlingen. In 2006, he joined Vector Informatik GmbH in Stuttgart where he is currently a Product Manager in the Embedded Software area. His work focuses on AUTOSAR and Functional Safety.



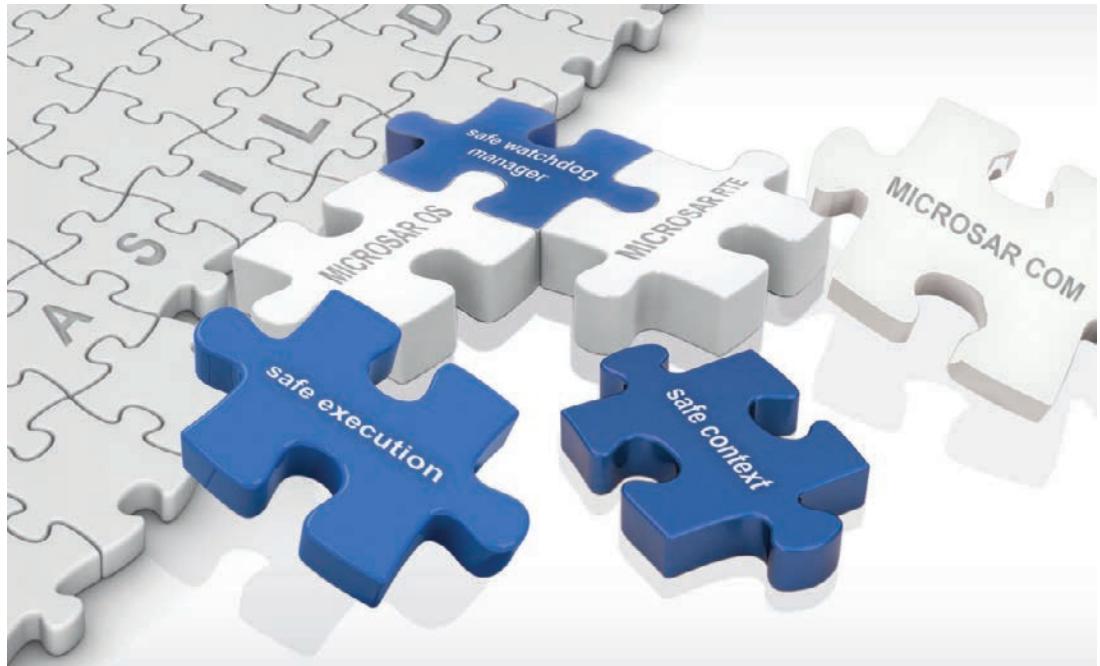
Dr. Thomas Wenzel
Since 2006, he has been working at the Institute for Vehicle Technology and Mobility, where he has been managing the competency field of Functional Safety since March 2010. Before assuming this position, he earned his doctorate degree in Control Engineering at Coventry University.



Martin Fassl
is a Graduate Engineer in Engineering Physics (Technical University of Vienna); since 1997 he has been working in hardware and software development in the telecommunications, aerospace and automotive fields. Since 2008, he has been working at TTTech Automotive GmbH as Product Manager for Development Tools and Standard Software.

Intrinsic Safety of AUTOSAR Basic Software

Integrating software components with different safety relevance in one ECU



The newly established automotive standard ISO 26262 defines processes for developing safety-related ECU software. A high level of intrinsic safety of the individual software components is required to ensure that derived system-specific safety goals conform to the standard. This is also necessary to prevent potentially hazardous situations from occurring in case of error.

In developing safety-related ECUs, the primary focus is on the safety of new customer functions. Another challenge must be overcome as well: Older ‘tried and true’ functions that are carried over must be combined with ‘service functions’ (such as diagnostics or bootloaders) and critical core components, and they must conform to all functional safety requirements of ISO 26262. This requires evaluating the risk potentials of both new and familiar functions and implementing them according to the standard. Risk potential is classified as an Automotive Safety Integrity Level (ASIL). When the various classification levels are assigned to vehicle functions, this results in a broad distribution of levels ranging from QM (“quality managed”, no safety relevance) to ASIL D (maximum safety relevance). In integrating functions with different safety relevance levels, the standard requires that “freedom from interference” must be assured [1]. Carryover parts and software developed based on a QM process must not interfere with the safety-relevant function. As a result, combining functions with different

ASIL classifications can require cost-intensive developments, because proven functions need to be modified. This is often impossible to achieve when one considers the narrow time windows of development cycles and the close mutual interactions of many sub-components.

To attain a lean implementation of the standard’s requirements, a modular software platform is ideal; it must permit integration of software functions of different producers and different safety classifications – ranging from QM to ASIL D – on a single ECU. Such an implementation will be presented in the following. For the customer, the platform enables reuse of proven but non-safety relevant software in unmodified form, while maintaining the highest levels of safety requirements.

Integrating software functions from different manufacturers and mixed ASIL classifications in one ECU

The core requirement of ISO 26262 for the integration of mixed critical system parts is that the safety-relevant software must not be affected or disturbed. The concept of “freedom from interference” consists of three sub-requirements:

- > No unintentional changes are made to the memory of the safety-related function, e.g. it is not destroyed by “faulty pointer accesses”.
- > The safety-relevant function always has the specified execution time available, i.e. it is not erroneously interrupted, started too late, etc.
- > The data that the safety-related function receives or sends remains unaffected – i.e. it is not delayed, blocked, modified, etc.

In the following, a strategy is described that allows the coexistence of safety-relevant and non-safety relevant software modules on one ECU, such as a safety-relevant inverter monitoring unit and a non-safety relevant diagnostic module. The individual actions of this strategy assure that all three requirements are fulfilled.

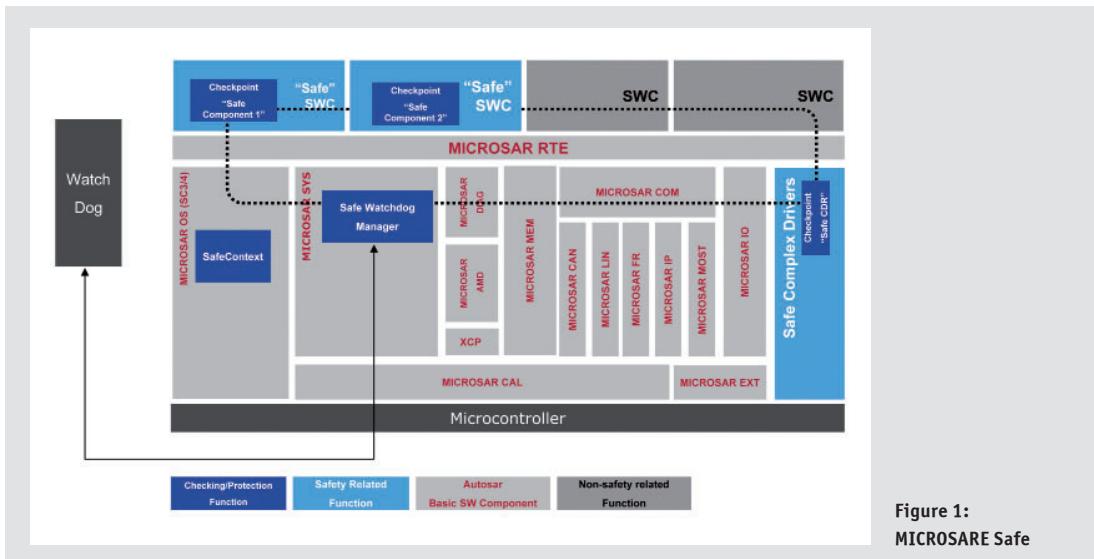
The standard does not absolutely require that existing software (e.g. AUTOSAR basic software) must be redeveloped to guarantee freedom from interference, as long as the safety-relevant software is not negatively affected or interfered with. Nonetheless, freedom from interference between all software components – up to classification level ASIL D – must be constantly assured for the safety of the overall system. This is accomplished by additional safeguarding of software components by independent safety modules for memo-

ry access, program flow and communication areas as shown in **figure 1**. Presented in the following are the functions of individual safety modules that are used to achieve validation of the overall system in ASIL D according to the requirements of ISO 26262 – with reuse of existing basic software components. German based quality specialists TÜV has pre-certified this solution to ISO 26262.

Protection against memory violations

Protection against unauthorized access to the memory area of a safety-relevant software component is assured by an operating system that supports memory protection mechanisms, specifically a MICROSAR operating system of Scalability Class 3 or 4 (AUTOSAR OS SC3/SC4). In this case, a Memory Protection Unit (MPU) implements the memory protection mechanism; this solution assumes the use of a suitable microprocessor. The software part of the MICROSAR operating system known as SafeContext controls the isolation of software components during task switches and interrupts. This prevents one software component from writing to the memory of another software component without permission. SafeContext was developed to ASIL D and is therefore authorized to reconfigure the memory ranges of the various tasks and interrupts that are protected by the MPU at runtime. This arrangement also assures conformance to ASIL D in saving and restoring context data, including the MPU configuration.

The concept of the OS Application specified in AUTOSAR is used to configure the operating system. This involves logically grouping operating system objects such as tasks and interrupts and assigning them to MPU configurations. Basic software modules



developed according to the QM process are combined in a separate OS Application and are isolated from the software with ASIL classification. This also prevents the memory of ASIL software from being overwritten by the basic software.

Protection against runtime interference

The Safe Watchdog Manager (SWdgM) is the safety mechanism that is responsible for timing and program flow monitoring (**Figure 2**). Safety-relevant functions are monitored to ensure that the functions are executed in their correct sequential flow. Checkpoints in all relevant software components regularly report to the Safe Watchdog Manager on the program flow. This enables reliable detection of all types of runtime interference. If an application, interrupt or function is not activated in a timely manner, the Safe Watchdog Manager detects this and initiates a reaction to the error. Along with time durations, correct sequence in the program flow is also monitored. The Safe Watchdog Manager services the independent Hardware Watchdog, and – as a last resort – can reset the ECU to a safe state in case of errors. If the Safe Watchdog Manager itself is not correctly activated, the independent watchdog is not serviced, and the error reaction is also reliably initiated in this case.

Protection against corruption of input/output data

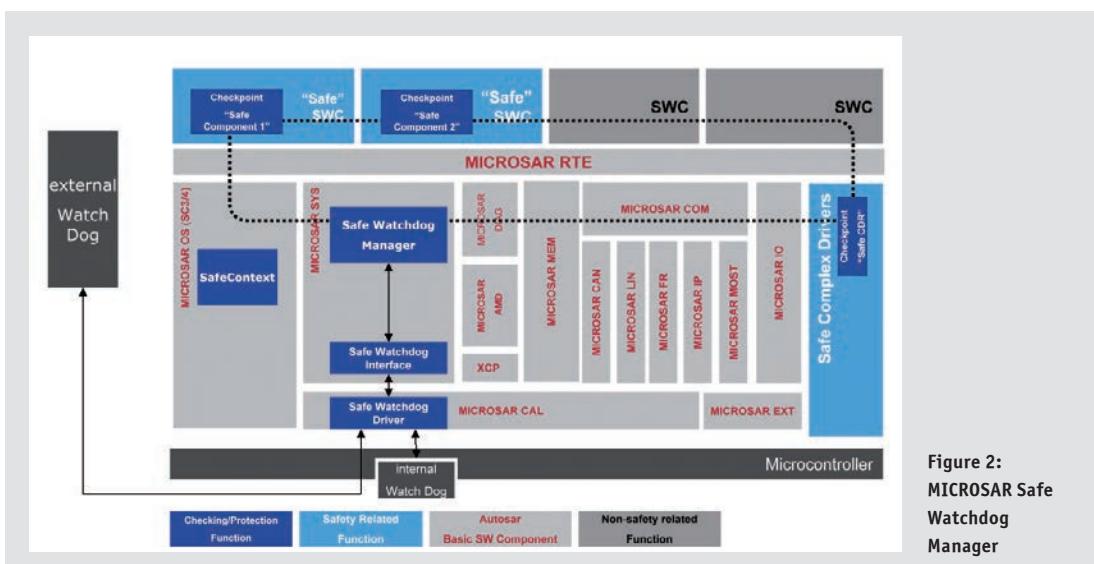
The End-To-End Library (E2ELib) safeguards safety-relevant communication in the Tx and Rx directions (**Figure 3**). The data is protected by a checksum here to enable detection of any corrupted

data contents. A message counter is used that can detect an incorrect message sequence, message failure or undesired repetitions.

Summary

All three of the safety modules presented here supplement one another in their functionalities, and together they achieve freedom from interference. They fulfill the highest requirement levels of ISO 26262 and can also be used in ASIL D ECUs. When they are integrated in MICROSAR – the AUTOSAR solution from Vector – they also represent validated basic software from a single source.

The development effort that would otherwise be necessary to bring software up to higher ASIL levels is avoided, where it is unnecessary, by using software with various levels of safety relevance and providing freedom from interference for these software components. This leads to substantial cost optimizations. Integrating or linking pre-certified safety modules developed in compliance with ISO 26262 ASIL D creates an overall software platform that conforms to the highest necessary ASILs. Safety modules can be integrated together with software applications and AUTOSAR basic software components developed according to a QM standard. Integrating ASIL D safety modules that guarantee freedom from interference makes it possible for software modules of different safety relevance to coexist. This can result in considerable cost savings in the development of safety-relevant modules. It also shortens development cycle times.



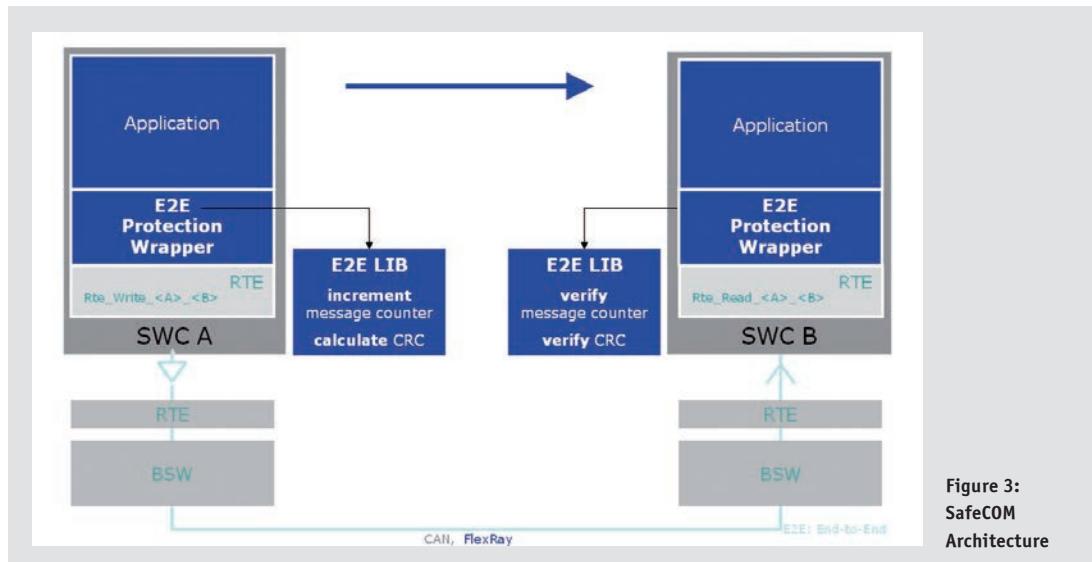


Figure 3:
SafeCOM
Architecture

[1] ISO/DIS 26262: Road vehicles – Functional safety, Part 1-10.

International Organization for Standardization, 2009.

<http://www.iso.org/>

Translation of a German publication in Hanser automotive, issue
3-4/2012

Figures:

All figures: TTTech Automotive and Vector Informatik GmbH



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

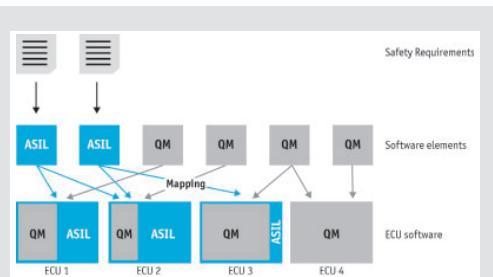


Figure 1:
Example of a Mixed-ASIL system with function mapping to multiple ECUs

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.

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**):

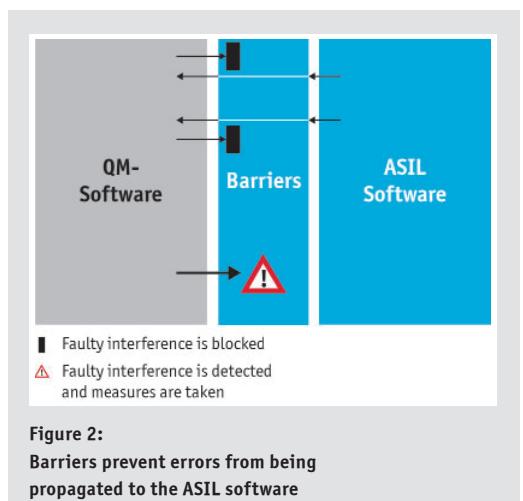


Figure 2:
Barriers prevent errors from being propagated to the ASIL software

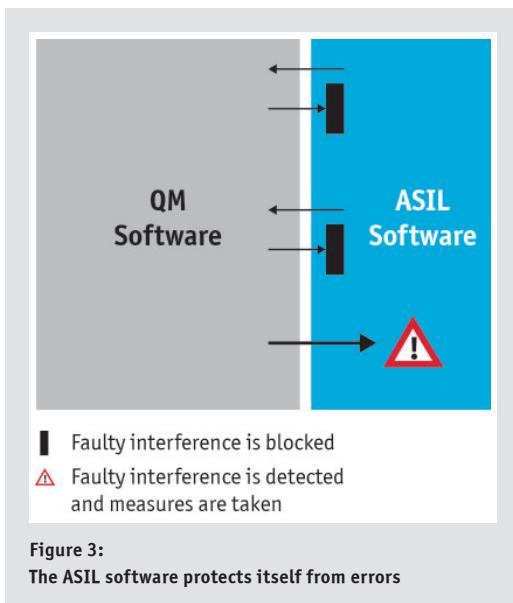
- > Redundant and possibly diversified computation – including validation of the ASIL software's algorithms
- > Redundant and possibly diversified storage of data
- > Deactivation of interrupts in critical areas, e.g. in areas in which comparisons and decisions are made.

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.

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.

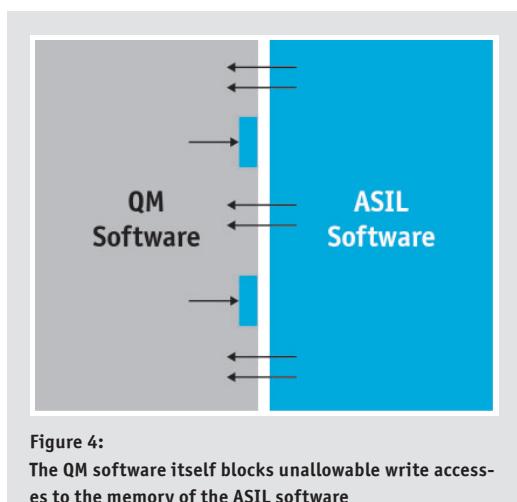
Therefore, one highly efficient approach is to leave the basic software on the QM level and enable coexistence with the ASIL software without the use of an MPU. MICROSAR Safe offers this type of solution under the name SilentBSW – which is a non-interfering basic software.

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.

The article is based on a speech of Günther Heling at the VDI conference Baden-Baden-Spezial, VDI report 2172, 12/2012

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.



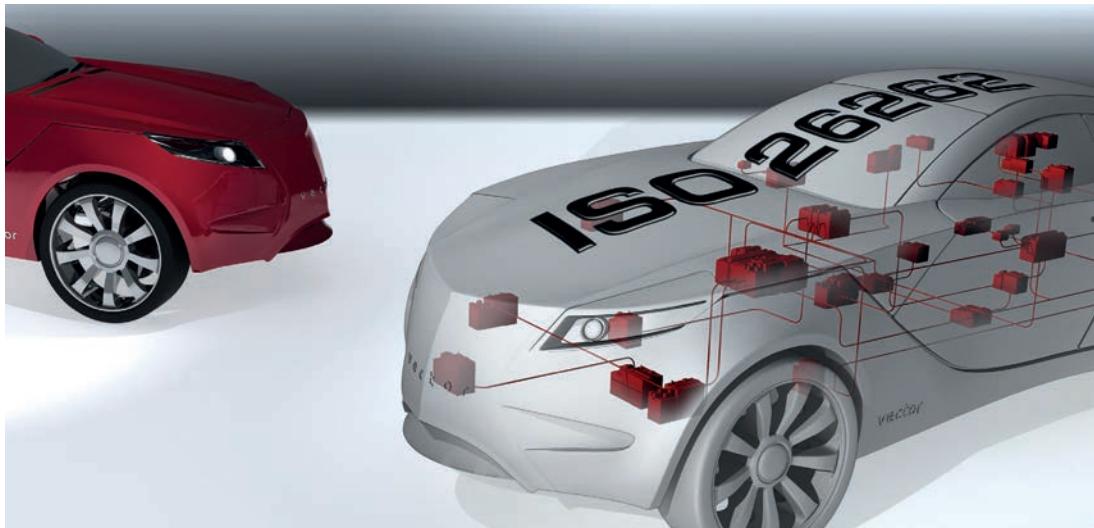
Dipl.-Ing.(FH) Jochen Rein
is Manager Product Management and Sales for Embedded Software at Vector Informatik GmbH in Stuttgart.



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.

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

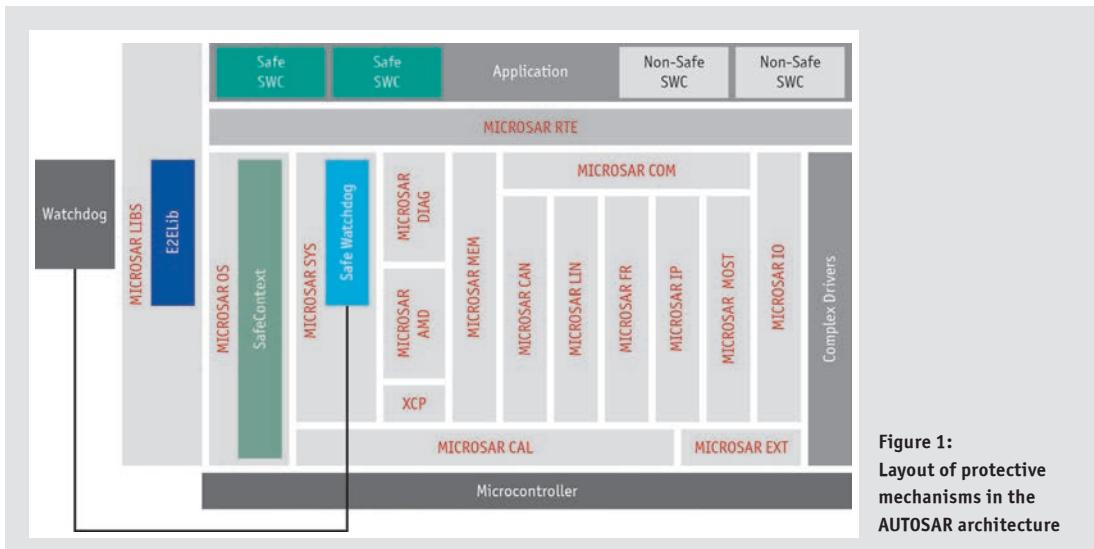


Figure 1:
Layout of protective mechanisms in the AUTOSAR architecture

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

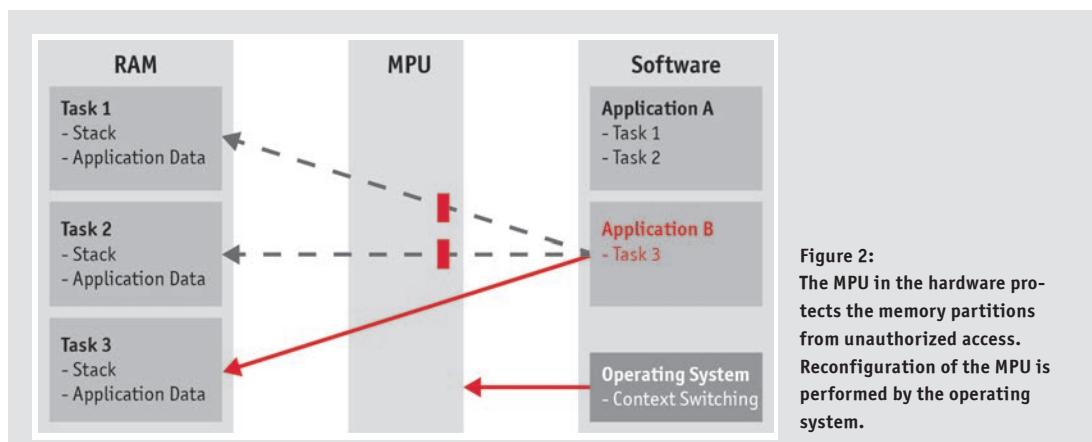
All figures: Vector Informatik GmbH

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 einem Steuergerät" ("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:

- Homepage Vector: www.vector.com
- Product Information MICROSAR Safe: www.vector.com/vi_microsar_safe_26262_en.html





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 PREVision, 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, ①. Operating situations and modes can be predefined and standardised, and automated ASIL assignment can be used.

FUNCTIONAL SAFETY CONCEPT

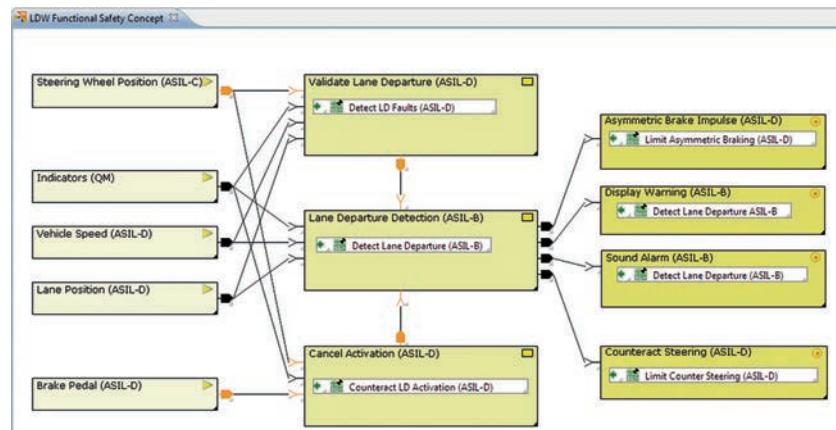
In creating the functional safety concept, safety goals are refined by functional safety requirements, and allocated to the system functions that should fulfill these requirements. This approach progressively leads to the functional safety concept, which is independent of later implementation of functions in hardware or software. ② 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 Hazard Analysis												
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.			Could occur in almost all driving cycles.	E4	Could have potentially fatal consequence S	S3	In most situations, the driver will rely on other indicators to realise that he is straying from the lane	C1	ASIL-B	
2												
		Missing reaction A lane departure warning is not activated, even though the vehicle is straying from the lane without indicating.			Could occur in almost all driving cycles.	E4	Could have potentially fatal consequence S	S3	In most situations, the driver will rely on other indicators to realise that he is straying from the lane	C1	ASIL-B	
3												
4												
		Reaction too late Counteraction measures are performed, but too late to avoid an accident.			Could occur in almost all driving cycles.	E4	Could have potentially fatal consequence S	S3	In most situations, the driver will rely on other indicators to realise that he is straying from the lane	C1	ASIL-B	
5												
					Could occur in almost all driving cycles.	E4	Could have potentially fatal consequence S	S3	In most situations, the driver will rely on other indicators to realise that he is straying from the lane	C1	ASIL-B	
6												
		Unnecessary counteraction Counteraction measures are performed although the vehicle is not straying from the lane.			Could occur in almost all driving cycles.	E4	Could have potentially fatal consequence S	S3	High probability that driver cannot control the situation due to high speed and unexpected action	C3	ASIL-D	
7												

① Hazard and risk analysis



② Functional safety concept

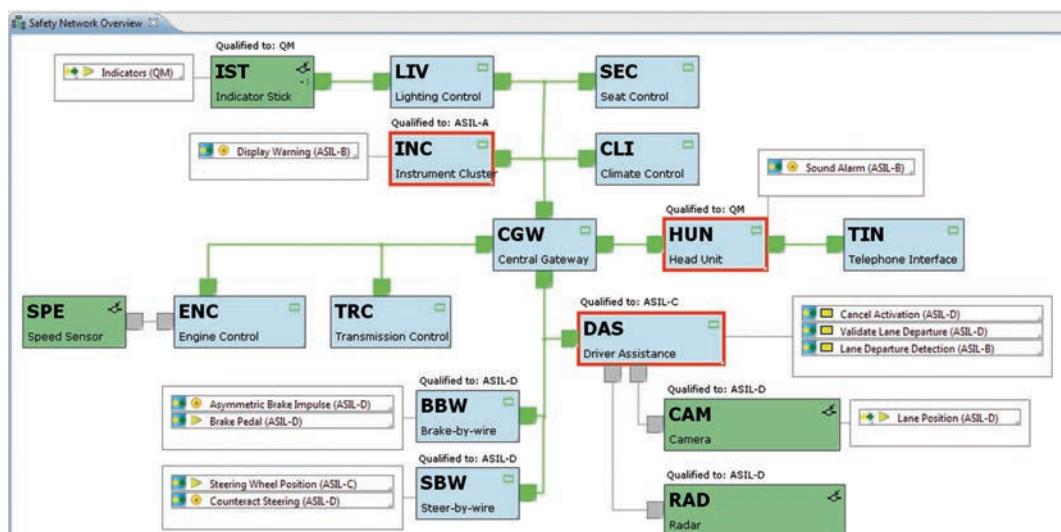
into technical safety requirements, and system functions are assigned to elements of the technical system architecture, ③. In an early stage of development, it is sufficient to make assignments to the components of the system. In a later stage, the implementation in hardware or software elements is done. 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. ④ 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



③ Technical safety concept

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.

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 compo-

ment 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:

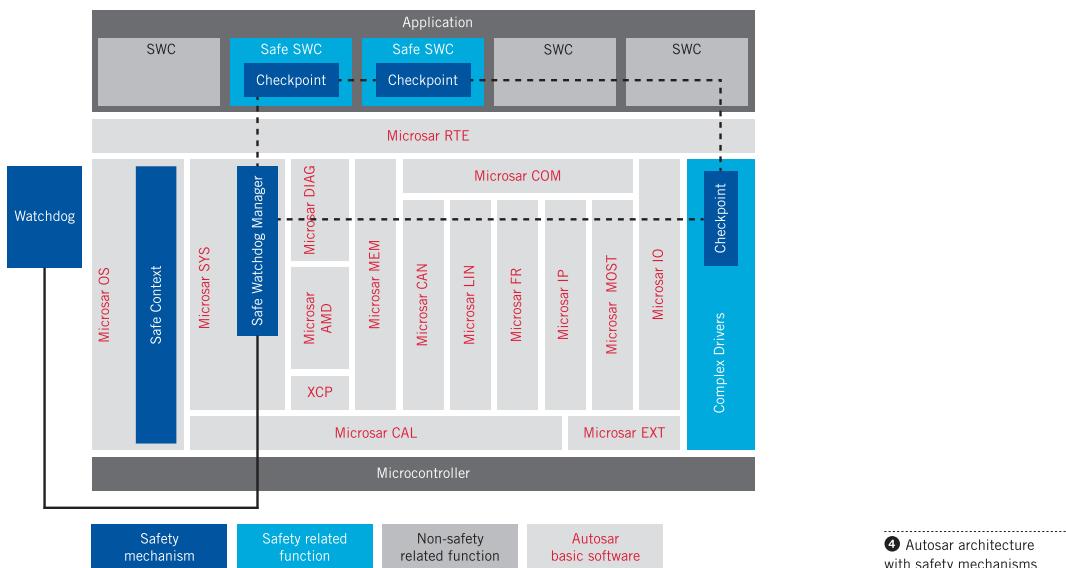
- Correct timing behavior of the software: The Watchdog Manager is used to check for correct timing behavior. The user defines checkpoints, and when a checkpoint is reached the application software calls a function. If a checkpoint is not reached within the allowed time frame, or if checkpoints are reached 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



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 implementa-

tion 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



Solutions for Functional Safety

Safe ECUs according to ISO 26262

Benefit from Vector's solution for functional safety in your ECU development projects:

- > Develop and analyze safety-related E/E architectures with **PREEvision**
- > **MICROSAR Safe** for freedom-from-interference in running non-safety and safety-related software content (mixed ASIL systems)
- > **Training and coaching** in functional safety to qualify technical specialists and managers, and **consultation** on safety-related development projects

Vector is your competent partner in developing safety-related ECUs – based on experience, dedication and practice-proven products.

- Information and downloads: www.vector.com/safety

Timing, memory protection and error detection in OSEK systems

by Peter Liebscher, Vector Informatik

A powerful certified OSEK implementation or AUTOSAR-conformant embedded operating system, together with a universal tool-chain, will make it possible to master the complexity of future electronic development.

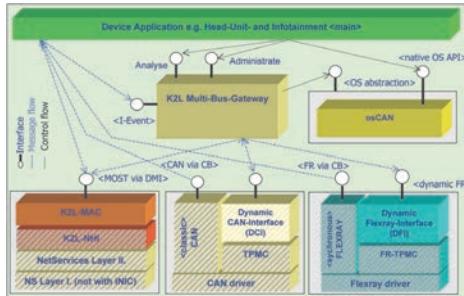


Figure 1. Representation of gateway, system and bus APIs

■ The real-time and multitasking operating system OSEK is the standard in automotive embedded developments today. Its most important properties include low consumption of processor resources and memory, and event-driven task management that effectively handles both cyclic and non-cyclic program blocks. Continuing advances in automotive electronics and the new HIS and AUTOSAR standardization initiatives also place new demands on the operating system. Areas of emphasis in future operating system versions will be timing and memory protection.

In the 1990s the large automotive OEMs introduced the OSEK/VDX operating system specification with the goal of establishing a uniform standard for software in electronic control units. The wide variety of proprietary embedded operating systems at numerous suppliers had proved to be an obstacle to smooth integration in light of the growing significance of automotive electronics. Besides defining the actual operating system core, OSEK also defines communication services and functions for network management.

Since most automotive suppliers had already committed to a preferred operating system beforehand, automotive OEMs had to introduce OSEK persuasively in some cases. DaimlerChrysler, for example, made OSEK mandatory

as a standard for new developments, both for in-house developments and those at suppliers. The company organized OSEK training courses, created OSEK design guides and supported operating system producers. It also financed one OSEK licence per supplier, whereby only certified OSEK operating systems were permitted. The costs of introduction reached a high point in 2002 and then dropped significantly. In the meantime OSEK has generated its own success, and now most ECUs in the automotive field run on OSEK operating systems.

Efforts have paid off: applications based on OSEK have fulfilled expectations by their improved code quality, structuring and the ability to integrate components from different suppliers. At gateway producer K2L solutions with osCAN, the OSEK/VDX-conformant operating system from Vector Informatik, have demonstrated fast reaction times and precise timing. Last but not least, what has proven to be decisive for OSEK in conjunction with a table-driven interpreter concept are flexibility gains leading to shorter development times, higher quality and functionality and ease in creating variants. Leading the way here were comparisons between OSEK with its preemptive scheduling and a fixed coded static approach with cooperative scheduling. The "osCAN" OSEK implementation has proven itself, not only in ECUs but also in the interface hardware

of the same company. In the MOST Interface VN2600 osCAN's capabilities were put to the test under a 133 MHz Altera Excalibur controller: its processing of up to 35,000 Events/s corresponds to a data throughput of 1.7 MB/s.

Requirements of an operating system grow in parallel with the continuing penetration of electronic technologies. In particular the advance of safety-relevant applications in the vehicle, such as fully electronically controlled steering and braking systems (X-by-wire), make deterministic behaviour essential under peak loads and fault conditions. For example, the specification of OSEKtime will supplement the event-driven OSEK with a time-triggered variant.

Fault tolerance, error detection mechanisms and memory protection also play an important role in achieving system reliability. This has acquired special relevance, because in the future an ECU will handle multiple applications running simultaneously. In "Hersteller Initiative Software" (HIS; Hersteller = producer or OEM) the large German automotive OEMs have come to an agreement on the standards needed to implement the named functions. Working committees have been established in the areas of software, software testing, process assessment, simulation and tools as well as flash programming. The AUTOSAR (automotive open system architecture) consortium is responsible

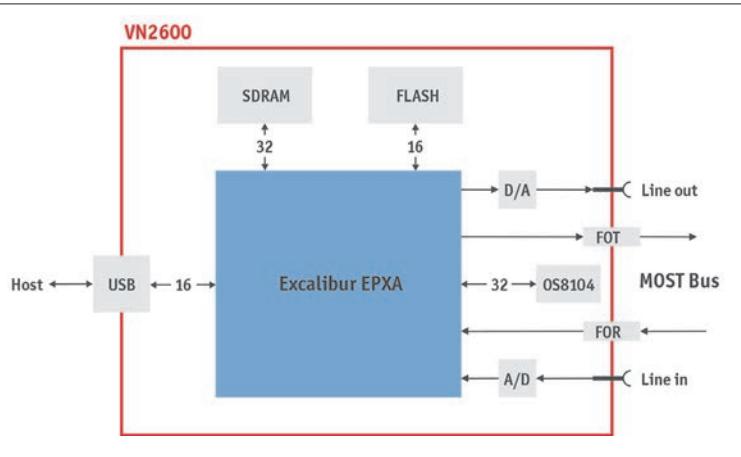


Figure 2. Function block diagram with MOST interface VN2600 and an Altera Excalibur controller

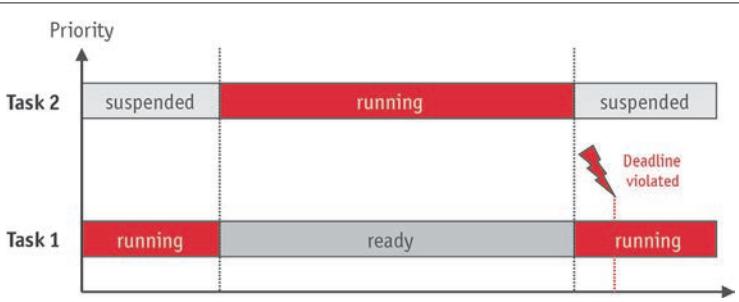


Figure 3. Deadline monitoring to detect deadline violations. The error source of Task 2 is not found in Task 1.

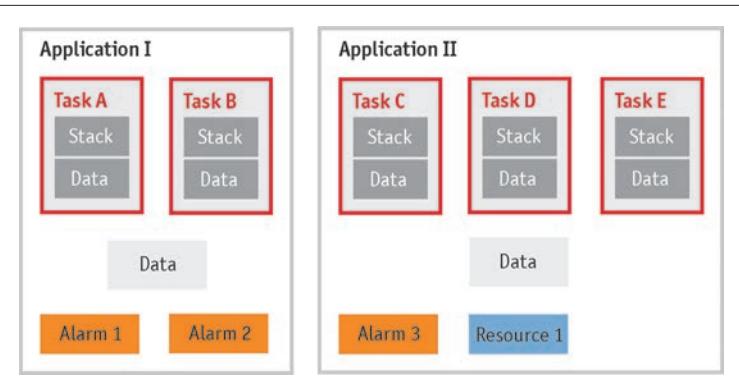


Figure 4. Tasks belonging to the same application must be able to access the same memory areas.

for the latest standards for future vehicle generations, whereby the HIS group is integrating its results into AUTOSAR and is representing uniformity interests there. When one considers that over 50 ECUs operate on a luxury automobile today and that there appears to be no end to potential new automotive electronic applica-

tions in the foreseeable future, it becomes clear why limiting the number of control modules in the vehicle has become a pressing topic. This objective can only be achieved if multiple applications run on the same control module. The multitasking OSEK operating system was specified for this purpose. However, other

properties are required of the operating system for use in safety-critical systems and to integrate the software of different producers. For example, an application must not disturb other applications running in parallel. To prevent this from happening, new operating system features are aimed at optimal monitoring of the time behaviour of individual tasks and universal memory protection.

Progress has varied considerably in the various development levels as a result of these efforts. OSEKtime, specified since 2001 for time-triggered tasks, with its deadline monitoring, only begins to cover functions that will be necessary in the future. Deadline monitoring detects whether a task has ended by a prescribed point in time. Unfortunately, the method cannot discern the causes for deadline violations. For example, if the monitored task was interrupted by a higher-priority task, then the monitored task is not really responsible for being unable to satisfy the prescribed time window. In HIS-conformant OSEK extensions memory protective functions are defined in addition to deadline monitoring. The most advanced is AUTOSAR with its "execution time enforcement" and "arrival rate enforcement" methods, which give low-priority tasks a mandatory minimum time budget, too. These methods are able to clearly identify error sources. Furthermore, in the various operating system versions, Vector Informatik has integrated options for run-time measurements.

Memory protection functions restrict a task's access to the memory space assigned to it. This applies especially to preventing write accesses to other data segments, detecting stack overruns, and detecting execution of incorrect code. Tasks belonging to the same application, on the other hand, must be able to jointly access the same memory areas. However, special system functions such as drivers could require full unrestricted memory access.

A distinction is made here between so-called "trusted applications" with full access rights and "non-trusted applications" with restricted access rights. These names can lead to some confusion: non-trusted applications are programs that, due to restricted memory access, cannot cause any damage. The trusted applications, on the other hand, are given quasi blind trust. The latter are easy to use but represent risks to system security and cannot provide identification of errors or error sources. It is good design practice to place functionalities in non-trusted applications whenever possible. Vector Informatik therefore offers implementations that also permit calling of non-trusted functions. They are intended for safety-critical applications and offer a maximum level of monitoring. A related proposal for handling this issue in

AUTOSAR is currently being discussed in a working sub-committee. The cited timing and memory monitoring functions can only be implemented efficiently with suitable hardware support. What are needed for memory protection are memory protection units (MPUs) that are tailored to the needs of automotive applications in terms of options offered for number and sizes of memory blocks. In many cases today the smallest units that can be managed are blocks 16 KB in size. In the automotive embedded field, however, substantially smaller memory units are needed. Essentially,

the hardware requirements of current and future OSEK real-time systems can only be fulfilled by a complete redesign of today's processor cores. Desired features are currently being designed together with semiconductor producers. Among the most important requirements, besides the named monitoring functions, are an interrupt controller for different interrupt levels with low latency times, hardware support in task switching, and processor cores with as few registers as possible. For hardware-related and time-critical automotive applications what counts is the ability to react as quickly as pos-

sible. Many of these applications consist of drivers and interrupt service routines (ISRs) which in contrast to the workstation field belong to the application here. It is problematic that on today's controllers the ISRs often can only be disabled completely. In general, disabling mechanisms must be made more efficient in implementations, since this basic

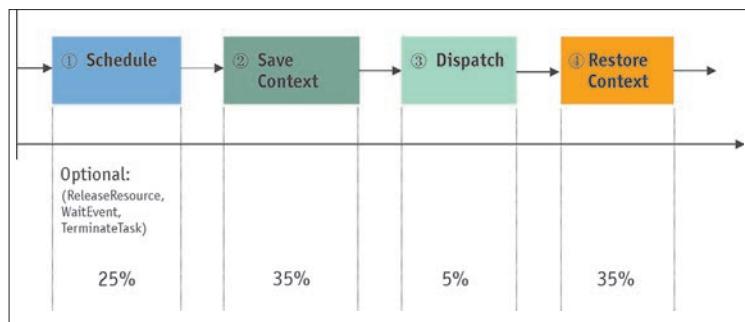


Figure 5. Phases of a task switch

function is executed very frequently. The quick task switches by which real-time embedded systems “thrive” are currently given just rudimentary support in hardware. The majority of resources is consumed in saving and restoring the context. The context is made up of core registers, register banks, memory access registers, floating point and arithmetic registers of the stack pointers, special peripheral units and a number of operating system variables. A fully hardware-supported context switch would be ideal here.

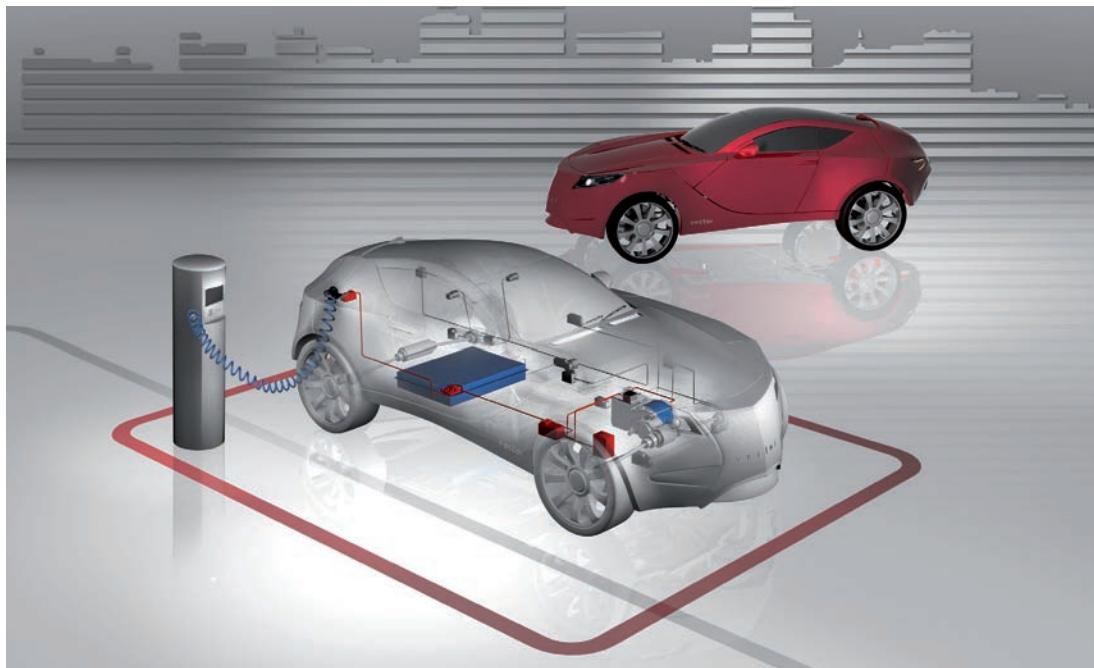
Furthermore, it has been shown that processors with a low number of registers offer better performance. Many registers can only be used meaningfully in typical workstation environments, because that is where individual program sequences run for a relatively long time without interruption. A potential trend here could lead in the direction of so-called softcore processors and to compilers that permit configuration of the registers used. Interesting possibilities for mastering industrial complexity are being researched in ongoing projects involving the intensive application of mathematics to the engineering sciences. Systematic analytical methods can be used to reveal critical situations in the time behaviour of an OSEK system that would otherwise not be found even by extensive testing. In this context, tools from the SympTAVision company enable targeted searches for “bottlenecks” and “hot spots”, informing users of worst-case situations. The advantages of systematic analysis lie in

reduced testing effort, productivity gains, quality improvement and comprehensive system optimization. Scientists on the Verisoft project have taken this one step further in concluding that it is possible to develop absolutely error-free embedded systems and electronic components. They turn to the methods of formal verification to verify entire systems consisting of hardware, software, operating system communication, applications, etc. In cooperation with project partner Infineon, the TriCore 2 processor, the future flagship of the company’s 32-bit microcontroller device line, offered the first evidence that this innovative technology could be applied to highly complex designs. The long-term Verisoft project set up under the project leadership of Prof. Dr. Wolfgang J. Paul, of the University of Saarbrücken, is being supported by the German Federal Ministry for Education and Research.

Foundations and base technologies have been created for achieving reliable electronic systems in the automobile. Specific challenges must still be overcome by controller producers. Apart from that it is the responsibility of automotive OEMs and suppliers to utilize the available resources optimally. A powerful certified OSEK implementation or AUTOSAR-conformant embedded operating system, together with a universal tool-chain, will make it possible to rationally master the complexity of future electronic development. ■

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.

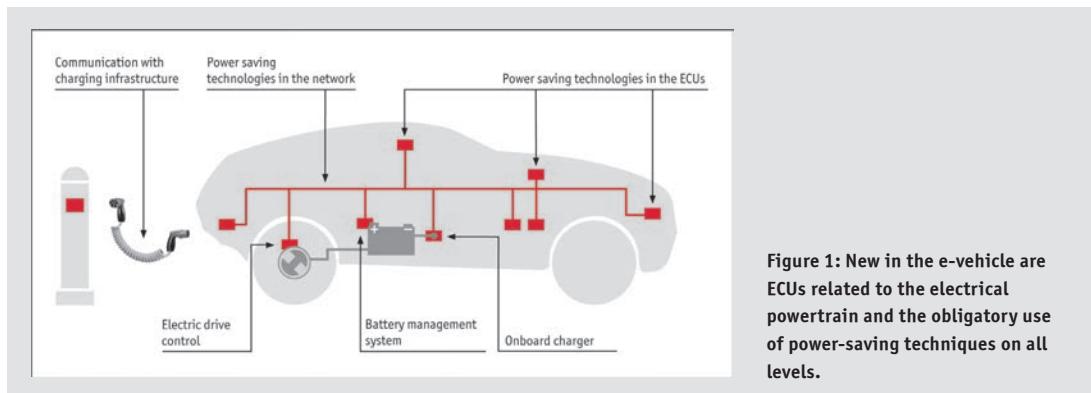


Figure 1: New in the e-vehicle are ECUs related to the electrical powertrain and the obligatory use of power-saving techniques on all levels.

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 CO₂ 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 processors 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.



Figure 2: The power supply module of the VT System determines the DUT's current consumption with high measurement precision and time 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

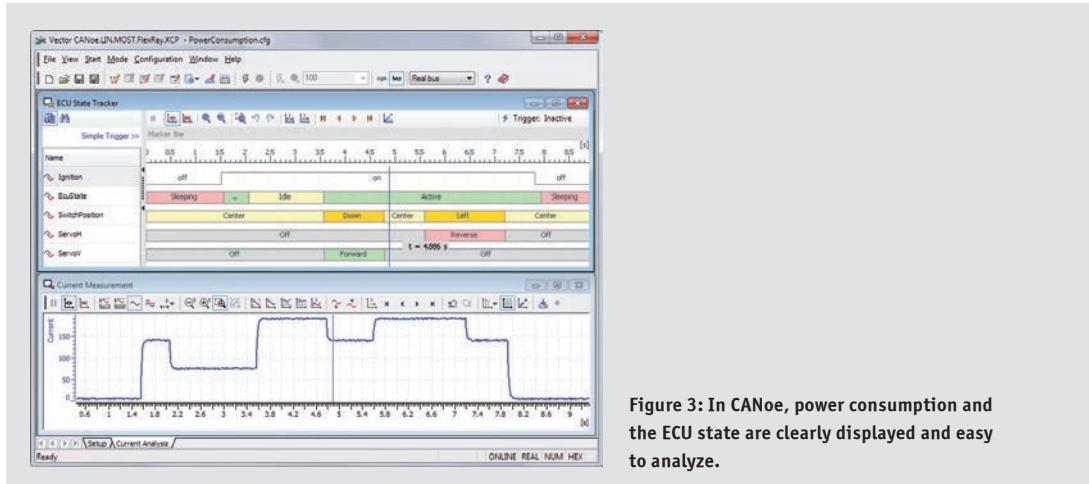


Figure 3: In CANoe, power consumption and the ECU state are clearly displayed and easy to analyze.

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.

**Translation of a German publication in
Automobil Elektronik, August/2011**

All figures: Vector Informatik

Links:

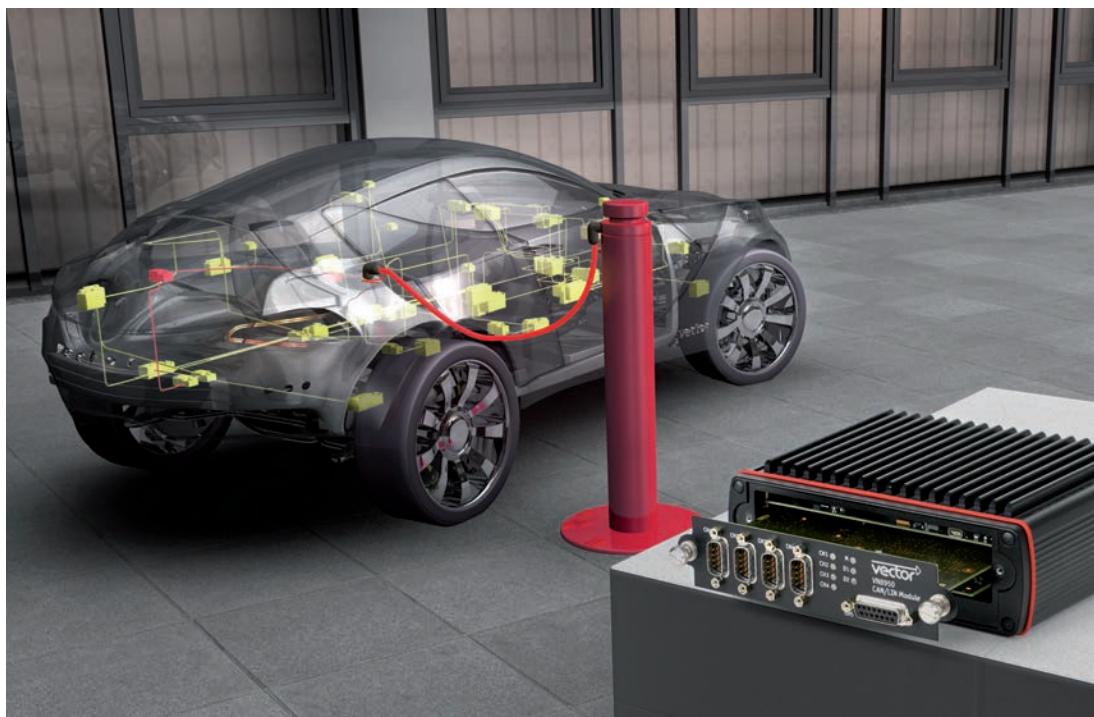
Homepage Vector: www.vector.com

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**).

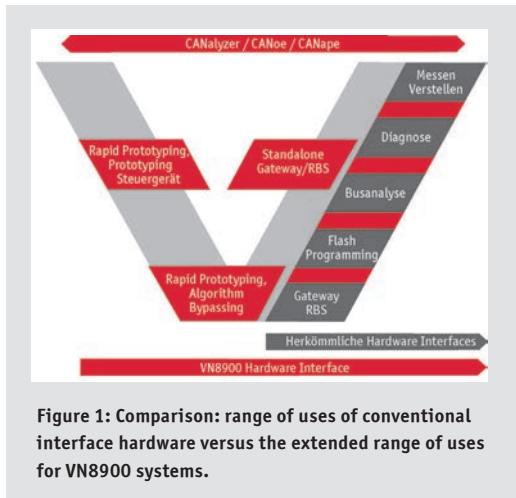


Figure 1: Comparison: range of uses of conventional interface hardware versus the extended range of uses for VN8900 systems.

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 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.

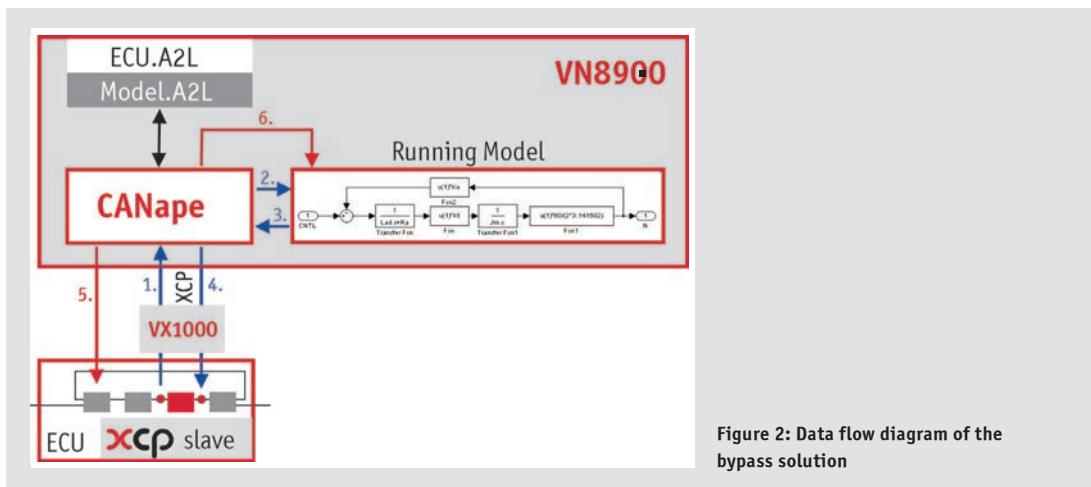


Figure 2: Data flow diagram of the bypass solution

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 system 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.

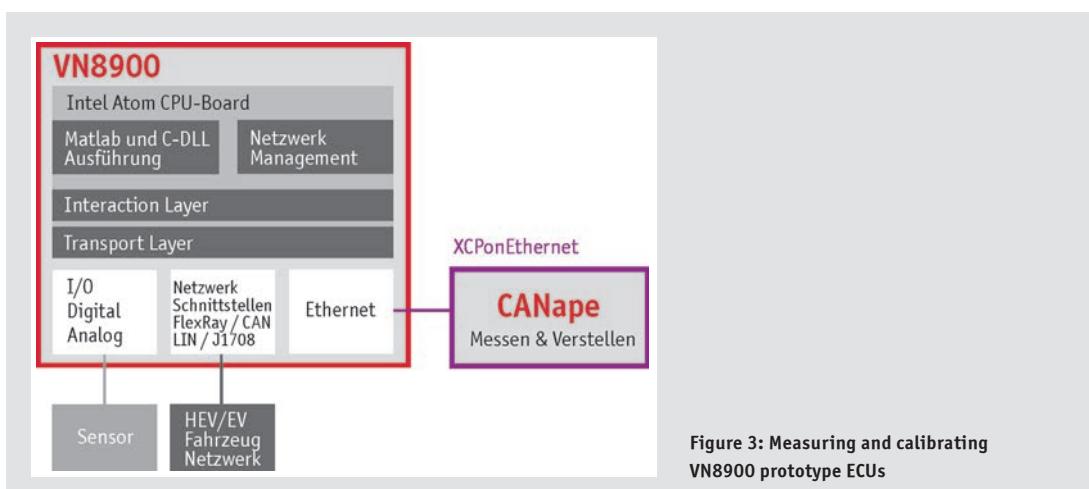
Unlike existing solutions for ECU prototypes, no cross-compiler is used in the VN8900 approach. This means that all familiar and proven CANoe functions – such as network management, the interaction layer and transport protocols (e.g. for diagnostics) – can be used directly in ECU simulation.

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.

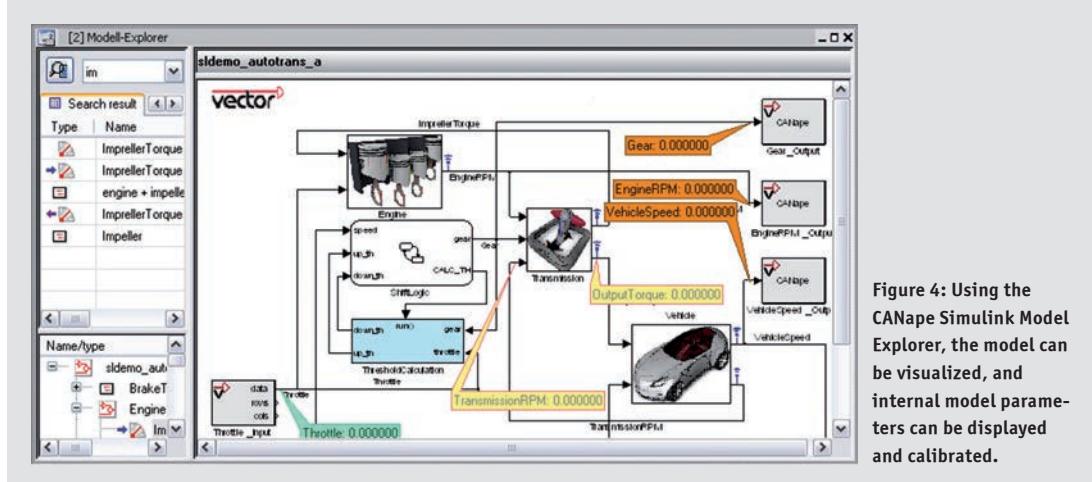


Figure 4: Using the CANape Simulink Model Explorer, the model can be visualized, and internal model parameters can be displayed and calibrated.

Links:

Homepage Vector: www.vector.com

Product Information CANape: www.vector.com/vn8900

**Translation of a German publication in
Hanser Automotive, 03-04/2011**



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.

Electric Mobility Makes Great Strides

626.6 kilometers under real conditions on one battery charge



Today, there are numerous research projects that focus on topics related to electric mobility. One aspect that is still considered a critical issue is the limited driving range of electric vehicles. The so-called "Schluckspecht" project holds special interest here; in the Solar Challenge 2010 in South Africa, a vehicle by this name covered 626.6 km – the longest distance an electric car has ever driven on a single battery charge in real traffic on public roads. This milestone was enabled by state-of-the-art drivetrain technologies and power electronics paired with highly professional implementation of the control and networking systems.

To build an electric car with a long driving range, optimizations are needed in all relevant disciplines. Necessary are a drive with high efficiency, light and compact batteries with large storage capacities (energy density), suitable power electronics, well-tuned control algorithms and efficient network communication. Of course, weight plays a crucial role here; this means that the mechanical construction of the chassis and body must be lightweight yet sturdy, and finally, safety aspects and in particular an effective brake system must not be neglected.

College project with a long tradition

The Offenburg University of Applied Sciences is also addressing this topic, and it can already reflect on over ten years of experience in electric mobility with its "Schluckspecht" student project. Team Schluckspecht has been participating in the European Eco Marathon

since 1998, for example, a competition that gives special recognition to the most energy-efficient vehicle. In 2008, the "Schluckspecht III plus" concept vehicle took first place in the fuel cell category with a low equivalent combined fuel consumption of just 0.032 liters of Super gasoline per 100 km; this is equivalent to a distance of over 3,100 km on one liter of Super. Together with its research partners, the team developed all key components of the motor, chassis and wheel suspensions and systems ranging to the vehicle electronics.

Encouraged by this and other successes, the team decided to participate in the South African Solar Challenge, a competition for alternative drive vehicles. It is held on public roads around the Republic of South Africa, where many different hill grades must be mastered. Because the "Schluckspecht IV E" (Figure 1) is a purely battery-powered vehicle that utilizes lithium-ion batteries, the team was participating outside of the regular competition, but



Figure 1: Schluckspecht IV E at the Solar Challenge in South Africa.

under the official oversight of the FIA (Fédération Internationale de l'Automobile). The goal was to make the public aware of the performance capabilities of the direct drive system implemented in the Schluckspecht vehicle. In the end, a driving distance of 626.6 km on a single battery charge was officially documented, more than had ever been driven by an electric car under comparable conditions on public roads.

Wheel-hub direct drive with iron-free exciter coils

The success story of the Schluckspecht IV E, a variant of the fourth generation of test vehicles, is based on a drive concept with wheel-hub motors, which the University of Offenburg developed in collaboration with the Stuttgart-based engineering company Evomotiv. Together with its twelve lithium-ion battery packs, the vehicle weighs about 320 kg and is driven by two wheel-hub motors, each with 42 poles and two kW of peak power. It is an advanced development of the "Schluckspecht City" vehicle variant that preceded it, and it achieves a high level of efficiency using brushless DC motors. In this motor type, the rotor has permanent magnets, while the excitation windings are located on the stator.

A special aspect of the Schluckspecht motor is the iron-free construction of the stator and excitation coils. Together with the direct drive and a sophisticated drive control system, this motor exhibits some interesting characteristics. For example, there are no cogging torques at all, which would otherwise occur in a conventional construction that is not iron-free. Periodic cogging torques cause mechanical oscillations and speed fluctuations, and they reduce efficiency. A crucial advantage – besides a higher start-up torque and low-noise operation – is that the wheel turns with hardly any resistive forces in the deenergized, excitation-free motor state; this makes it possible to omit a separation clutch as well as a transmission, differential, etc. Since direct drives generate forward propulsion precisely where it is needed, the

system attains an exceptionally high efficiency of up to 98 %. This engine concept was awarded the Bosch Innovation Prize in the year 2006.

Optimized commutation strategy with PWM drive control

Because of their brushless operating principle, wheel-hub drives operate nearly wear-free, just like inverter controlled asynchronous AC motors, which are the quasi-standard for industrial drive technology. One essential prerequisite, however, is electronic commutation, which ensures that the directions of the magnetic fields in the coils attract and repel the permanent magnets at the right times in alternation, based on the fundamental operating

principle of all electric motors. A prerequisite for this, and especially for the starting commutation, is precise knowledge of the rotor position. Several Hall sensors are used to acquire the positions of the permanent magnets. From this data, an evaluation unit generates four tracks (A, B, Strobe, Index) via a quadrature signal (**Figure 2**), which provide all necessary information on the turning direction and rotor position to a downstream type TMS320 digital signal processor (**Figure 3**).

The signal processor, whose features include two CAN modules for interfacing with the on-board network architecture, generates two PWM signals that are phase-shifted by 180° (**Figure 4**) to control the power electronics. The bridge transition can be used to switch each excitation winding to the positive (T1, T5, T3) or negative (T4, T6, T2) link and control the direction of the current flow or magnetic field. The project engineers and students use a relatively high PWM frequency to compensate for the low motor inductance. From the perspective of the coil, the frequency is switched four times per PWM cycle. Furthermore, to recover energy during braking (regeneration), the system can be switched between motor and generator modes by changing the duty cycle. The special aspect here is that it is possible to work with an optimized commutation strategy using the four alternating states at half of the PWM frequency.

Evomotiv and the Hochschule Offenburg each set up their own motor test benches for measurement and test purposes, and to calibrate motor parameters independent of the actual vehicle (**Figure 5**).

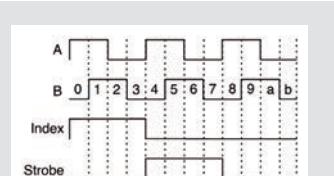


Figure 2: Quadrature signal

Networked electronics in the service of electric mobility

Basic elements of the Schluckspecht IV E electronic architecture are a central controller, a human-machine interface (HMI) ECU and several battery control modules. The latter continually monitor the voltage and temperature of the batteries. The central controller regulates the network of wheel-hub motors as a function of driver inputs or the HMI, and it also acts as the master for the battery control modules. In its role as master, the main controller can shut down the entire system very quickly in emergency situations by interrupting the energy supply. Moreover, this ECU serves as a central gateway (ZGW) for the vehicle's three CAN buses (**Figure 6**).

Vehicle control responsibilities include both safety engineering and vehicle dynamic control tasks. They are fulfilled by providing a dedicated motor controller for each wheel-hub motor. Vehicle dynamic control includes synchronizing the drive wheels when driving through curves – since there is no mechanical differential – as well as monitoring wheel slip. Electronic communication is distributed among the CAN1, CAN2 and CAN3 buses. While CAN1 connects the human-machine interface to the central controller, CAN2 is responsible for battery control. Because these two communication areas are not time-critical, they utilize the Low-Speed transmission rate of 125 kBit/s. CAN3, on the other

hand, is designed as a High-Speed CAN bus operating at 1 Mbit/s, because it networks the motor ECUs.

Remaining bus simulation enables parallel ECU development

During the development process for the ECUs and the software, the participants at the University of Offenburg and its partner Evomotiv were confronted with the problems that typically occur in complex communication structures. Typically, some developments are still in their beginning or prototype stages, while others are already further advanced. However, to test finished systems as realistically as possible developers must rely on the functionality of systems that do not exist yet – at least significant portions of them.

This problem is solved by what is referred to as a remaining bus simulation, in which appropriate software is used for computer simulation of the ECUs that do not exist in real form yet. The systems under test cannot detect any differences between simulations and real ECUs; therefore, full network communication is available. On the Schluckspecht project, CANoe from the company Vector Informatik was used as the standard analysis and simulation tool for this purpose. It is quite easy to create the remaining bus simulation using Windows software as soon as the responsible designers have fully and correctly parameterized the database for the network. The Vector Interaction Layer then ensures that every message is sent with the send type specified for it in the database.

From simulation to the real ECU

At the beginning of the project, the development team used this method to simulate over half of the ECUs of the Schluckspecht. The ECU applications created with CANoe's own CAPL programming language provide the foundation for the remaining bus simulation for great depth of testing, even without a vehicle. To test the

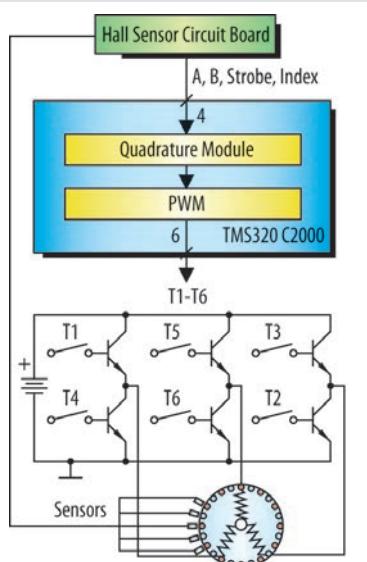


Figure 3: System diagram of motor/ converter bridge/ processor

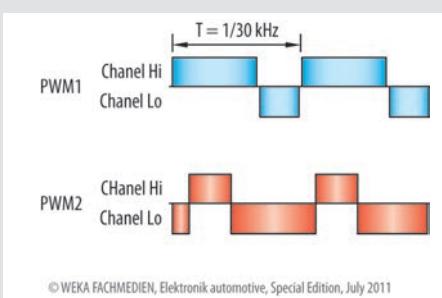


Figure 4: PWM signals for commutation

central gateway, for example, it seemed sensible to implement a special simulation interface. As the individual ECUs are gradually completed, the related part of the remaining bus simulation is simply deactivated. At the end of this process, the entire network is available in real form, and CANoe operates as a pure analysis and monitoring tool (**Figure 7**).

On test drives of the Schluckspecht IV E, the same configuration that served as the basis for the remaining bus simulation in the early development phase later served to monitor processes inside the vehicle, and if necessary to intervene in them. All nodes are deactivated in the Simulation Setup of the CANoe configuration, since all ECUs now exist in real form. On the competition drive in South Africa, engineers and student engineers monitored such parameters as the temperature, voltage and current of the batteries, and they could set different drive torques over the various driving stages. The HMI panel can be used to display and stimulate the brakes and turn indicators.

Remote monitoring of test drives

Another interesting aspect of the Solar Challenge is that all monitoring and control activities are performed from a support vehicle. The challenge here is to transfer data from the on-board vehicle network to the analysis computer in the support vehicle by wireless transmission communication. This is accomplished by special CAN/WLAN interface modules with a range of up to 500 m that effectively mirror the entire CAN traffic of the test vehicle via WLAN to the remote network on the support vehicle. This process is fully transparent to CANoe; the tool can continue to be used to display and evaluate system parameters in the competition vehicle as usual. The time stamps of the CAN messages, which are preserved in the transmission, permit (time-)consistent display on the receiving end. This is also possible over the opposite pathway, and

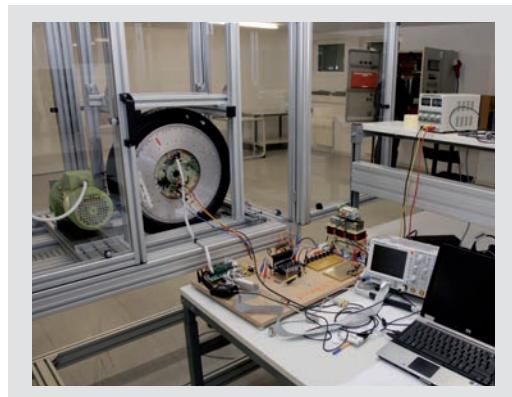


Figure 5: Motor test bench at Evomotiv.

technicians can stimulate the test vehicle's network from the support vehicle. In this case, stimuli from the user control panel in the support vehicle have a higher priority for acting on the system than interventions by the driver. In extreme cases, the vehicle can even be fully operated by remote control.

FlexRay is entering the electric vehicle

Knowledge gained in these competitions quickly flows into advanced developments by industrial partners in the framework of knowledge transfer or into the university's own projects. Evomotiv has long been working together with the scientists from Offenburg on an improved wheel-hub motor for a street version of the electric vehicle. The focus here is on achieving a significant increase in motor power from two kW to 15 kW and four-wheel drive instead of two-wheel drive. For safety-relevant components, such as the brakes, various TÜV approvals are required. Here too, state-of-the-art technologies and innovations are expected to contribute to success. Consideration is also being given to a system with a voltage of 400 V to supply the motors instead of today's 48 V system. Moreover, the use of FlexRay is planned for time-critical communication between ECUs, motors and the brake system and to implement the related control circuits. FlexRay – characterized by such features as high speed, real-time capability and fault tolerance – places significantly higher demands on participants' expertise and on the development and analysis tools that are used. Simulation and analysis systems like CANoe are especially in demand, since they combine high performance with multibus capability, and they can simultaneously process and display both FlexRay and CAN data.

Translation of a German publication in Elektronik automotive, issue July/2011

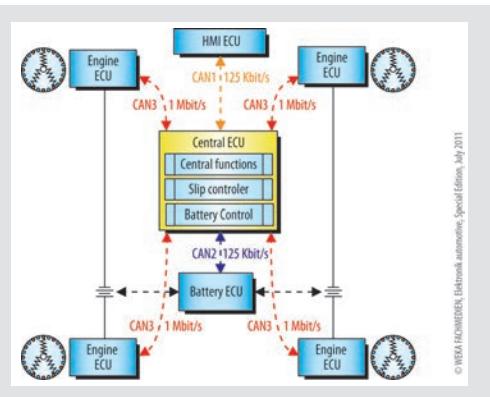


Figure 6: System overview of the full vehicle.

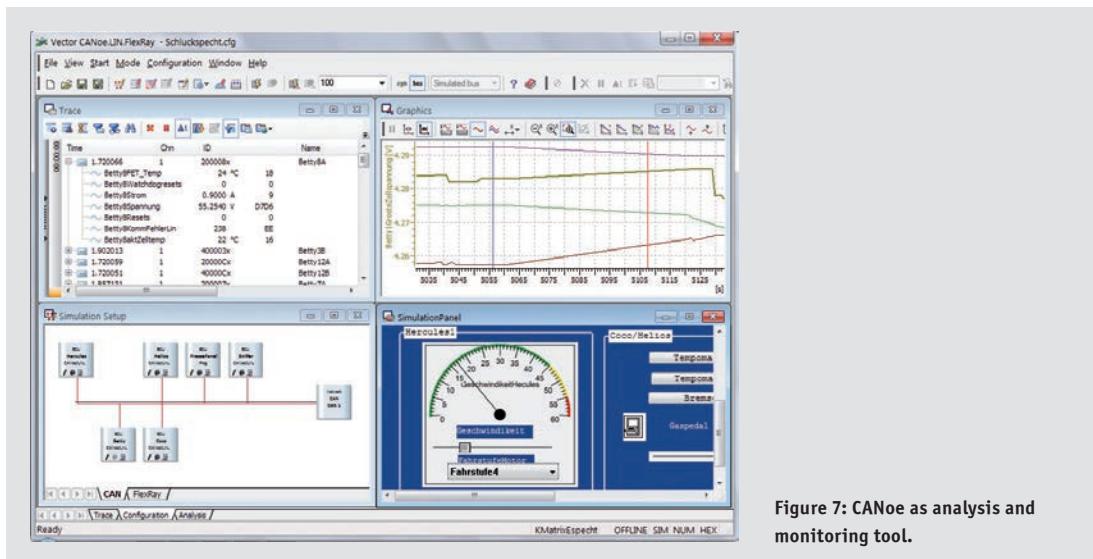


Figure 7: CANoe as analysis and monitoring tool.

Figures:

Offenburg University of Applied Science: initial figure

Evomotiv: figure 1, 2, 3, 4, 5, 6

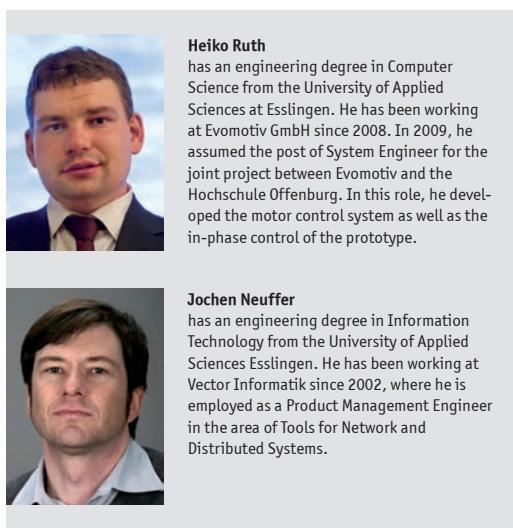
Vector: figure 7

Links:

Homepage Offenburg University of Applied Science: www.fh-offenburg.de

Homepage Evomotiv : www.evomotiv.de

Homepage Vector: www.vector.com





VN8900 – The Future of Network Interfaces

The modular network interface with integrated real-time computer and standalone functionality!

Take advantage of the features:

- > Optimal performance for CANoe/CANalyzer/CANape real-time applications with FlexRay/CAN FD/LIN/J1708/K-Line access and up to 8 channels
- > Minimal latency times and synchronized interfaces
- > Best suited for MiniHIL applications
- > High performance data throughput
- > Integrated interface for analog/digital interfaces
- > Easy to configure via USB plug & play interface
- > Fast SSD memory and rugged keypad for standalone operating mode

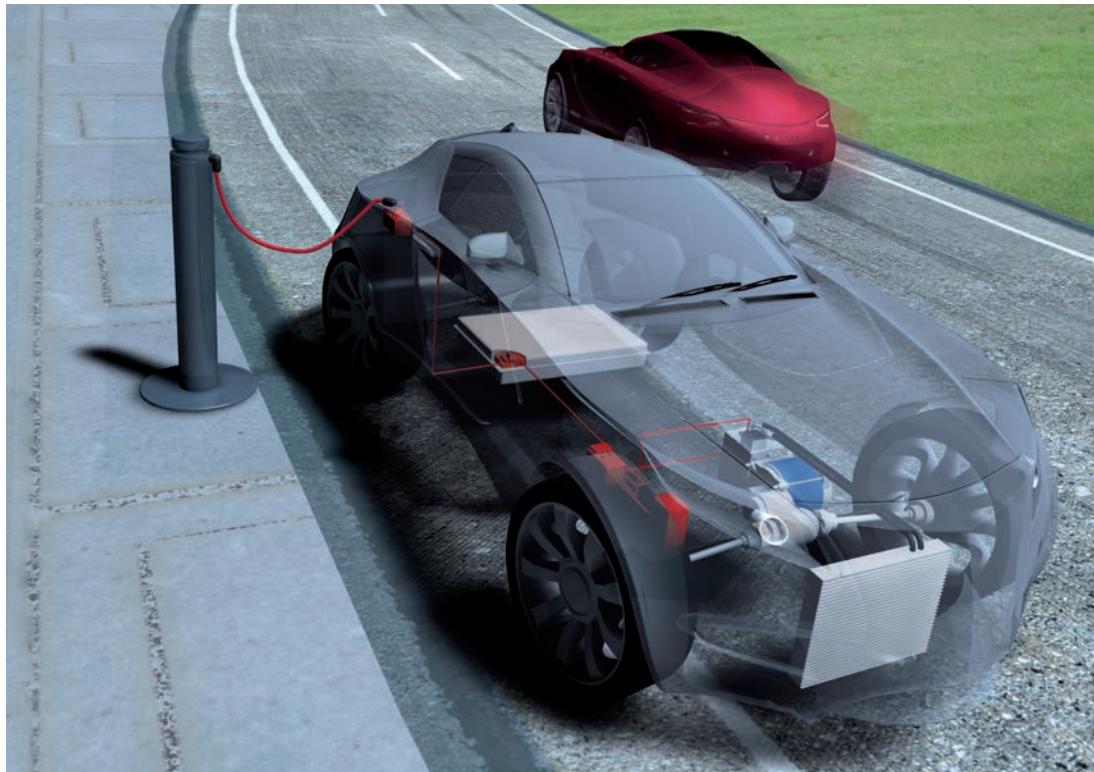
The VN8900 network interface can be used in multiple application areas like system simulations or bypassing applications with Simulink, remaining bus simulations, gateway implementations and test executions (MiniHIL).

► **More information:** www.vector.com/vn8900

► **NEW VN8912 Base Unit:** best applicable at test stands with extensive CANoe Configurations, MATLAB simulations or huge laboratory test environments

Convenient Charging of Electric Vehicles

Smart Charging with MICROSAR IP enables flexible charging processes and easy payment



Compared to conventionally powered vehicles, electric vehicles (EVs) have significantly shorter driving ranges due to the low energy density of their batteries. For EVs to experience a successful market launch, it is important to have a charging infrastructure in place that is widely accessible and easy to use. It is equally as important to have a standardized charging process. This article describes Smart Charging and its standardization in the ISO 15118 standard. With Smart Charging, in addition to a power connection, the vehicle also establishes a communication channel with the charging station. Today, Vector is already providing a first implementation based on its MICROSAR IP communication stack.

Today, charging of EVs generally takes place at home. Just a few charging stations are available in public areas – as part of model studies. Since vehicles are parked frequently, e.g. while shopping or at work, they can be charged during these times. In the future, a broadly based and standardized infrastructure will be built up for this purpose. It has to offer a standardized mechanism for charging the batteries, and it must also support a method for easy payment. To enable convenient payment, debiting of the small amounts should be conveniently handled by automated electronic billing.

International standardization and its distribution

Widespread establishment of a charging infrastructure can only be properly achieved if all aspects of the charging process are standardized across manufacturers. The connector and cable as well as the charging communication must be standardized for all EVs and charging stations. In Europe, charging communication is described in the framework of ISO 15118. In the USA, this is being done in SAE (Figure 1). In Japan, there is already the CHAdeMO standard and a charging station network of over 250 stations.

According to the “National Development Plan for Electromobility” by the German federal government, Germany should

become the lead market for electromobility. This plan calls for one million EVs to be on the roads of Germany by 2020.

Providing the energy

Charging of EVs can cause a severe load of local electrical distribution networks. Today's electrical grids require some time to react to such load changes. If several charging EVs draw high power simultaneously in one location, e.g. in a parking garage, this could lead to a local grid overload and outage.

Until now, no consideration has been given to the total power requirement for charging EVs on the electrical grid. As soon as the driver plugs in the vehicle's charging cable, charging begins at the maximum possible current, and this adds a certain amount of load to the grid. This might appear to be similar to the model of fueling up at a normal fuel station, where energy is always in stock and is easy to obtain in the form of gasoline. But the situation with electrical energy differs fundamentally. It cannot be stored as simple as gasoline and be drawn from storage. Nonetheless, by introducing an intelligent electrical grid (Smart Grid) and by using intelligent charging, it is possible to avoid overload and grid failure. In a Smart Grid, data is exchanged about power requirements, and the electrical grid can be optimized accordingly.

The power needed for a charging operation lies between 3kW and 20kW, or even over 100kW, depending on the available power connection and charging profile. By comparison, a typical citizen in Germany uses an average of 3-5kWh of daily electrical energy, depending on household size. To operate the grid so that it is more stable, the energy provider needs time to supply the charging energy. One way to obtain this time is to delay the start of the charging operation by several tens of seconds.

Charging method for DC or AC power

In charging the batteries, two different procedures can be distinguished. First, the battery can be charged with alternating current, which is available in the electrical grid as single-phase or three-phase AC. Nearly any electrical outlet may be used for charging here. However, the charger must be installed in the vehicle, which means additional weight. In the second variant, the battery is charged with DC electricity. In this case, the charger is located outside of the vehicle, in the charging station, and it generates the DC voltage for charging the batteries. In this case, the weight of the charger does not matter, but costs are higher for such a DC charging station. Since these two charging processes each have their advantages and disadvantages, they are used in parallel.

Communication between vehicle, charging station and energy provider

If the vehicle only has to communicate with the charging station for charging, the choice of transmission medium and protocol would be quite flexible. However, the charging station and vehicle also need to communicate with various servers on the Internet (**Figure 2**). Therefore, it makes sense to use the conventional protocols of IP-based networks. Since requirements call for just using the cable for the charging current – and no auxiliary lines for communication – communication is implemented directly via the charging cable (**Figure 3**). PLC technology (Power Line Communication) is available for this purpose. In this system, the data stream is modulated onto the power line. This system is more familiar under the names Homeplug AV and IP-over-powerline in the consumer products field; they offer a simple way to set up private computer networks via a building's power lines.

Definition of vehicle to grid communication interface			
ISO/OSI Layer	Europe	USA	Content
7 Application	ISO/IEC 15118 Part 1	SAE J2836	General information and use case definition
7 Application 6 Presentation 5 Session 4 Transport 3 Network	ISO/IEC 15118 Part 2	SAE J2847	Technical protocol description and Open Systems Interconnections (OSI) layer requirements
2 Data Link 1 Physical	ISO/IEC 15118 Part 3	SAE J2931	Wired physical and data link layer requirements

Figure 1: Charging communication is defined in ISO and SAE standards.

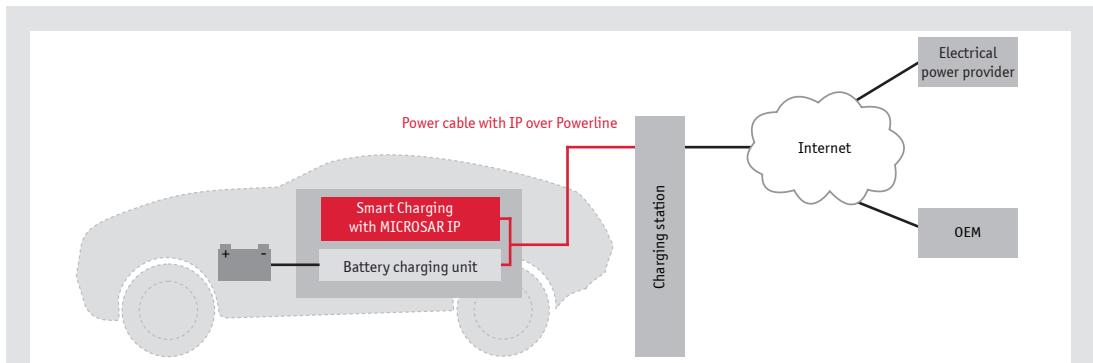


Figure 2: The vehicle uses the Internet Protocol to communicate with the charging station and servers on the Internet

In the vehicle's charge control module, a TCP/IP stack is used for communication. It provides socket communication over IP (Internet Protocol) for TCP (Transmission Control Protocol) or UDP (User Datagram Protocol). Overlaid on this protocol are several applications and protocols such as:

- > DNS (Domain Name System) for name resolution
- > TLS (Transport Layer Security) for encrypting the data on the transport level
- > V2GTP (Vehicle To Grid Transport Protocol), a new protocol for connection monitoring and data transfer

Furthermore, a module is needed that contains the Smart Charging functionalities from ISO 15118. For this purpose, Vector has developed its own standard module: SCC (Smart Charge Communication.) It establishes the connection to the charging station and exchanges parameters on the charging process over the connection. The module can be adapted to the requirements of specific carmakers and suppliers by its many different configuration options.

Currently, data transmission over the Internet is generally still performed using the IPv4 protocol. To circumvent the growing scarcity of addresses on the Internet, ISO requires that vehicles and charging stations support IPv6. Overall, a system is created in the vehicle that is very complex and resource-intensive, but it is also very powerful.

Vector already offers an implementation based on the MICROSAR IP stack (**Figure 4**); it was specially optimized for use in motor vehicles and conforms to the AUTOSAR standard. In addition to the TCP/IP stack, the charger requires a CAN stack to connect to the existing vehicle network. Communication with energy management and the user terminal are implemented via the CAN stack (**Figure 5**).

Procedure for a charging operation

On current EVs, the charging process is simple: the user simply plugs a connector into the charging station, and the charging process starts right away. In Smart Charging per ISO 15118, the



Figure 3: Example of a charge plug as proposed for standardization.

charging process is more complex. After plugging in the charging cable, the vehicle first establishes a connection with the charging station via PLC for communication. Then the vehicle obtains an IP address over DHCP, after which the SCC module queries the IP address of the charging station via a broadcast message (ChargePointDiscovery). Now the vehicle establishes a TCP connection and, overlaid on this, a TLS connection over which both the charging station and the vehicle are authenticated by certificates. Data such as service information, rate tables and charging profiles is exchanged and selected over this encrypted connection, and payment modalities are set. Now the cable is physically locked, so that it cannot be pulled during the charging process – as to prevent theft of the electrical power. Finally, the charging station switches the power on, which starts the actual charging process. During this process, the vehicle and charging station regularly exchange status information and electrical meter readings, and the vehicle acknowledges the reception of energy. During charging, the vehicle may be placed in a quiescent state to reduce its own energy consumption. It periodically wakes up from this state to execute a status update. The charging itself continues without stop. When charging is finished, the charging station shuts off the electrical power and unlocks the plug connection. The last acknowledged meter reading is transmitted to the energy provider over the Internet for billing.

Paying by Micropayment

As described at the beginning of this article, EVs have a short range due to the limited energy storage capacity of the batteries. To be equipped for unplanned or longer drives despite this limitation, efforts are made to charge the batteries as often as possible and/or in a short period of time. For example, full charging of a typical 20kWh battery will cost between 3 and 10 Euros, depending on the different rates. Often the amount is significantly less than this, because the battery is seldom entirely depleted and does not need a complete charge. So, a simple payment system is needed that avoids the need to make small individual payments for several short charging operations.

In principle, there are a number of different possibilities for paying for the electrical charges: cash payment, payment by card and PIN or an automated billing method. This might be based on electronic authentication and a suitable billing contract with an energy provider – similar to a contract for a mobile telephone. The latter makes the most sense for the small and unequal sums. As a side benefit, this also reduces the risk of vandalism to charging stations in public spaces, because the charging station only requires a plug outlet and a small display.

The price for charging an EV cannot be given as a lump sum; price tables are communicated to the vehicle to calculate the price. The combination of the price table, the charging profile and the available power at the charging station yields a number of variants for pricing

and charging duration. The selected variant is based on pre-configuration in the vehicle, so that charging can be started automatically.

Outlook

Certain aspects of intelligent charging of electric vehicles may still seem like a dream. However, the foundations are already being laid today, e.g. in the framework of ISO 15118. Release of the ISO 15118 standard is scheduled for mid-2012, with the goal of further expanding its range of application.

Extensions of the AUTOSAR-based basic software are also in planning or discussion. Vector is working on advanced developments as an active participant in ISO and AUTOSAR standardization committees to ensure that it can offer production-ready customer solutions in a timely manner.

Translation of a German publication in
Elektronik automotive, 07/2011

Figure 3: MENNEKES Elektrotechnik GmbH & Co. KG, 57399 Kirchhundem, Germany
All other Figures: Vector Informatik GmbH

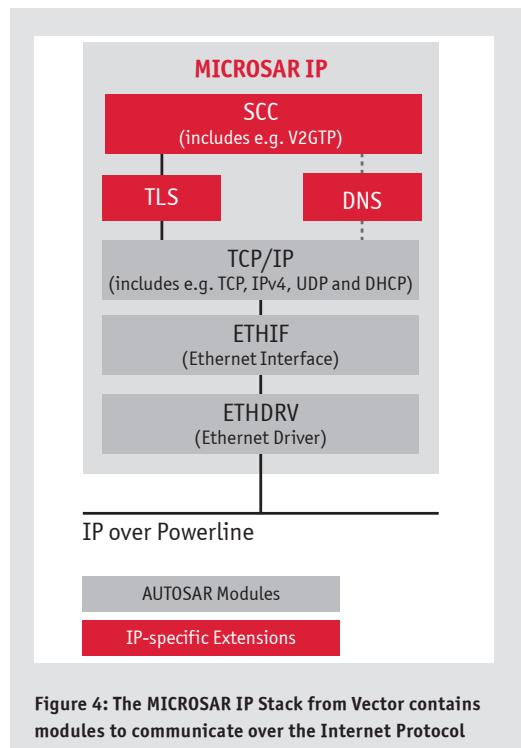
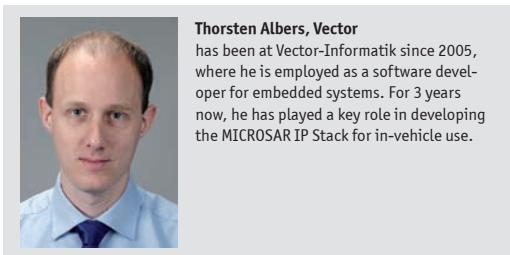
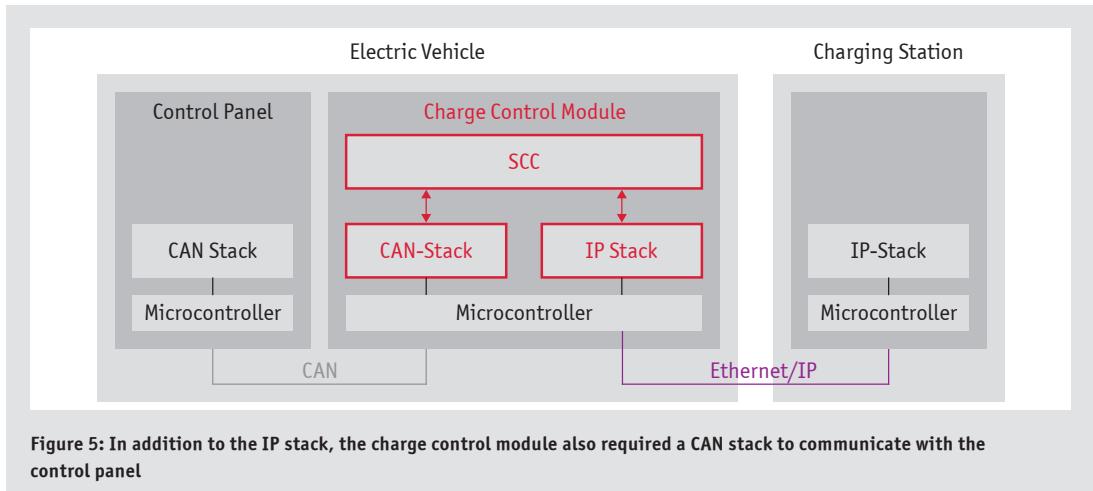
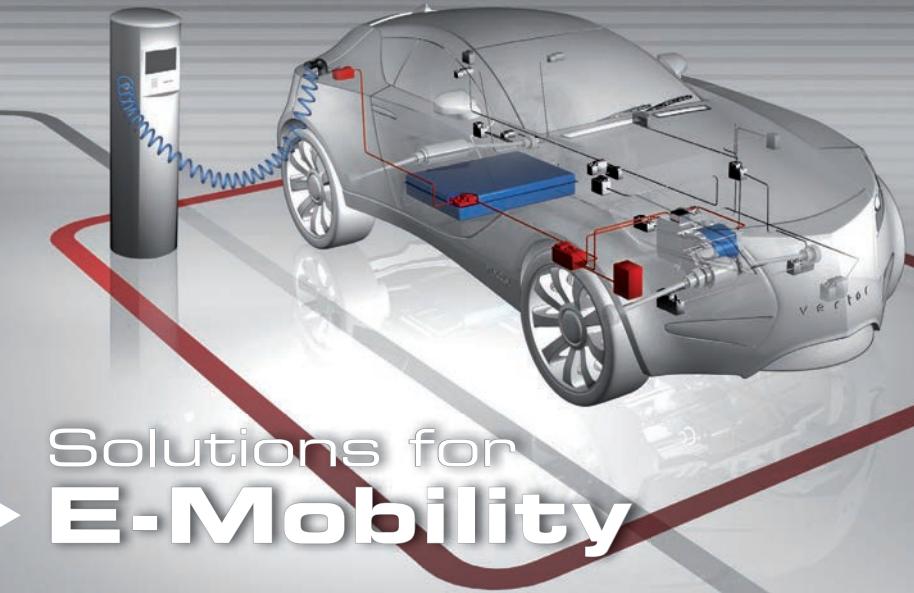


Figure 4: The MICROSAR IP Stack from Vector contains modules to communicate over the Internet Protocol





Solutions for E-Mobility



Vector works with you on future E-Mobility

Benefits from our solutions for various electric drives, battery systems and intelligent charging infrastructures:

- > **AUTOSAR software** for energy-efficient ECUs
- > Software for intelligent **Smart Charging**
- > Products for **measurement & calibration** with high sampling rates
- > Highly developed and proven **simulation** and **test tools**
- > **Services** ranging from requirements analysis to system testing

Vector - solutions for networked systems in the automobile.

► More information: www.vector.com/e-mobility

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

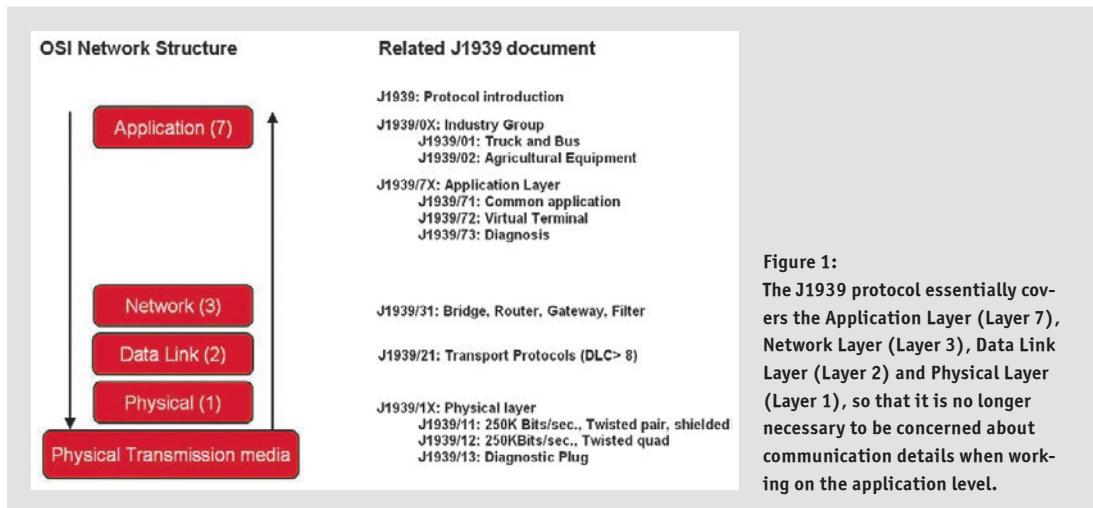


Figure 1:

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.

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 to the Data Link layer, etc.

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.

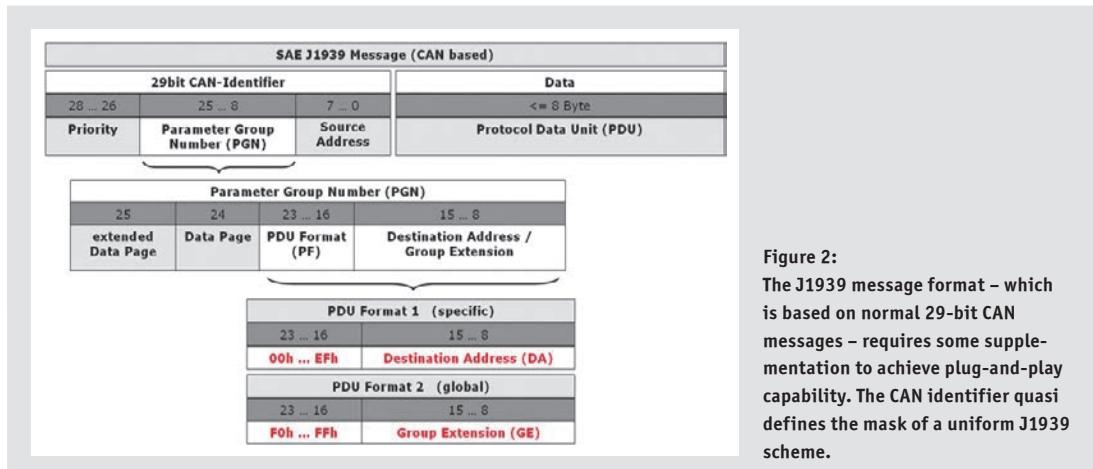


Figure 2:

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.

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 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

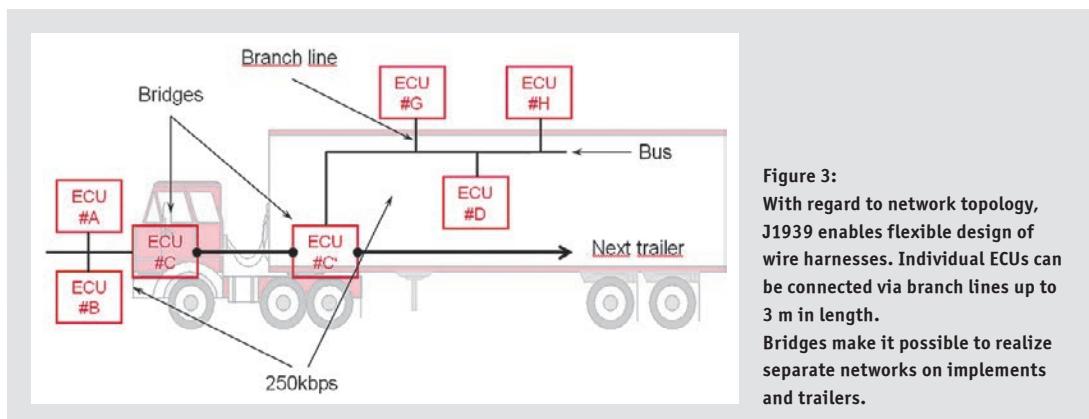
address; they are used for device configuration or ECU commands, for example. Broadcast messages (1:n), on the other hand, are simultaneously addressed to all bus nodes, which is practical when it comes to sending out measured values, error handling and diagnostic purposes.

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.

Testing and Diagnostics of J1939 Components and Systems

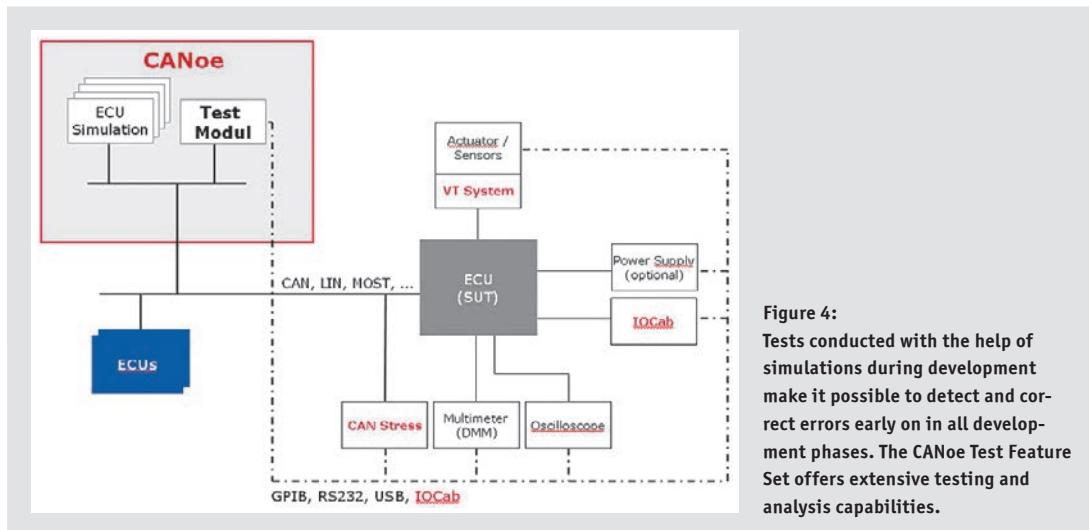
In view of the rising number of J1939 ECUs and the fact that software solutions in heavy-duty vehicles are becoming increasingly complex, a systematic strategy for testing and diagnostics also continues to gain in importance in the J1939 field. Tests are indispensable in all development phases, from 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.

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

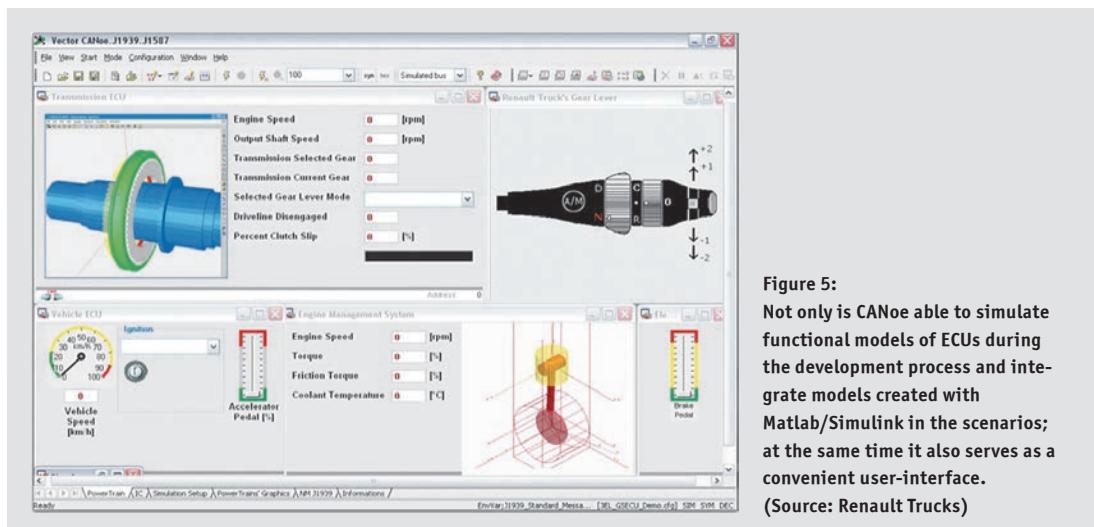


Figure 5:
Not only is CANoe able to simulate functional models of ECUs during the development process and integrate models created with Matlab/Simulink in the scenarios; at the same time it also serves as a convenient user-interface.
(Source: Renault Trucks)

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.

Translation of a German publication in Elektronik automotive, 6/2008

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.

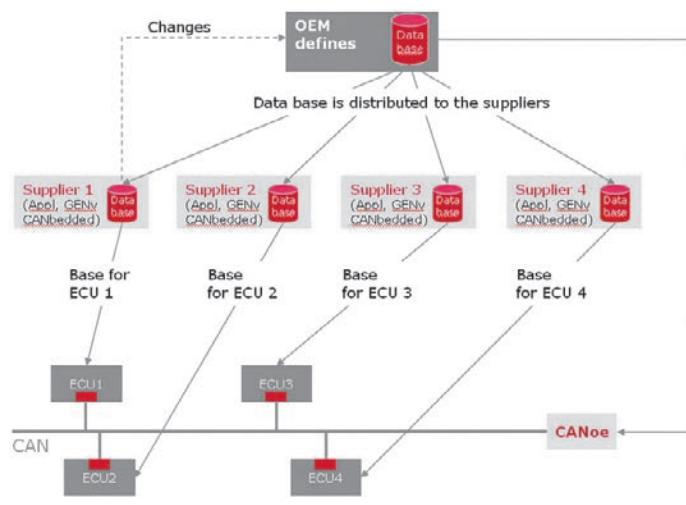


Figure 6:
Standard software components of the CANbedded J1939 package lead to quick development results without developers needing to be concerned with all details of the J1939 standard. A centrally managed database avoids duplicated developments and enables optimal work distribution.

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 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

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 kbit/s, Twisted Shielded Pair	Released status September 2006	Regularly scheduled revision every 5 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 kbit/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)

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.

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.

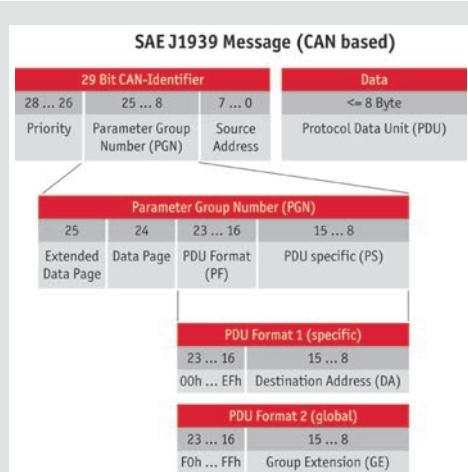


Figure 2: Layout of a 29-bit identifier in J1939 networks.

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.

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

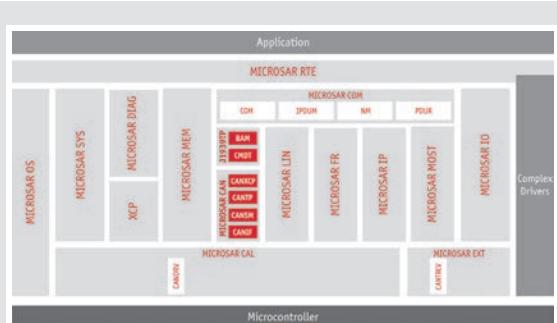


Figure 3: AUTOSAR basic software from Vector contains the two J1939 transport protocols BAM and CMDT.

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.

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.

Translation of a German publication in Elektronik automotive, 12/2010

All Figures
Vector Informatik GmbH

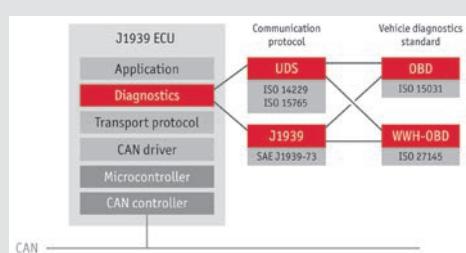


Figure 5: On-board diagnostics in commercial vehicles is implemented by CAN-based protocols.

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.

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.

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:

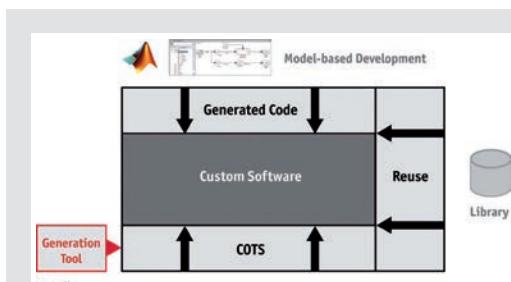


Figure 1:
development models in contrast (green, blue) and a potential merging (red).

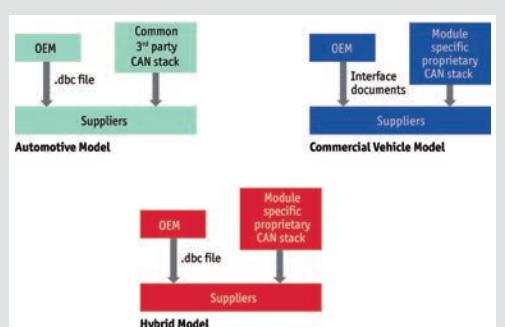


Figure 2:
Use of standard software components can reduce the share of application-specific software.

- > The communication description should be made with an OEM-independent 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.
- > 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

Figures:

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.

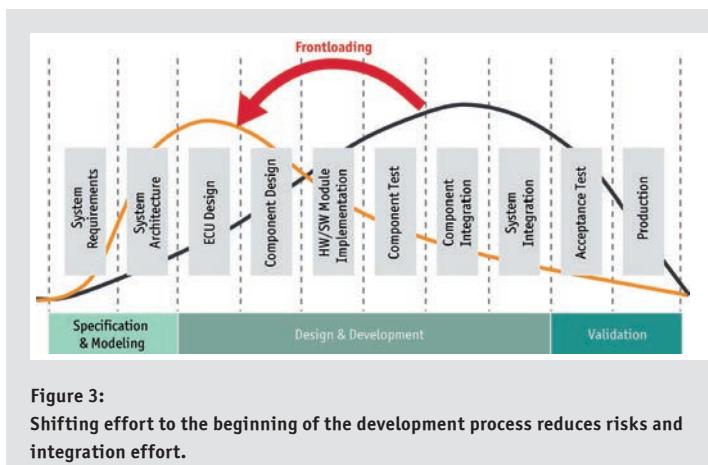


Figure 3:
Shifting effort to the beginning of the development process reduces risks and integration effort.

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 use on mountains and snow, there are also PistenBully versions without a load-



Figure 1:
Up to 620 PistenBullys are produced in Laupheim every year

ing 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

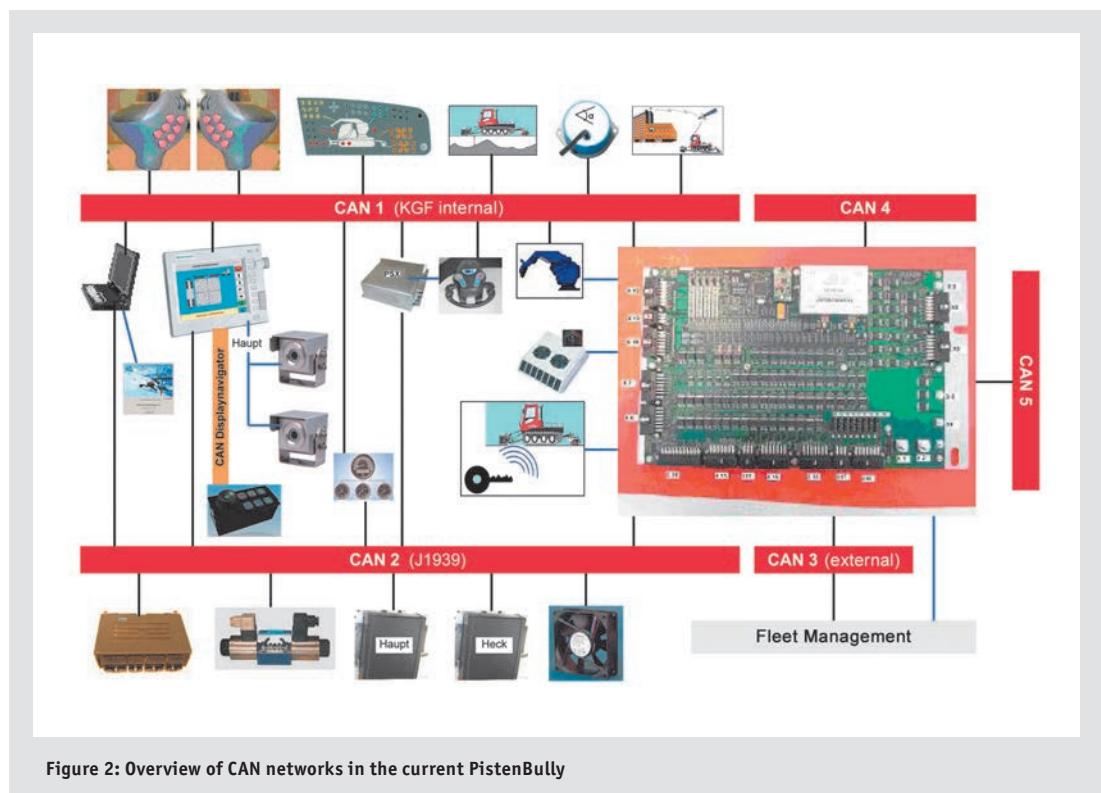


Figure 2: Overview of CAN networks in the current PistenBully

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 electronics 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.

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 in-



Figure 3:
Controls and display components in the PistenBully cockpit

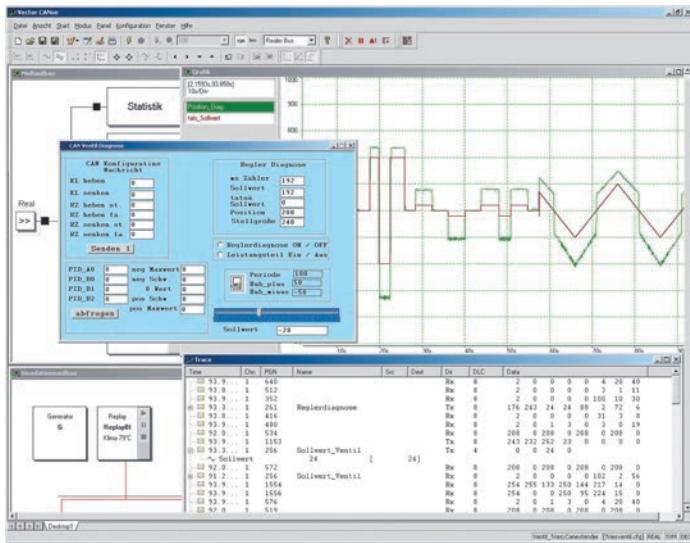


Figure 4:
CANoe as joystick simulator for testing hydraulic valves

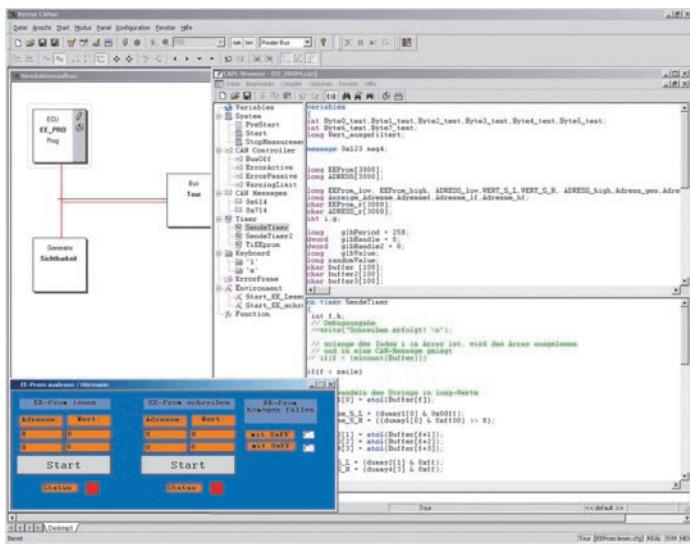


Figure 5:
CANoe in flash mode

tuitive 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 Calibration 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

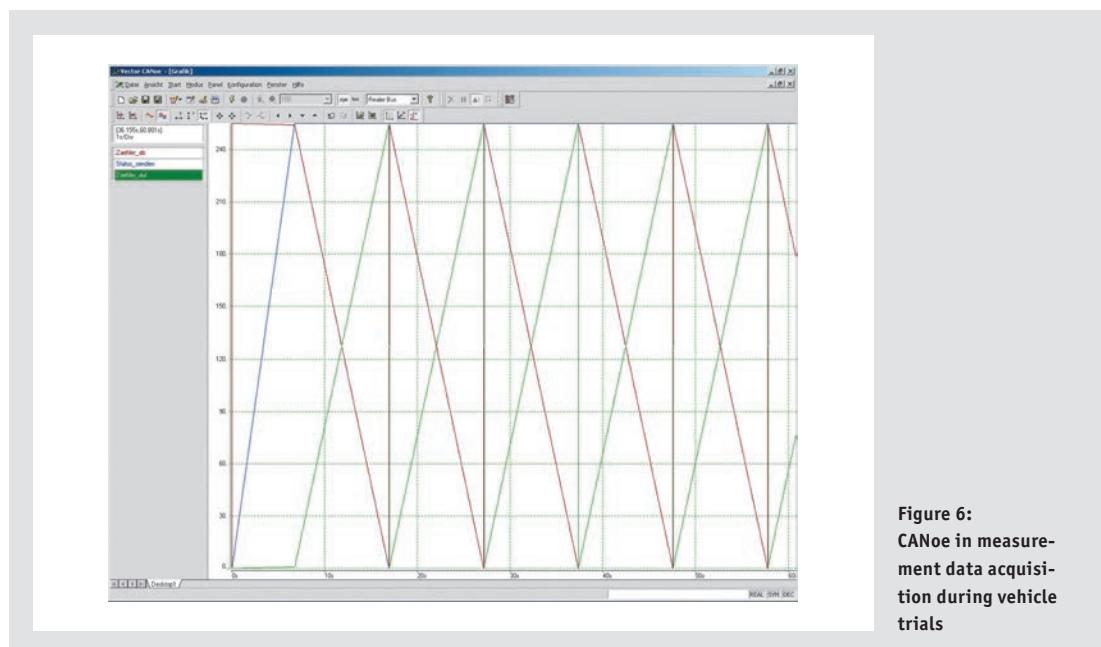


Figure 6:
CANoe in measurement data acquisition during vehicle trials

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.

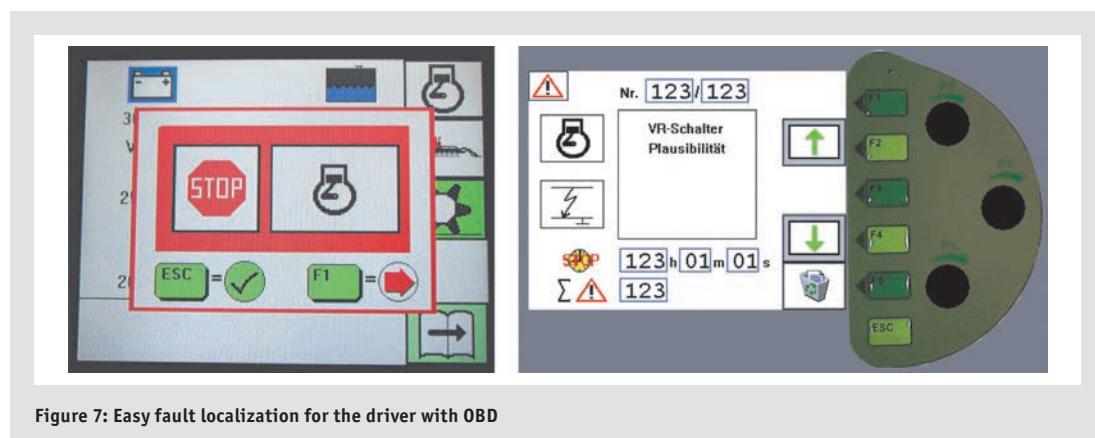


Figure 7: Easy fault localization for the driver with OBD

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.

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 pro-

gramming 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.

Translation of a German publication in Elektronik automotive, 5/2006

All figures: Kässbohrer Geländefahrzeug AG

**Authors:**

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.

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.

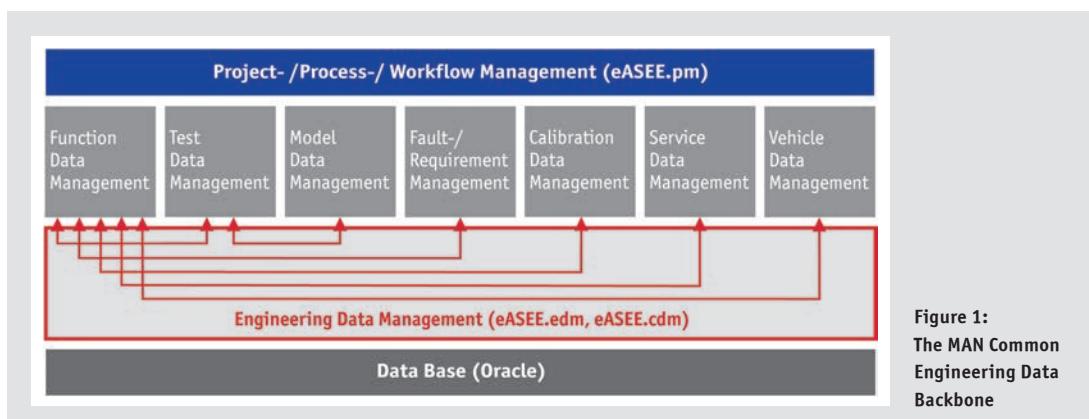
System design is becoming increasingly complex due to growing requirements for networking. Individual ECUs are being installed on different platforms and different countries, which increases the number of variants. This requires flexibility in communication structures and in mapping functions to ECUs. 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.

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.

Translation of a German publication in Hanser Automotive, 9/2008

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.

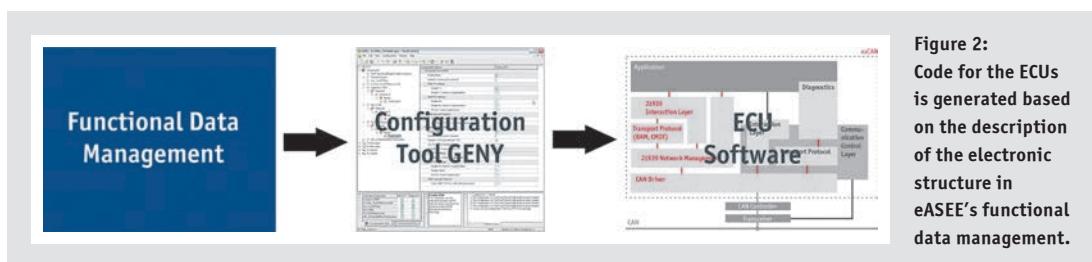


Figure 2:
Code for the ECUs is generated based on the description of the electronic structure in eASEE's functional data management.

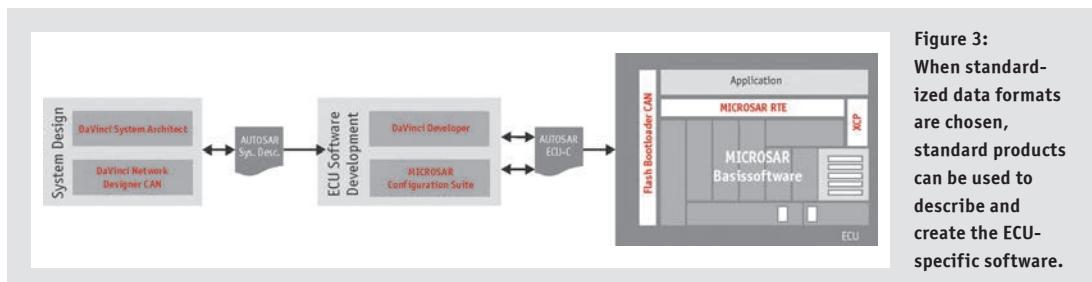


Figure 3:
When standardized data formats are chosen, standard products can be used to describe and create the ECU-specific software.



SOLUTIONS FOR Commercial Vehicles

The reliable, efficient way to design and test networks

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, CANopen and AUTOSAR protocols
- > Benefit from efficient configuration and extensive diagnostic, testing and analysis solutions

The seamless interaction of Vector tools and Vector's comprehensive services will improve the efficiency of your entire development process, from design to analysis.

► Information and Downloads: www.j1939-solutions.com

► Conformance Testing
CANoe J1939 supports SAE
J1939-82 Compliance Tests

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 enables efficient and reliable localization of the real causes of errors and ideally to correct them right away.

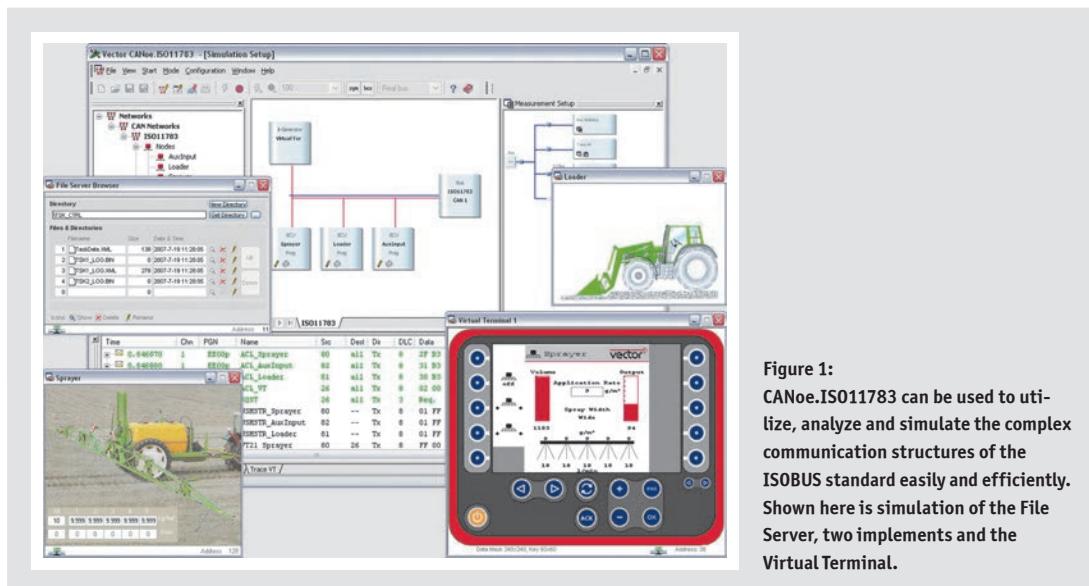


Figure 1:
CANoe.ISO11783 can be used to utilize, analyze and simulate the complex communication structures of the ISOBUS standard easily and efficiently. Shown here is simulation of the File Server, two implements and the Virtual Terminal.

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

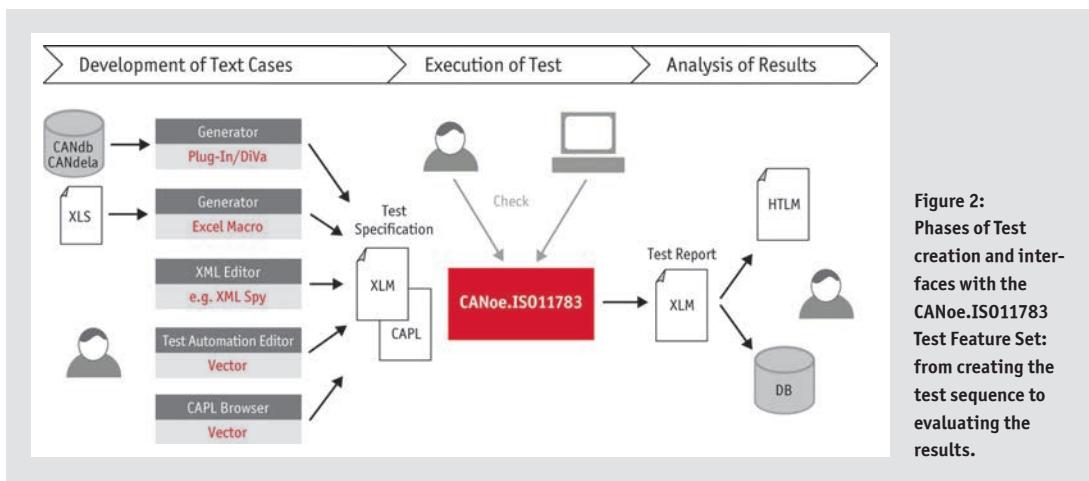


Figure 2:
Phases of Test
creation and inter-
faces with the
CANoe.ISO11783
Test Feature Set:
from creating the
test sequence to
evaluating the
results.

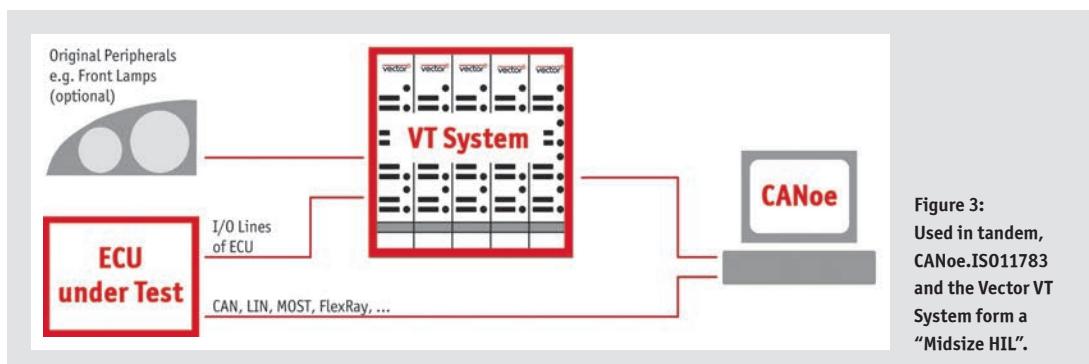


Figure 3:
Used in tandem,
CANoe.ISO11783
and the Vector VT
System form a
"Midsize HIL".

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.

**Translation of a German publication in
Elektronik automotive, 10/2009**

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/CDIS 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 to 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 indispensable 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



Figure 1:
John Deere ISOBUS operator's terminal with the
Kverneland fertilizer spreader operating interface

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 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



Figure 2:
Simulation of tractor and fertilizer spreader in CANoe.ISO11783

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.

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 ISO-BUS 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,

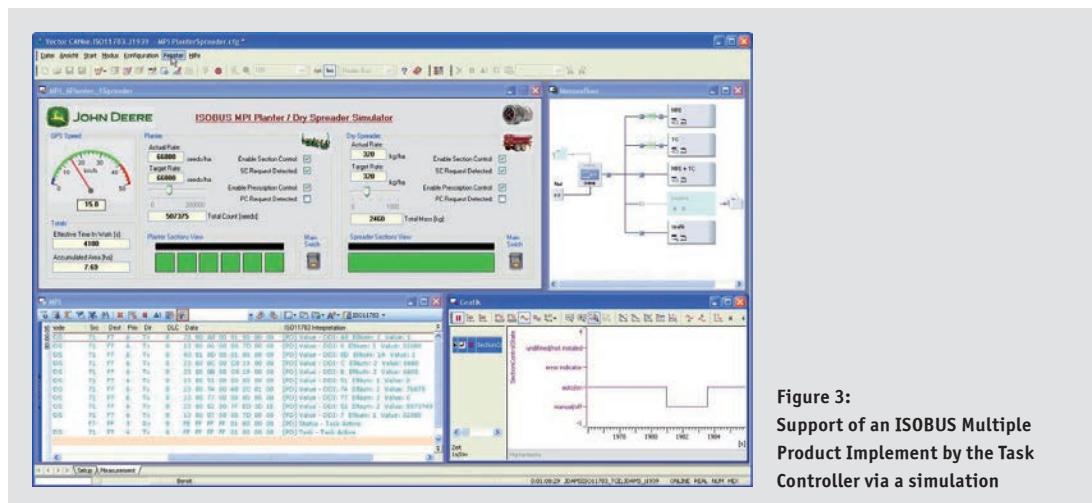


Figure 3:
Support of an ISOBUS Multiple Product Implement by the Task Controller via a simulation

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.

Translation of a German publication in

Elektronik automotive, 11/2011

Figures:

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 Deere'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 custom-
er-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.



Solutions for Open Networks from one Source

Open CAN-based protocols are the basis of networking in commercial vehicles, avionics and industrial control technology. Vector supports you in all development phases of these systems:

- > Systematic network design with CANoe, ProCANopen and CANeds
 - > Successful implementation with source code for CANopen, J1939 and more
 - > Efficient configuration, test and extensive analysis with ProCANopen, CANoe and CANalyzer
- Multifaceted trainings and individual consulting complete our extensive offerings.

Thanks to the close interlocking of the Vector tools and the competent support, you will increase the efficiency of your entire development process from design to testing.

- ▶ Further information, application notes and demos:
www.vector.com/opennetworks

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.

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,

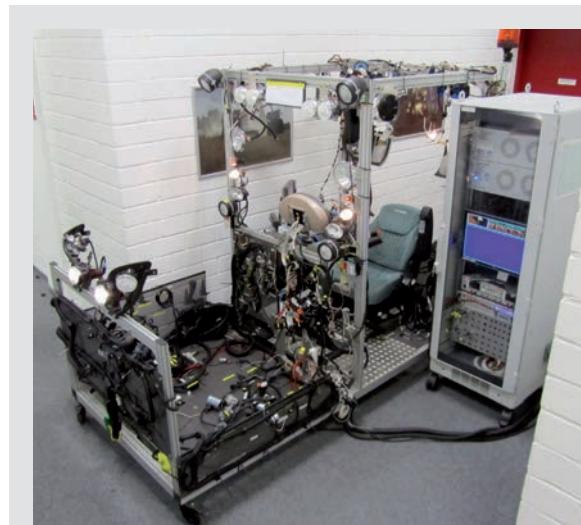


Figure 1: The breadboard fixture of the tractor contains all electronic components of the real tractor model. Automated tests can be executed to real-time conditions with CANoe.ISO11783 and the RT target.

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

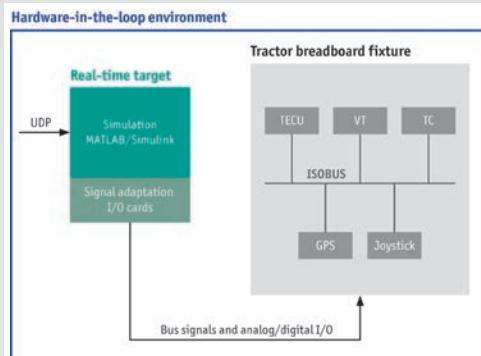


Figure 2: The original HIL system is composed of the real-time target, tractor breadboard fixture and UDP interface for communication with a remaining bus system. The armrest with integrated user controls is implemented as a real component, for example, and must therefore be operated manually.

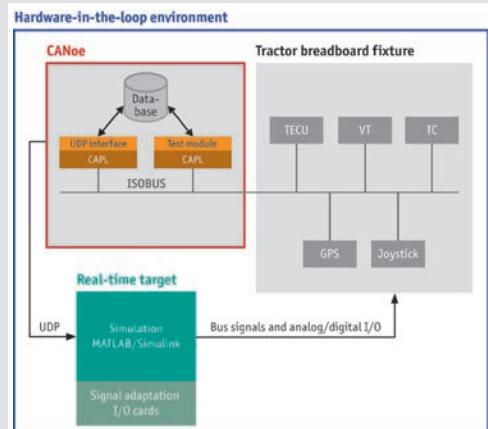


Figure 3: Integration of the CANoe interface in the existing overall system: The test module controls the existing test environment. The armrest, with its integrated controls, is implemented as a virtual component in the test system and can therefore be operated either manually or automated by the test program.

functionality in test modules, using either the C-like script language CAPL or the XML standard.

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, Giga-tronik 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,



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.

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. 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**

Figure Credits

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

**>> Contact information for all locations of the Vector Group:
www.vector.com/contact**



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.

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 indispensable 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.

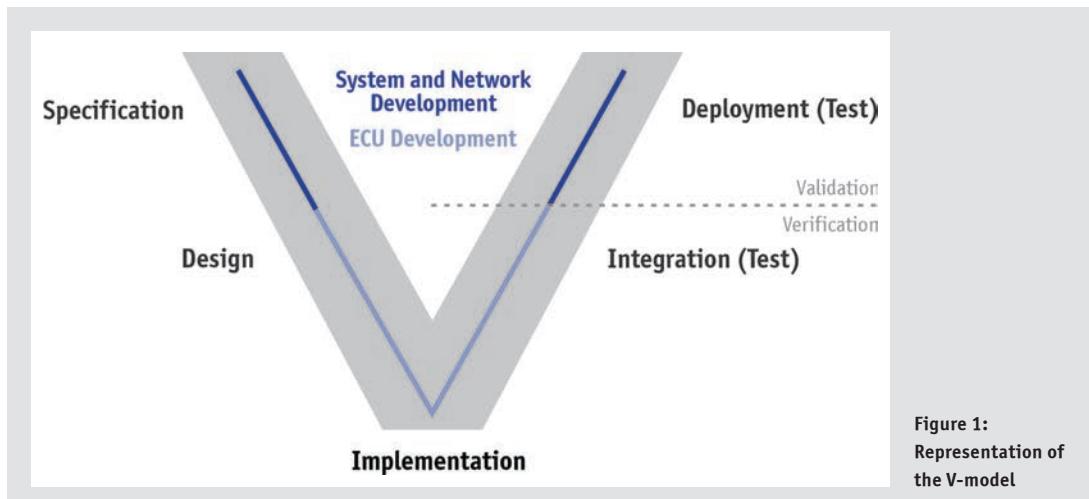
A way out of this dilemma might be offered by a software tool that could be used to create a prototype of the future total system in a simple way. Optimally, this tool would also offer extensive test functions.

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.

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**).

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.

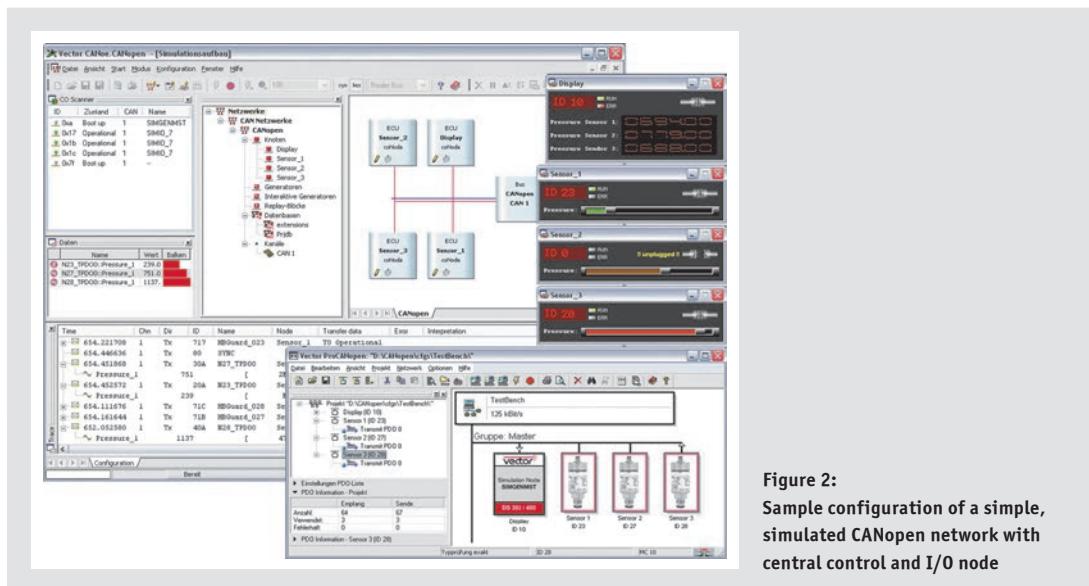


Figure 2:
Sample configuration of a simple, simulated CANopen network with central control and I/O node

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.

Translation of a German publication in IEE, 3/2008

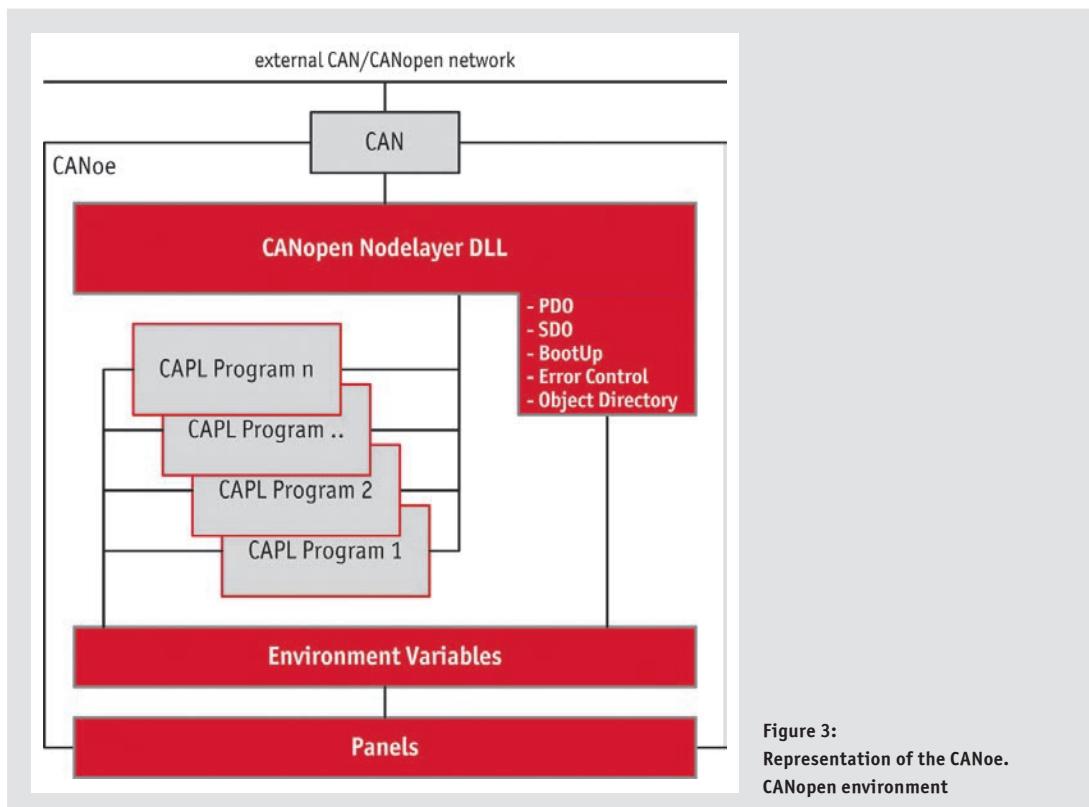


Figure 3:
Representation of the CANoe.
CANopen environment



Mirko Tischer,
Product Manager for CANopen.

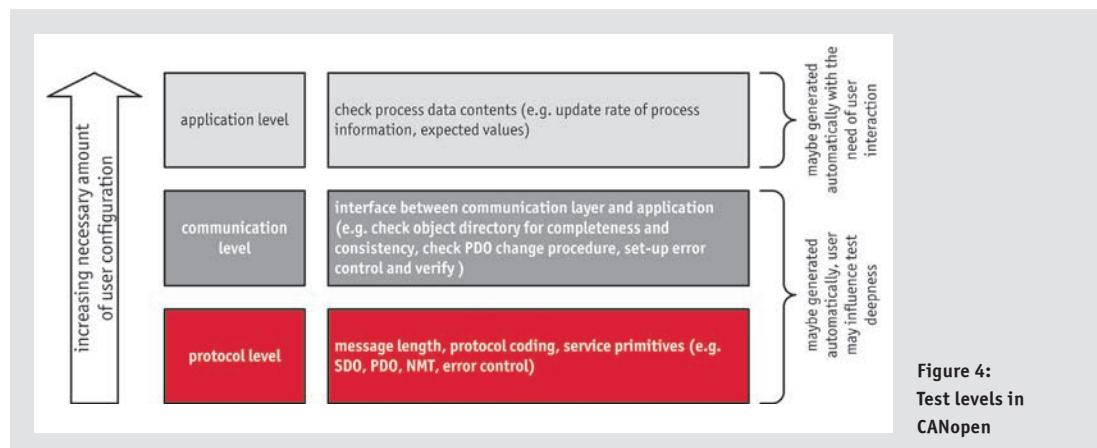


Figure 4:
Test levels in
CANopen

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

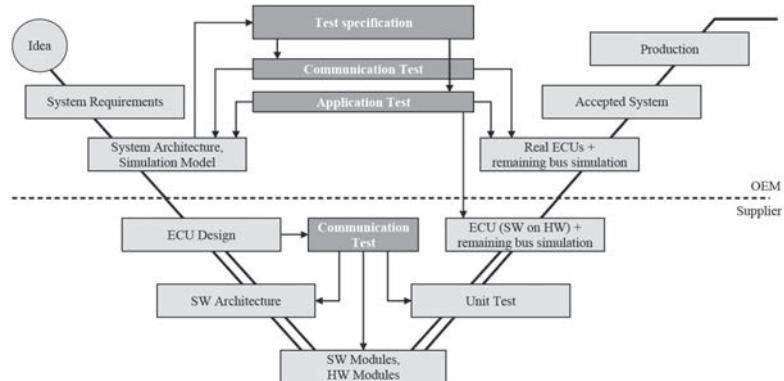


Fig. 1: Development process of a CANopen system

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 that checks all

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 overall system. This can signifi-

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 device producers often ne-

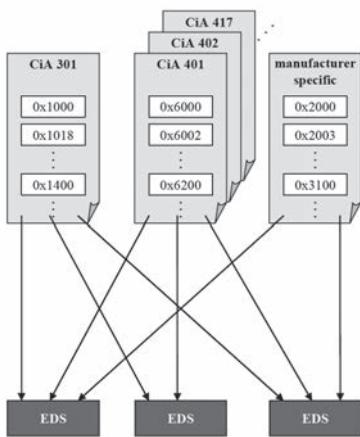


Fig. 2: Device functionality

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 suit-

able 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 ▶

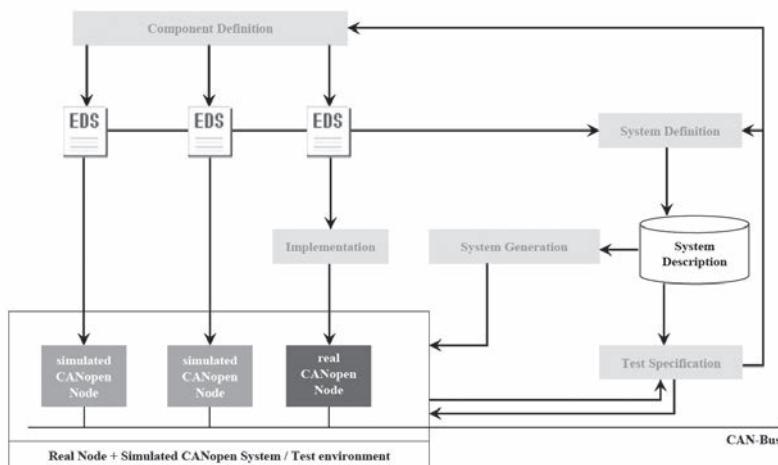


Fig. 3: Test sequences of a CANopen system

process for generating test sequences must be repeatable as often as needed. Changes to the device design can affect the device

es receive or send them. Among other things, precisely these aspects must be tested. This subject matter can be explained by the

Attribute	Value	
Index	60B3 _b	
Name	GPS date	
Object code	Variable	
Data type	Unsigned32	
Category	See /CiA447-2/	

31	26	25	20	19	18	15	0
r		GPS date day	GPS date month	GPS date year			

Fields	Value	Definition	Unit
GPS date year	0000 _b , FFFD _b , FFFE _b , FFFF _b	Minimum value Maximum value Failure Signal not available	Years
GPS date month	00 _b , 01 _b , 0C _b , 0D _b , 0E _b , 0F _b	Reserved Minimum value (January) Maximum value (December) Reserved Failure Signal not available	Months
GPS date day	00 _b , 01 _b , 1F _b , 20 _b to 3D _b , 3E _b , 3F _b	Reserved Minimum value (1 st) Maximum value (31 st) Reserved Failure Signal not available	Days
r	11 1111 _b	Reserved	

Fig. 4: "GPS date" object description

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.

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 devic-

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 ►

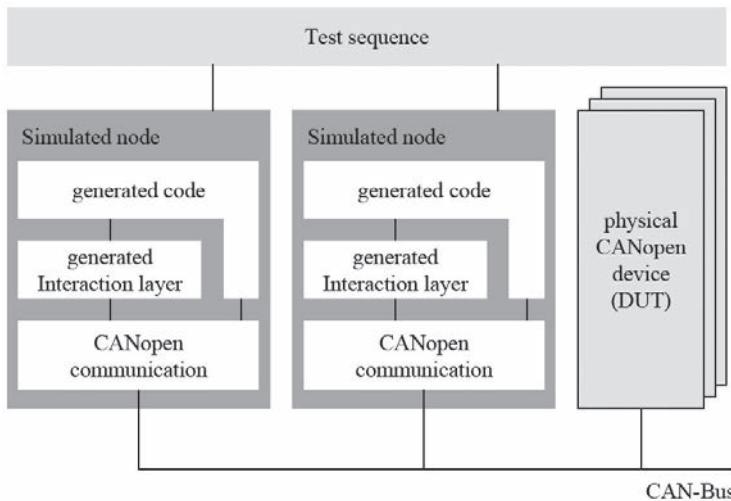


Fig. 5: Generated test environment

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 gener-

ation 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 stimulat-

ed 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

Improving Electronic Engineering Efficiency with Automated Processes

Rising cost pressure is forcing manufacturers and their suppliers to jointly and consistently master product development. Our case study shows how Volvo over time is achieving effective interaction of engineering processes, tools and people on the basis of product and application life-cycle management (PLM/ALM). 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 processes 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 – specifically in times where cost are to be cut and performance must substantially grow.

1 Improving Efficiency with Better Engineering Processes

The current automotive crisis demands cost reduction all along the product creation – including pre-development and product engineering. OEMs and suppliers both struggle to invest now in their engineering in order to fill the pipeline with successful cars some years down the road, while the available capital and resources are very much restricted. The challenge is to develop successful products under the pressure of decreasing development budgets. Specifically in areas where engineering and collaboration processes are only partly established, such as in the engineering of electronic systems, there is substantial room to improve efficiency. When done intelligently such improvements will sustainably cut cost all along the product life-cycle because rework will be reduced and changes are easier to implement.

There are numerous levers for engineering efficiency improvement. Many automotive companies, both OEMs and suppliers, 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. 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 E/E systems development.

We will show with this article how engineering processes can be improved and automated, thus enhancing efficiency, quality and lead time. Such changes need leadership and good orchestration to be successful. We therefore show how a sustainable change process is successfully managed together with impacted engineers and how the concepts can be transferred to other companies.

2 Product Life-Cycle Management: Process, Persons and Tools

Product Life-Cycle Management (PLM) 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

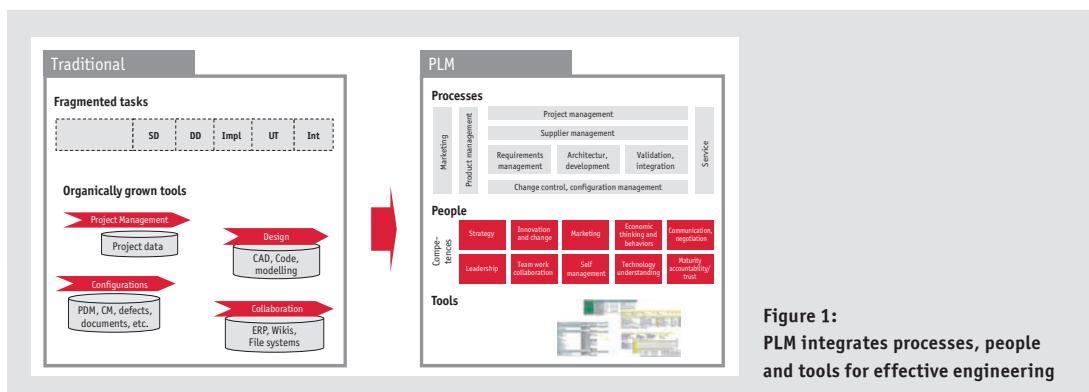


Figure 1:
PLM integrates processes, people and tools for effective engineering

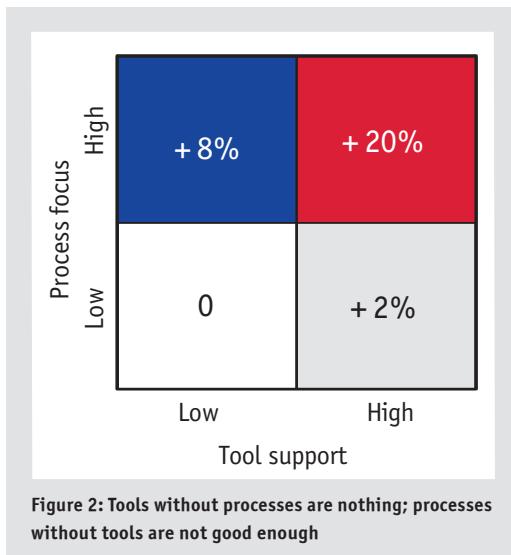


Figure 2: Tools without processes are nothing; processes without tools are not good enough

the business of the enterprise and its customers and partners. It combines people, processes and tools for the effective engineering of products. PLM is only cost-effective if there are the right engineering processes and if they are automated and instrumented with appropriate tools [1,2]. Sustainable cost reduction is achieved when people, processes and tools harmonize instead of the typical fragmented and isolated silos with rework at each interface.

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 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 needs both process and tools support as indicated by **Figure 2** based on a study at London Business School [1,2]. 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 an E/E-PLM 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 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.

One example of an E/E-PLM tool is the “eASEE Automotive Solution” – a complete tool suite of integrated data and process

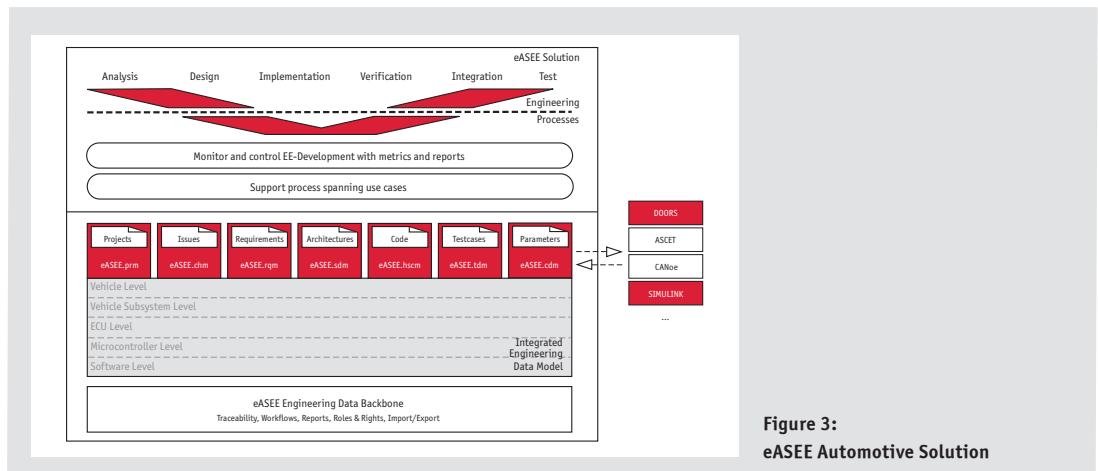


Figure 3:
eASEE Automotive Solution

management modules [3] (**Figure 3**). Based on a rich and extendable data model for features representing the logical and the physical system architecture and the software architecture, a set of eASEE modules can be used for 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.

Being able to not only reuse information but also guide engineers through complex tasks generates immediate returns by making engineers more flexible and avoiding errors, specifically during last-minute changes and corrections under time pressure. Or, consider the time and effort necessary to move engineers from one project to another. Having standardized PLM solutions around a standard product life-cycle reduces the learning curve to allow focusing on real technical challenges instead of organization overhead.

3 Managing the Change

With current cost pressure PLM and new tools are introduced to automate engineering workflows. However, the expected benefits are often not visible. Instead, employees are frustrated and continue working with their current work practices. The new 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. To manage such change and to ensure that impacted

engineers not only pay lip service but actively support and buy into the new processes and tools, their needs and typical work flows must be understood to avoid that process overheads and heavy tools solutions hinder their creativity. This implies pro-active preparation way before a tool decision is made and good leadership, coaching and support through pilots and roll-out.

To adequately support the change process during PLM introduction, Vector has developed over the years a change methodology that is adapted to specific business goals (**Figure 4**).

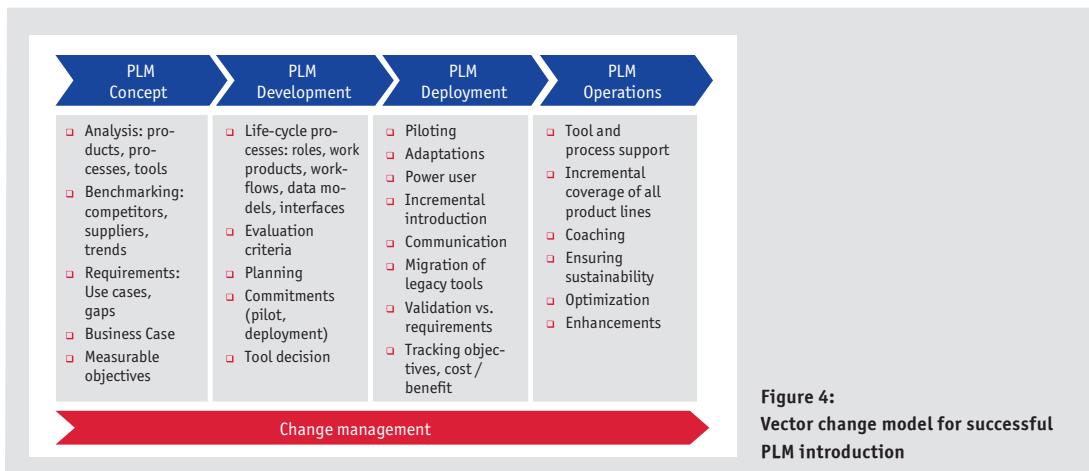
A clear and objective-driven improvement strategy driven by business needs, such as cycle time reduction, reuse with shared platforms, less requirements changes and reduced rework ensures that process changes and tool support indeed translate to efficiency improvement. Such change process though it might look obvious typically needs an external catalyst in order to be followed through and thus avoiding that in the midst of such project, engineers are overwhelmed by the amount of small yet severe changes that impact their daily operational work.

4 Experiences with eASEE-based PLM at Volvo

Current industrial experiences with an E/E-PLM implementation on the OEM side are shown by the ELEKTRA project at Volvo Car Corporation (VCC). We followed above mentioned change process (**Figure 4**) and thus use the same headlines. PLM deployment and PLM operations are combined in ch. 4.3 due to the ongoing deployment activities.

4.1 PLM Concept

Initial Situation: The main drawbacks of the initial situation at VCC were non-connected tool chains and a document driven subsystem



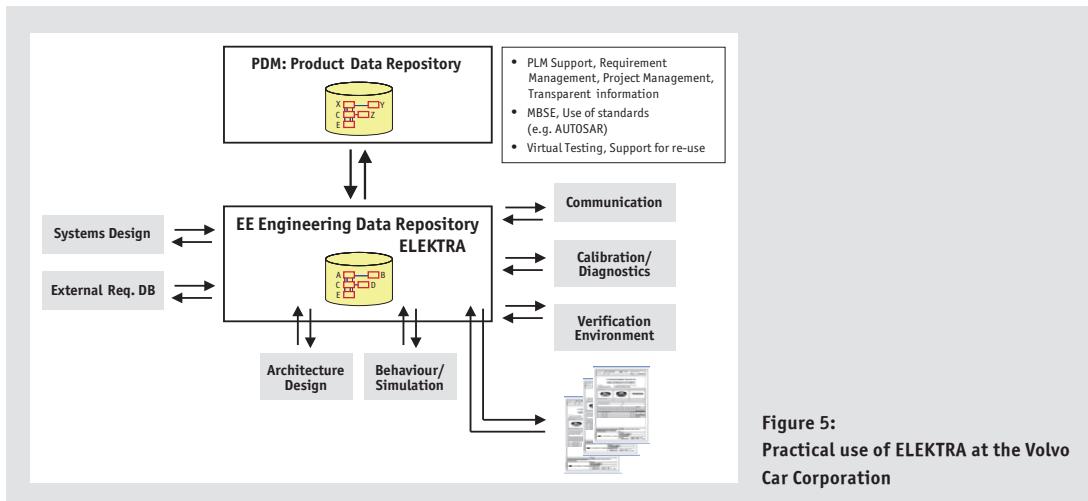


Figure 5:
Practical use of ELEKTRA at the Volvo Car Corporation

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: VCC's 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 ECU software and hardware. The main characteristics of such a solution should be:

- > Alignment of processes.
- > Single source of information.
- > Simple hand shaking mechanism for changes.
- > Reusability of information.
- > Utilize AUTOSAR standard.
- > Use commercially available tools.
- > Continuous development instead of step-wise.

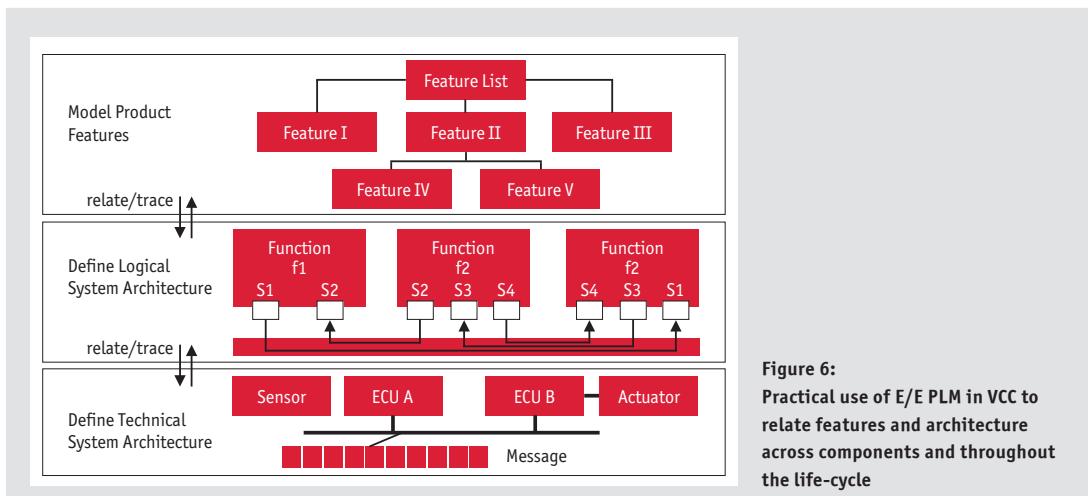


Figure 6:
Practical use of E/E PLM in VCC to relate features and architecture across components and throughout the life-cycle

4.2 PLM Development

After an intensive market research and pre-phase, VCC decided to implement the information backbone in a close development partnership with Vector. The VCC application is called ELEKTRA, based on the Vector product "eASEE Automotive Solution" (Figure 5). VCC uses the parts "System Data Management", "Test Data Management", "Requirements Management" and "Change Management" of the integrated eASEE tool suite. Figure 6 shows the practical use of E/E PLM in VCC to relate different features and artifacts throughout the life-cycle.

The possible efficiency win of this approach was investigated by a wide set of interviews, done in the E/E-engineering departments. The result was an estimate of 2% efficiency increase in the first year of the ELEKTRA implementation and an efficiency increase of annually 13%, beginning with the fourth year of the ELEKTRA implementation. This estimate with the related profits for VCC was the reason, that ELEKTRA still is the number one IT-project in the entire VCC organization.

As shown in Figure 7, VCC and Vector agreed to a four year implementation plan to realize a continuously increase of the ELEKTRA functionality and to achieve the completeness of the ELEKTRA application.

4.3 PLM Deployment and PLM Operations

A project like ELEKTRA is a long term activity which has to be funded by big budgets of money and significant internal resources. Therefore the **decision preparation** needs care:

> 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 partner must be reliable and the chemistry between the main actors should secure the probability to realize a true long term partnership. As an important base for this, both – the customer and the vendor – should share the same vision of future E/E-engineering methods and processes.

Especially at the beginning of these kinds of projects the danger of too big ambitions and a too broad scope of functionality are huge. In the **planning and implementation phase** VCC had a lot of success with the approach to focus on one certain use case. For the first versions of ELEKTRA, VCC concentrated on the automated creation of SRD (subsystem description e.g. for locking) and SWRS (software requirement specification e.g. for a body controller). Further the close links between requirements and test cases was part of the first ELEKTRA release. Further design verification methods are documented in ELEKTRA. This focus was manageable for VCC in its role to define the requirements and for Vector to implement the agreed data model and the core functionality in the eASEE Automotive Solution. And – nevertheless, even with this first release a lot of the drawbacks of the former organization and tool environment could be eliminated. Further both parties need a strong and competent leadership on project and management level.

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

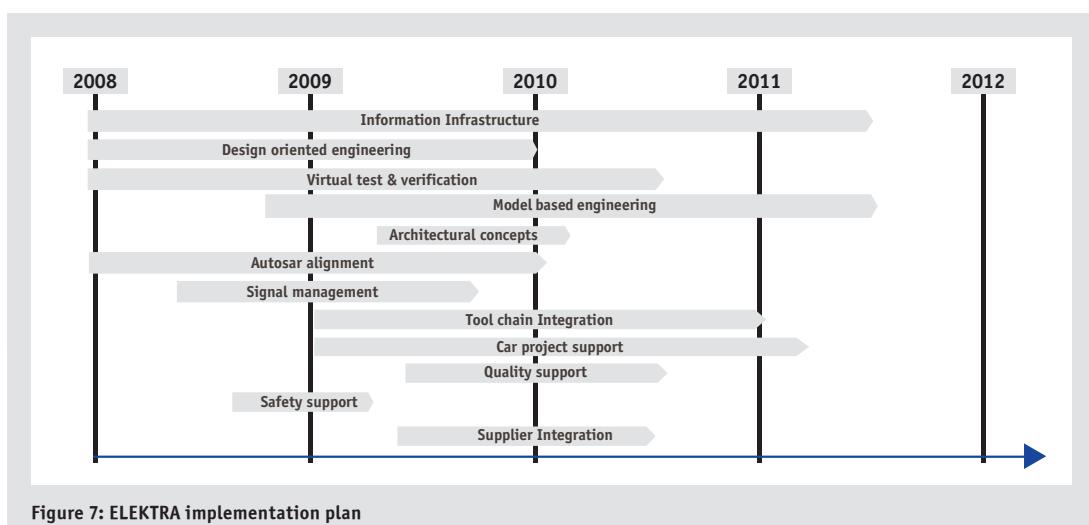


Figure 7: ELEKTRA implementation plan

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. At least VCC made good experiences with an iterative-incremental development process for ELEKTRA – 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 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. VCC managed this situation with special ELEKTRA trainings for key users and initiating “Jump Start Projects”. In these projects the VCC engineers were supported by well known and well with VCC specific engineering knowledge experienced consultants, which have been trained in ELEKTRA before. In this combination these engineers did the same work as a VCC engineer, but in addition they were able to train and coach the VCC colleagues in doing their daily work in the ELEKTRA environment. The “Jump Start Projects” are focused and have a predefined duration (typical two months).

5 Summary and Conclusions

For most companies, there is a wealth of untapped opportunities to cut costs from their development projects. 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. 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.

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 we found the following lessons learned:

- > PLM concept: First improve the process then the tools based on concrete improvement objectives that are set, measured and used to correct deviations.
- > PLM development: Evaluate tools under realistic conditions. Agree specific requirements to the process and tools which are then used to drive changes.

- > PLM deployment: Manage the changes as they impact the entire organization. Pilot changes, coach and train engineers, highlight power users that will set the pace.
 - > PLM operations: Support users and ensure continuous improvement.
- Embarking on a state of the practice E/E PLM 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.

Translation of a German publication in Automobil Elektronik, 11/2012

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. and R. Dumke: Software Measurement. Springer, Berlin, Heidelberg, New York, 2007
- [3] eASEE Product Lifecycle Management. <http://www.vector.com/easee>
- [4] Chrissis, M.B., M.Konrad and S.Shrum: CMMI. Guidelines for Process Integration and Product Improvement, ed. 2. Addison-Wesley, Reading, USA, 2006.

Dr. Christof Ebert,
Vector Consulting Services GmbH, Germany

Jerker Andersson,
Volvo Car Corporation, Sweden

Eduard Metzker,
Vector Informatik GmbH, Germany

The drive to bring technological products to market maturity more quickly and efficiently not only places severe demands on product development but on underlying technical business processes as well. Vector supports you with the consulting services and tools needed to optimize your technical business processes and sustainably embed them in your development organization.

www.vector.com/consulting
www.vector.com/eASEE

Functional Safety Industry Best Practices for Introducing and Using ISO 26262

Functions such as adaptive cruise control, crash protection systems, active body control and ESP are increasing in complexity and taking an ever more active role in controlling the car. These functions are realized by systems of sensors, actuators and interconnected electronic control units. The systems must be designed to function under a variety of operating conditions and must adhere to a number of mechanical, hardware and software constraints. In order to be able to manage the emerging product liability risks associated with such systems as well as ensuring the high level of quality required of automotive systems, significant improvements to engineering processes are necessary. In this article, we describe our experiences in adapting companies' development processes to conform to safety standards and to cope with the challenges mentioned above. We detail key success factors in overcoming these challenges and provide practical examples from working with global OEMs and tier-one suppliers on implementing safety standards in E/E development.

Some years ago an electronic parking brake system was introduced in order to assist the driver as well as to save on weight and mechanical overhead and cost. The principle was very simple. Once the brake was activated it would prevent the car from rolling and as soon as the driver activated the throttle, it would release, thus relieving the driver from handling the synchronization of releasing the brake while simultaneously pushing the throttle. The concept worked fine and of course, the electronic parking brake had two redundant channels straight from the parking button to the brakes. During a test drive on a hot summer day, the driver stopped the car to check something outside and activated the electronic parking brake. He left the engine running as it was a short stop and he only intended to briefly leave the vehicle. The car was, after all, secured by the parking brake. A few seconds after he had left the vehicle, it suddenly accelerated and crashed into a wall. What had happened? The electronic parking brake system just worked fine. But, when the driver left the car, he naturally opened the door. This allowed hot air into the vehicle. The air condition activated itself to sustain the desired interior climate. Since it needed more power, it slightly increased the throttle – which released the brake...

Safety-critical systems have the potential to cause physical harm should they fail in their intended function. Failures can be due to random hardware faults (e.g. short circuits) or systematic design errors (e.g. software defects). The risk associated with the system is reduced by minimizing the probability of a failure occurring and limiting the consequences of unavoidable failures. With the increasing complexity of the vehicle, its electronic components and their interworking, safety concepts will therefore be at the core of any new design, be it for the change of an existing function, such as in above example, or be it for a completely new function.

Functional safety is a property of the system as a whole rather than just a component property, i.e., it depends on the integrated operation of all sensors, actors, control devices, etc. The goal is to reduce the residual risk associated with a functional failure of the system below a threshold given by the assessment of severity, exposure and controllability [2,3,4].

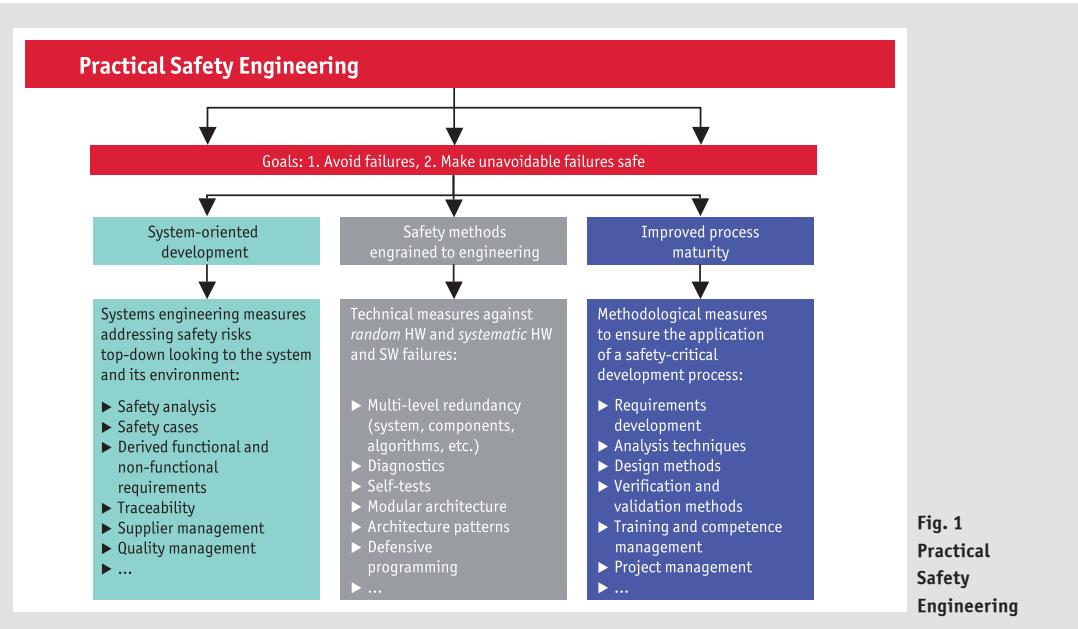
Safety as a basic concept has a long tradition in specific disciplines, such as the design of the passenger cell to protect drivers or the anti-blocking braking system that has strongly reduced fatal accidents [4]. However, today we have to acknowledge that safety has reached a new level of impact and risk. It is not anymore individual components that add to safety, but rather their interworking at the system level, that is the entire vehicle with its physical assembly of mechanical and electronic subsystems.

This article provides experience and guidance on how automotive E/E safety engineering can be successfully introduced. Standards such as IEC 61508 [1] and ISO 26262 [2] provide some guidance, but a thorough understanding of what safety means on engineering and management levels is absolutely critical to achieve a culture of safety within product development! Note in this context, that while IEC 61508 is still use in practice, there is a fast move towards the ISO 26262 which is the leading standard in functional safety for the automotive industry.

Our experiences from safety engineering in industries such as automotive, transport and automation show that safety engineering is only successful when considering three needs in parallel (see Fig. 1), namely

1. System-oriented development
2. Safety being an integral part of engineering methods
3. Improved process maturity

This article will detail these three needs and show how they can be translated into concrete development practices. With practical examples from working with global OEMs and tier-one suppliers on implementing safety standards in E/E development, the article will be of help to those engineers and companies just starting with safety engineering or optimizing their current practices towards more efficiency and effectiveness.



Challenges in applying safety standards in the automotive industry

Engineering safety-critical automotive systems is still a major challenge and often is associated with significant development overheads. The main obstacles towards achieving effective and efficient safety-engineering in the automotive industry can be summarized as follows:

1. Component-oriented development rather than considering the system-wide impacts of functions – and risks
2. Inadequate process capabilities due to many ad-hoc used methods and techniques and uniformed usage of what is supposed to be mandated by ISO 26262.
3. A rather complex standard with 600 requirements that leads to overheads and bureaucracy in many organizations rather than intelligent and efficient tailoring to needs.

The development of automotive E/E systems is still too component-oriented. Safety, however, is a property of the system as whole. The consideration of safety aspects across a number of components often requires a more systematic approach to requirements and quality management than is currently practiced as well as organizational changes in the development departments. It is often not possible, with current processes to ensure the traceability from top level safety goals at the system level (e.g. the electronic parking brake must not activate above certain speeds) to software safety requirements (e.g. plausibility checks on sensor values).

Safety-engineering is only achieved when following a strict process-oriented development with a rigorous application of project, quality and supplier management practices. This means for instance following documented process steps for which the engineers, and not just safety managers, had been trained. It also implies that work products are documented to achieve traceability across work products, and especially to validate safety requirements and to manage changes. These pre-requisites are often missing in the development processes we have observed and must first be established to form a basis upon which the safety-specific measures can be effectively applied. As a typical example, requirements on supplier development practices are often included in the requirements specification but not verified or enforced by the OEM during the development of the systems.

Current safety standards are rather abstract and do not give concrete guidance as to how the required measures can be efficiently integrated into existing processes [3]. The detailed changes and extensions to the processes required to achieve conformity need to be identified on a case by case basis for each organization. On the other hand, there is a trend visible in the recent evolution of the standards towards prescriptive details on how engineering processes, specifically formalized safety requirements and safety case specifications and their verification should be treated. The risk with this approach in standards is that engineers would just follow what is described and not really consider what is pragmatically necessary, looking to safety needs, risks and economic trade-offs. In consequence, safety engineering will be expensive and complex but not necessarily exhibit improved safety to the user.

This can be best understood when taking the example of accidents such as the London ambulance system (causing deaths during a reconfiguration activity) or the Therac medical radiation system (causing deaths due to radiation overdose) [11]. Both have been designed following the required standards, and still failed to exhibit safety because they lacked in usability. With untrained drivers that do not understand the complexity of underlying devices, this problem will accumulate. The message clearly is that safety standards do not provide sufficient guarantee for safety if above criteria of systems engineering and thinking paired with disciplined high-maturity processes are followed.

Success factors for safety engineering

From our experiences in introducing safety concepts to automotive OEMs and tier-one suppliers around the world, we see three needs to establish a safety engineering culture – on top of already institutionalized disciplined management and engineering practices, namely

1. System-oriented development
2. Safety being an integral part of engineering methods
3. Improved process maturity

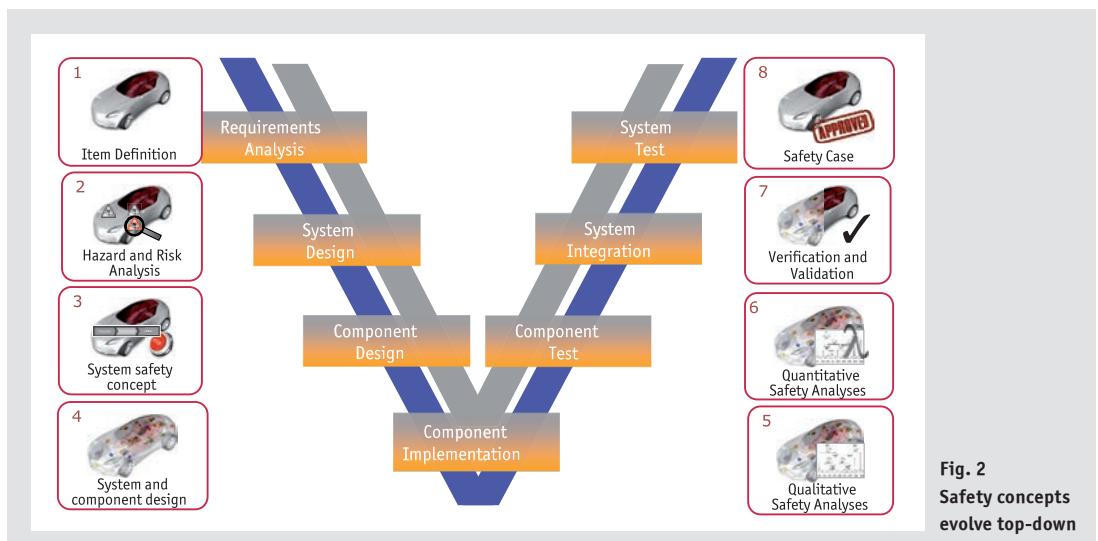
Safety demands a systems engineering perspective. Complexity growth within the electrical subsystems of the car, their networked interconnection, and increasing variability of features make embedded E/E systems more and more vulnerable. Such risks demand strong protection on various levels along the entire life-cycle. Most relevant above all is to identify the relevant safety risks how they

can be avoided or mitigated against at a system and software level. Especially for automotive embedded systems, safety must be handled starting with system specification, analysis and design. Once initial architecture and design is complete, the multitude of different components and their interaction will not anymore allow to retrospectively design for safety.

In the relationship management of automotive supply chains the principle of divide and conquer is hierarchically established and deeply integrated into the management and engineering culture. This obviously had its merits while nuts and bolts were being acquired that were selected from supplier catalogues. However, an electronic steering system, for example, cannot be acquired by referencing such standard specifications. It must be designed as an integral part of the vehicle, taking into consideration its physical, dynamical and mechanical behavior as well as electronic and many other constraints. Nevertheless, OEMs are still overly prudent not to disclose too much systems information, while their tier-1 suppliers are often looking towards their specific subsystems and trying to verify features on that level only.

The V-model with all its merits must be understood not from a piecemeal perspective where verification happens only on the lowest level, but rather as an integrated framework, where requirements and specifications for safety design, implementation and verification are derived top-down from systems requirements and design principles. Fig. 2 provides some examples of safety engineering techniques and how they are driven from a systems engineering perspective.

Safety concepts must be defined at the system architecture level. A systematic requirements management is required to document



the safety requirements derived from the system level safety analyses (using methods such as FMEA, FTA etc.), the allocation of these safety requirements to individual components and, after successive application of safety analyses at the component (hardware and software) level, to specific safety-related technical requirements on each component.

Each single engineering activity must be enhanced by adequate safety methodologies. Fig. 3 described the derivation of component specific technical requirements from the safety analysis and top-level safety goals. Vertical and horizontal requirements traceability provides a solid basis for managing the dependencies between safety requirements at various levels of the product architecture.

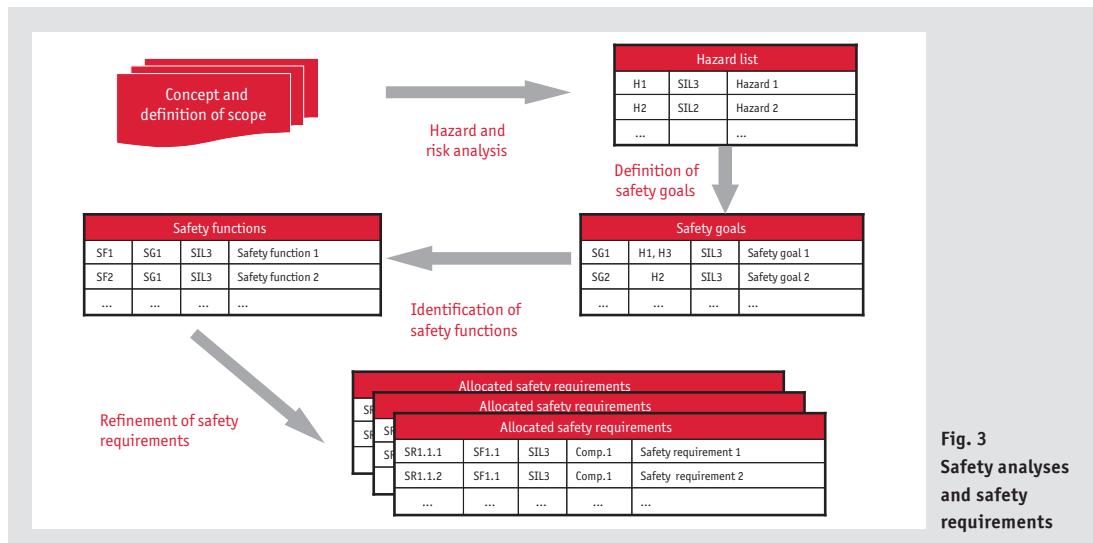
Due to the shift from component towards system-oriented development a re-allocation of responsibilities within the organization and between manufacturer and supplier is often required. There must be a clear responsibility for functional safety at a system level and the provision of the system safety case (to demonstrate the adequate implementation of functional safety). From a supplier perspective the critical requirements must be agreed with the OEM early in the proposal phase to evaluate the impact on BOM and development costs. Management and commercial analysis is heavily impacted by safety engineering. Tender analysis for instance demands close interaction to evaluate which requirements are appropriate given the level of risk and which measures might not justify the additional cost involved. From a legal perspective, the identified and agreed critical requirements must be documented in the product specification or product requirements specifications as applicable while ensuring throughout the development process

that they are traceable to technical requirements at the hardware and software level.

The development of safety-critical software will require placing more emphasis on excluding systematic errors through software architecture design, software implementation guidelines, and software verification and validation. Specific safety methods and techniques must be fully integrated within engineering. This is obvious when it comes to using coding standards and rules for programming safety-critical software [3,5]. But coding is not enough. The high quality demands of modern E/E systems can only be ensured with systematic engineering processes. Starting late and looking only to components as many were used to, is not the answer. A high overall quality level and thus early defect detection and removal is only feasible by combining a multitude of cascaded design, verification and validation activities starting with requirements engineering and only ending with end of service and retirement of a product or service.

Hardware development demands a documented development process including the use of analysis techniques for assessing the reliability of the hardware such as design FMEAs and worst case tolerance calculations. Potential hardware failure rates must be quantitatively modeled and used to apply hardware diagnostic measures in order to provide a more arguably robust design.

Safety engineering during design demands systematic defect prediction, detection, correction and prevention. The first step is to identify risks and hazards as they would relate to malfunctions and how defects would be critical to performance. The underlying techniques are statistical methods of defect estimation, reliability prediction and criticality assessment [8]. Systematic defects in the



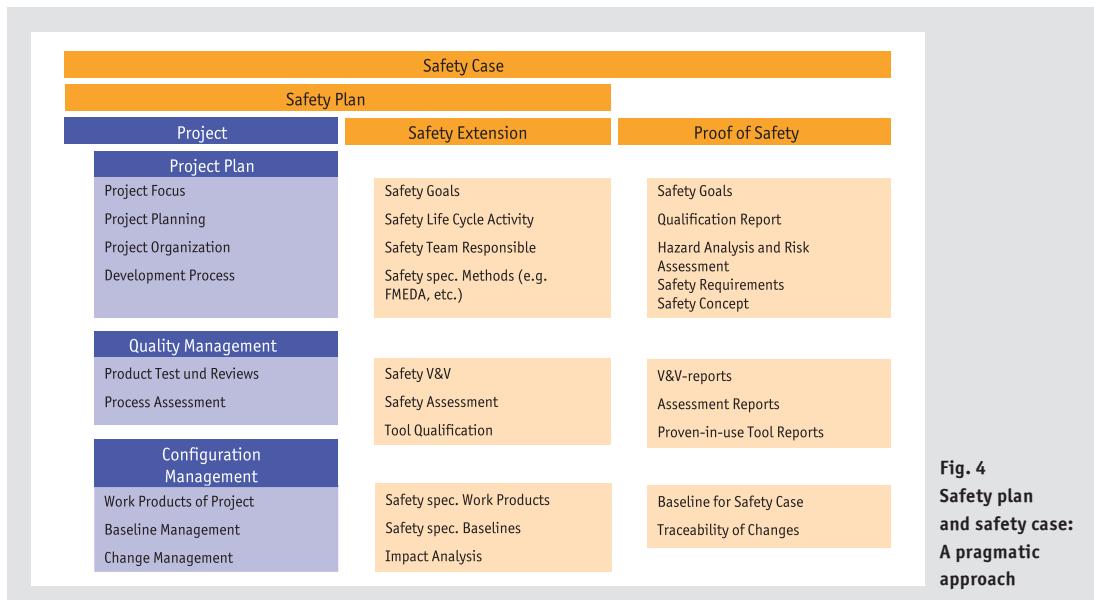
design or implementation have to be detected by quality control activities, such as inspections, reviews, unit test, etc. [9,10]. Each of these techniques has their strengths and weaknesses which explains why they ought to be combined to be most efficient. It is of not much value to allocate a large number of engineers to testing, when in-depth requirements reviews would be much faster and cheaper. Once defects are detected and identified, the third step is to remove them. This sounds easier than it actually is due to the many ripple effects each correction has to a system. Regression tests and reviews of corrections are absolutely necessary to assure that quality won't degrade with changes. A look to cost of non-quality and efficiency demands focus on preventing defects from re-occurring. Often engineers and their management state that this actually should be the first and most relevant step. We agree, but experience tells that again and again, people struggle with defect avoidance simply because their processes won't support it. In order to effectively avoid defects, engineering processes must be defined, systematically applied and quantitatively managed. This being in place, defect prevention is a very cost-effective means to boost both customer satisfaction and business performance, as many high-maturity organizations have shown [8,10].

In order to fulfill the requirements of the standards, process maturity is required. In other words, planned tasks for safety are tracked until completion, safety requirements are systematically identified and traced throughout the entire process and it must be shown that all necessary activities were performed conform to the process and that the product meets all its requirements (i.e., the

safety case). In our experience, the CMMI [6] and SPICE [7] frameworks provide a good basis for implementing safety processes. Often people claim that these models only apply for software, which is not true. Specifically CMMI had been designed for systems engineering and the development of components consisting of software, hardware and mechanics. As an ISO standard itself (ISO 15504) it nicely relates to ISO 26262 and helps in its implementation. Adapting development processes to conform to the safety standards as part of a CMMI or SPICE process improvement program ensures that the necessary pre-requisites are achieved which then form a solid basis for ensuring that additional measures for safety are effectively introduced in a controlled and sustainable manner.

Many standard requirements demanded by a thorough safety process are covered by the CMMI framework, which applies specifically to systems engineering and its underlying component design based on mechanics, hardware and software. Given the current focus of OEMs on supplier management and demanding the use of CMMI, this framework is thus a good – and cost-effective – basis for implementing safety standards. Note though, that the process framework needs to meet maturity level 3 requirements as this is where engineering is detailed. In addition safety specific practices are required (e.g. hazard analysis, safety assessments, compilation of a safety case) which can be integrated in process areas such as requirements development or product integration.

By addressing the safety aspects in combination with other process improvement measures, an efficient process realization can be achieved [8]. For example, a safety plan can be realized as an



extension to existing project and quality management plans and a safety case can be realized by instantiating these plans with reference to the results of the various analysis, verification and validation measures. Fig. 4 shows how the traditional development plans can be enhanced with safety-related extensions. The column on the left shows some examples from the development plan as it is today practiced, looking to areas such as project planning, quality assurance and configuration management. The middle column shows extensions for these three areas to design for safety. The column on the right shows which evidence, related to the plans, can be used in building the safety case. The “command and control” principle which we know from aerospace obviously applies to automotive safety engineering as well. Needless to say that the three mentioned areas of planning, QA and CM are just examples, and all process areas have to be extended, including domains such as training or measurement.

Systematic process improvement needs professional organizational change management. *Often we face organizations that apply the ISO 26262 in a bureaucratic and formal way (“we need to get certified”), which creates unnecessary high cost and demotivates engineers due to the formalism.* For the reasons described above (i.e., move towards system-oriented development and the need to establish basic process capabilities) adapting development processes to conform to the safety standards nearly always needs to be addressed as part of a systematic and wider ranging process improvement effort. This includes:

- > Aligning process improvement measures to the organization’s business goals to ensure that safety-related improvements are not introduced at the expense of other goals such as cost and efficiency.
- > A detailed understanding of the strengths and weaknesses of the existing processes. It is rarely possible to redefine development processes “from scratch” in working organizations. The additional measures required by the safety standards must therefore be intelligently integrated into the existing development practices.

In order to address the organizational changes associated with the move toward a system-oriented development, the improvement of basic process capabilities, as well as the establishment of safety-specific roles and development activities, an early consideration of organizational change management aspects is essential. This includes ensuring support of senior management in terms of communication of the organizational goals, the provision of resources and the authority to enforce the changes.

Conclusion

Safety engineering and thus the adoption of the leading safety standard ISO 26262 in the development of automotive systems is no rocket science and is absolutely necessary to avoid severe product liability risks in the future! We highlighted three key components for sustainable safety engineering in automotive systems, namely system-oriented development, safety methods close tied to engineering and improved process maturity.

Let’s go back to our introductory example to show what would be different. System-oriented development would have helped to consider the entire system and its signals, thus not using the throttle signal on the CAN-bus but rather taking the accelerating pedal position (which was in unused state). Integrating safety methods across the life-cycle would for instance create test cases in parallel to establishing the safety case. Those would be broken down to ensure on component and product level a full traceability from safety requirements and safety case to design decisions and their verification. Finally a higher process maturity would have demanded to review operational scenarios from different perspectives and evaluate alternative designs before implementing what looks most cost-effective but does not fulfill safety standards.

An efficient implementation of the standard requirements can be achieved through the systematic management and improvement of development processes combined with the introduction of few focused safety-specific measures. In introducing a safety culture to OEMs and suppliers we found two efficiency handles to reduce cost of safety engineering:

A maturity level 1 organization wanting to ramp up towards ASIL-C or ASIL-D typically pays 30-50% more for engineering. A combined introduction of safety engineering with improving process maturity to maturity level 3 would still create this cost, but additionally yield immediate returns through defect phase containment, easier change management and improved predictability – each contributing with 10-20% to reducing engineering cost.

Having a system perspective when specifying and implementing ASIL-C or ASIL-D requirements and designing components top-down in a distributed architecture achieves substantial benefits due to not demanding ASIL-C on each component and controller but achieving trade-offs by allocating cost factors to safety designs and then comparing overall system cost. We found at an OEM with this method that a combined electric and mechanical solution was from a life-cycle perspective much cheaper than having the same function fully implemented electrically.

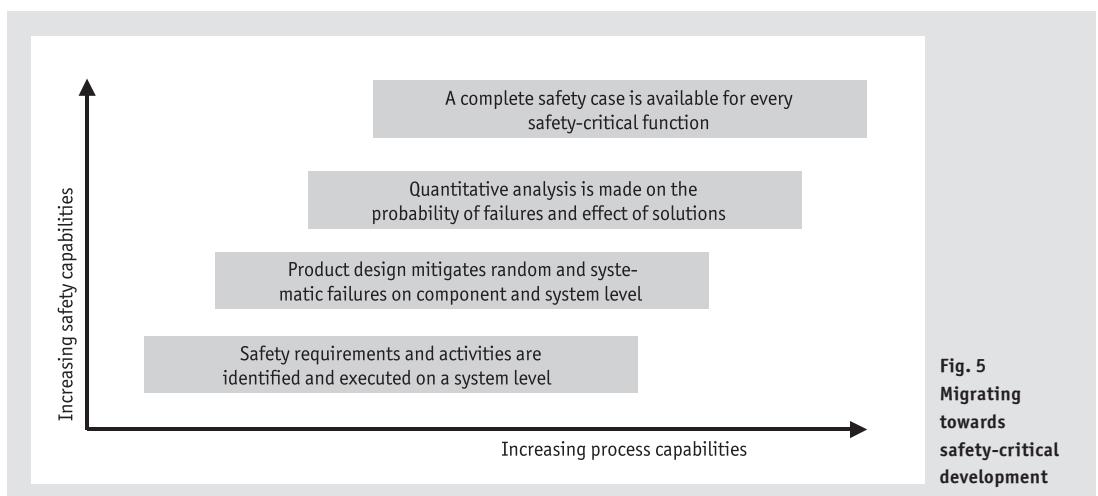
Let us conclude with an outlook on how safety further impacts the automotive industry:

- > Automotive OEMs in many cases still need to improve their process capabilities to fulfill the requirements of the safety standards and to better collaborate with suppliers
- > Suppliers of established safety critical components need to further improve field observation and abilities for complete safety case. Examples: Engine management systems, driving dynamics.
- > Suppliers of new and innovative components need to build up good basic process capabilities as a reliable foundation for safety. Examples: Innovative driver assistance functions and powertrain.
- > ISO 26262 will evolve based on experiences and to cover new challenges and development techniques.
- > Safety capabilities will become part of standard supplier evaluations.

Migrating to a safety-conform development will only be successful if it is understood and performed as a cultural and thus organizational change. However, too often, safety is understood primarily as a technical challenge where few additional requirements are added to an already overly long specification. A safety culture needs to be established. Management and engineering needs to understand the challenge of safety as a multidimensional need which impacts management processes, responsibilities and engineering methods. Functional safety needs to be seen as a critical product liability issue with all consequences on disciplined and formalized development. Engineers need to understand safety needs on the system level and adopt their engineering methods towards systematic and traceable decision-making from architectural to functional and component-levels.

For organizations starting towards a safety culture across their engineering we recommend an integrated and incremental approach to achieving safety standards while at the same time improving basic process capabilities. Fig. 5 provides an approach for facilitating step-wise yet deterministic migration towards full safety coverage.

Necessary improvements towards systematic safety engineering in automotive systems need to be made in coordination with other organizational goals such as cost, quality and efficiency. The good news though is that state of the practice techniques, such as engineering and management processes following CMMI Maturity Level 3 goes a long way in achieving many of the pre-requisites for fulfilling the requirements of the standards.



References

- [1] IEC 61508: Functional safety of electrical / electronic / programmable electronic safety-related systems (E/E/PES), IEC, <http://www.iec.ch>, 1998. See also: <http://www.iec.ch/zone/fsafety/scope.htm>
- [2] ISO 26262: Automotive Functional Safety, ISO, <http://www.iso.org>, 2011.
- [3] Smith, D. J. and K.G.L. Simpson: Safety Critical Systems Handbook: A straightforward guide to functional safety, IEC 61508 (2010 ed.) And related standards. Elsevier, New York, USA, 2010.
- [4] Pimentel, J.R., ed.: Safety-Critical Automotive Systems, Soc. of automotive engineers. SAE books international, Warrendale, USA, 2006.
- [5] Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications (Real-Time Systems Series). Springer, New York, 2011.
- [6] Chrassis, M.B., M. Konrad and S. Shrum: CMMI for Development: Guidelines for Process Integration and Product Improvement (SEI Series in Software Engineering), ed. 3. Addison-Wesley, Reading, USA, 2011.
- [7] ISO/IEC 15504:2004. Information technology – Process assessment. ISO, <http://www.iso.org>, 2004.
- [8] Ebert, C. and R. Dumke: Software Measurement. Springer, Heidelberg, New York, 2007.
- [9] Vector Informatik: Model-based Functional Safety in E/E system development. 2012. Accessible at: http://www.vector.com/portal/medien/cmc/press/Vector_Safety_AutomobilElektronik_201204_PressArticle_EN.pdf
- [10] Shull, F. et al: What we have learned about fighting defects. Proceedings of the 8th International Symposium on Software Metrics. IEEE, Los Alamitos, USA, pp. 249-258, 2002.
- [11] Leveson, N. G.: Safeware: System Safety and the Computer Age. Addison-Wesley. Reading, MA. 1995.

Contact Information:

Dr. Christof Ebert
 Vector Consulting Services
 Ingersheimer Straße 24, D-70499 Stuttgart
www.vector.com/consulting
 Mailto:consulting-info@vector.com

**Christof Ebert, Vector, Stuttgart, Germany**

Dr. Christof Ebert is managing director at Vector Consulting Services. He supports clients around the world to sustainably improve product strategy and product development and to manage organizational changes. Prior to that, he held global management positions for ten years at Alcatel-Lucent. Dr. Ebert serves on a number of advisory and industry bodies, teaches at the University of Stuttgart and has authored several books including his most recent book "Global Software and IT" published by Wiley. Contact him at christof.ebert@vector.com

Managing Risks in Global Software Engineering

Globally distributed software development poses substantial risks to project and product management. Not all eventualities can however be buffered, because in the global economy, developing and implementing products must be fast, cost effective and adaptive to changing needs. Therefore, there is a need to utilize different techniques to effectively and efficiently mitigate risks. This article systematically introduces risk management in global software engineering (GSE) for product development, service and maintenance. Methods include using basic project, supplier and quality management techniques, process frameworks (e.g., CMMI), product life-cycle management, effective communication processes, SLA based escalation, competence management, and innovation management. A longitudinal empirical field study over several years from a captive SW center of a world-wide leading ICT company in India provides practical experiences and indicates how to effectively apply GSE risk management practices

1 Introduction

With global markets and a world-wide race for talent with software skills, global software engineering (GSE) model, is fast growing. GSE ranks as one of the key principles and drivers of software engineering – seen from a business, education and research perspective. It is the consequence of the rather friction-free economic principles across the entire software industry.

Companies continue increasing their global development efforts. Over half have significantly increased offshore work in the past years [1,2]. **Fig. 1** shows for five different business sectors ranging from automotive and manufacturing to finance, consumer, ICT and health, how functions inside enterprises are already engaged into offshoring. Obviously IT functions have the highest degree of offshore capacity across the five sectors. However, the core of these sectors, namely R&D and engineering functions are at the steepest growth rate. It is not anymore that support functions and services are outsourced, as we were used to. Today's emphasis is on globally utilizing research and engineering to

develop products. GSE, at the crossing point of both the IT sector and the engineering function, naturally builds the spear head of this radical business change [3]. As a consequence, IT software and services exports are growing fast which we indicate for the two biggest source countries, namely India and China (**Fig. 2**) [4].

Many companies start global software engineering (GSE) due to perceived cost differences. An increasing share also takes advantage of skill availability in different parts of the world or quality awareness and process focus in some specialized industries and suppliers. Most of these companies engage globally active outsourcing companies to achieve fastest ramp-up of their globalization targets. After a while into that business they realize that savings are much smaller and problems are more difficult to cure than before.

What went wrong? Why do so many companies struggle to achieve the targets they initially set for their offshoring or GSE activities? What can we do better to make GSE a success? These questions have stimulated this article where we integrated experiences and empirical evidence from GSE over several years in different context and companies.

Offshoring penetration (%)

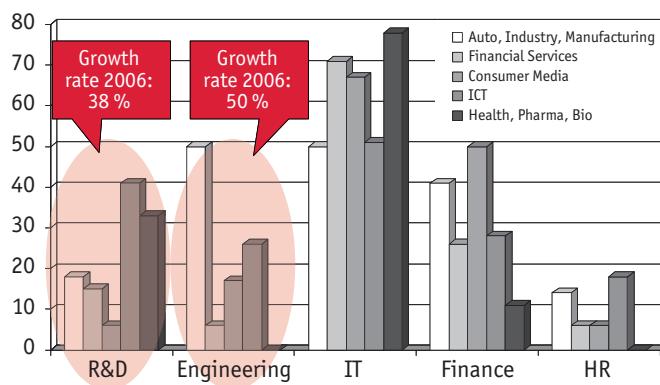


Figure 1:
Offshoring and globalization across
industries

While the cost advantage and skill pool further fuels GSE, we should be conscious on the related risks that come on top to the regular project risks. Not knowing them and not mitigating them, means that soon your project belongs to the growing share of failed global endeavors. This article will look into risk management and mitigation for GSE-projects. It will discuss risks and provide many practical hints how to increase the success rate.

The article will investigate risk management in GSE. We have analyzed typical risks as we observe them with clients in a variety of software and IT companies. We will provide principles and practices for effectively managing these risks. While it is relevant to systematically identify, evaluate, mitigate and monitor risks, most benefits will be drawn from looking to practical examples and lessons learned in industry.

Concrete empirical results are provided for specific risk mitigation levers from an offshore ICT software center in India, where we have been looking to a total of close to hundred projects over the past decade. Our research method is an empirical field study with longitudinal data collection over several years [5]. The projects from which we extracted data are of different size between few person weeks and close to hundred person years. They include application software as well as embedded software. Being GSE projects by nature, the type of global engineering varies between a captive mode, where requirements and architecture decisions are done in the Western hemisphere, while development and test are done in India. There are also scenarios included where software is developed in a shared mode with distributed teams.

We have practically applied and evolved all mentioned guidelines both in own projects as well as working with clients in different industries. Due to the rather similar approach to GSE across industries with regard to selecting, managing and maintaining supplier agreement and processes, the mentioned results can be readily used in other companies.

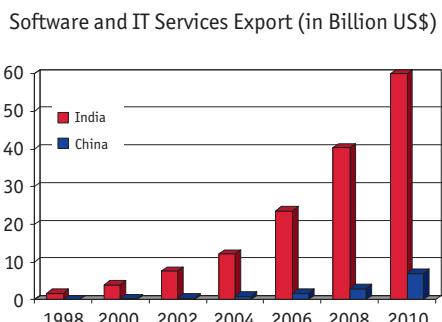


Figure 2:
GSE export revenues from India and China

2 Global Software Engineering Risks

Increasingly companies realize that despite all the need and drive for GSE, it is a risky business. Not all software engineering tasks and projects benefit from GSE! Many turn it down after having tried. 20% of sourcing contracts are cancelled first year, 50% downstream for not reaching objectives [5,6]. Why do so many externalized development activities fail?

Savings from GSE is around 10–15 % of cost reduction after a 2–3 year learning curve. Initially outsourcing demands up to 20 % additional efforts [3,5]. How can such risks are reduced and thus GSE efficiency be improved?

As companies turn to GSE, they find the process of developing and launching new products becoming increasingly complex as they attempt to integrate skills, people and processes scattered in different places. While the classic centralized software development approach allowed solving problems in the coffee corner or around the white board, GSE teams are composed of individuals who are culturally, ethnically and functionally diverse. They work in different locations and time-zones and are not easily reachable for a chat on how to design an interface or how to resolve a bug that prevents test from progressing. Only 30 % of all embedded software is developed in a global or distributed context, while the vast majority is co-located [1] due to the experienced risks with GSE. How can these risks are mitigated and thus flexibility be improved?

Risk management is the systematic application of management policies, procedures and practices to the tasks of identifying, analyzing, evaluating, treating and monitoring risk. On top of regular risk repositories and check lists, several specific risks must be stressed in global development projects. They relate to two major underlying risk drivers, namely insufficient processes and inadequate management.

To systematically identify risks and evaluate appropriate risk mitigation, we will look to the major drivers for GSE and then elaborate how they are impacted by specific GSE risks. Throughout our research over the past 10 years on global software engineering, we see four major drivers fueling the need for GSE, namely efficiency, presence, talent and flexibility. **Fig. 3** provides an overview on these drivers and how they relate to specific GSE risks. Let us look to these four GSE drivers and related risks. GSE risk management will then be illustrated with concrete benchmarks in the following chapter.

1. Efficiency. Software and IT companies need to deliver fast and reliably while at the same time the competition is literally a mouse click away. Hardly any other business has so low entry barriers as IT and therefore stimulates an endless fight for efficiency along the dimensions of improved cost, quality and time to profit. GSE clearly helps in improving efficiency due to labor cost differences across the world, better quality with many well-trained and process-minded engineers especially in Asia and shorter time to

profit with following the sun and developing and maintaining software in two to three shifts in different time zones. Directly related risks to the efficiency target are project delivery failures and insufficient quality.

2. Presence. Global R&D and software engineering has become part of companies' growth strategies, because they are closer to their markets and they better understand how to cope with regional needs, be it software development or services. Such global growth is a self-sustaining force, as it demands increasing capacities in captive or outsourced software engineering centers. Directly related risks to the ambition of presence in distributed markets are instability with overly high change rates (requirement changes) and inadequate IPR management. Risk of requirement changes is specifically included here, as we observed higher rate of change in distributed teams when compared to co-located teams.

3. Talent. Computer science and engineering skills are scarce. Many countries do not have enough resources locally available to cope with the demand for IT and software products and services. Fueling this trend, many younger people got nervous with media-driven misperceptions about the danger of outsourcing and GSE for the entire field of software, that they decided to rather engage in fully different fields. The consequence is a global race for excellent software engineers. GSE is the instrument to provide such skills and handle the related supplier-processes. Directly related risks to the drive for global talent allocation are staff turnover rates, insufficient competencies and wage and cost inflation.

4. Flexibility. Software organizations are driven by fast changing demands on skills and sheer numbers of engineers. With the development of a new and innovative product many people are needed with broad experiences, while when arriving in maintenance, these skill needs look different and manpower distributions are also changing. Such flexible demand can not anymore be

handled inside the enterprise. GSE is the answer to provide skilled engineers just in time and thus allows building flexible eco systems combining suppliers, customers with engineering and service providers. Directly related risks to the flexibility goal are poor supplier services, lock-in, and distance & culture clashes.

Obviously not all companies that engage in GSE look to all four drivers with the same motivation. As a matter of fact, we even see a kind of trajectory where a vast majority of companies starts with efficiency needs (i.e., cost savings), and then moves on to presence in local markets, and only after these two forces are understood moves further to talent and flexibility. Also, it is clear that these four factors feed themselves. The more energy a company spends on for instance building a regional pool of skilled software engineers, the more it also considers how to best utilize these competencies to, for instance, build a regional market or develop new products for such markets.

In consequence not all companies will face mentioned risks in same depth and at the same time. In order to validate this risk list, we performed two types of analysis. First we did a profound analysis of GSE projects from ICT during the timeframe of 1996 to 2007. Looking to over hundred projects performed in ICT software centers in India, China, Brazil and Eastern Europe, we found a consistent pattern of the top ten risks. We then validated this initial list by looking to companies we are consulting with and also to published field studies [1,5,6,8,9,10,11,12,14]. Depending on the specific GSE-layout (e.g., with or without external supplier), the ranking list of these top-ten risks is as follows:

1. Project delivery failures
2. Insufficient quality
3. Distance and culture clashes
4. Staff turnover (mostly for captive centers)
5. Poor supplier services (only for outsourced GSE)

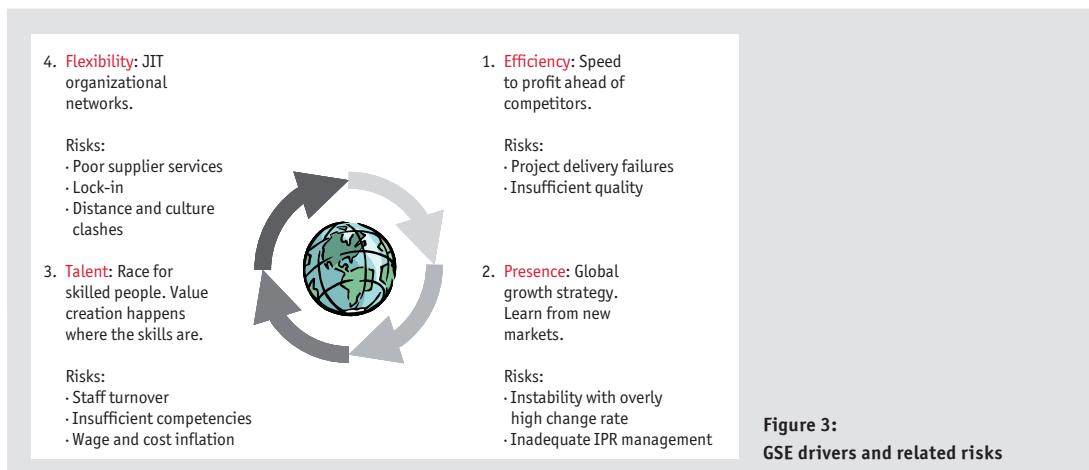


Figure 3:
GSE drivers and related risks

6. Instability with overly high change rate
7. Insufficient competences
8. Wage and cost inflation
9. Lock-in (only for outsourced GSE)
10. Inadequate IPR management

We will furtheron look to these risks in detail, sorted according to the four drivers as depicted in **Fig. 3**.

A comparison of the above risk list with published research, specifically, Boehm [14], shows that except for items 3, 8 and 9 all other risk items are traditional project risks. However they combine differently and their effects influence each other. For instance, distance and culture clash results from developing projects in low cost countries with a corresponding reduction of experienced manpower in previous high cost countries. A conflict arises to manage restructuring on one side and transition knowledge from the same people to newcomers.

3 Risk Mitigation Patterns

3.1 Efficiency

Project delivery failures. A standard risk for many projects, the risk of being late or over budget amplifies in probability and impact due to the intrinsic difficulties of managing a global development team.

As mitigation, project and team managers must be educated in estimation, planning, dependency management, uncertainty management, project monitoring and communication. The latter is crucial as experience shows that projects fail not because of unknowns but because of not willing to know or to communicate known facts.

We have seen from ramping-up internal software teams in Eastern Europe, India and China that solid processes not only accelerate introduction of GSE but also serve as a safety net to assure right training, good management practices, etc. We conducted a controlled experiment when ramping up offshore development teams

in China. Our experience was that the building of such globally distributed development team was fastest and most reliable in the case where the demanding organization was on CMMI maturity level 3. The same was done with lower-maturity demanding organizations with the effect that the CMMI maturity level 2 organizations could manage with some external support, while the maturity 1 organization failed due to highly inefficient interface frictions and lots of rework.

We recommend maintaining an organization-wide risk repository with all project risks together with identified mitigation actions. At the start of a new project, the project manager has to take this organization wide risk repository and check what specific risks are applicable to his project together with any new items. The second, more a medium term approach, is to train all project managers. Using the CMMI and certifying in professional project management is an effective mitigation.

Another important and easy mitigation action is building on past project experiences. The key parameter for project success is schedule adherence. We suggest doing a periodic Root Cause Analysis (RCA) on completed projects and identify the key issues that contributed to delays. On these issues we can do a Pareto analysis to define focused actions for the most critical and repeating issues. **Fig. 4** shows the impact of project delivery risk mitigation indicating a clear reduction in spread of schedule deviation, over years, as we increasingly apply our learning from previous projects in to next projects.

Insufficient quality. Working in different places or with teams in different organizations means, that many work products are moved across such places and teams with the risk of insufficient quality. Often the underlying rationale is that teams suppose that there will still be sufficient validation "downstream" so that quality deficiencies accumulate. Many global development projects suffer from a "ping pong" approach of work products being thrown back and forth due to poor quality. These stories repeat each other – independent of countries and culture. The designer in the Mexican

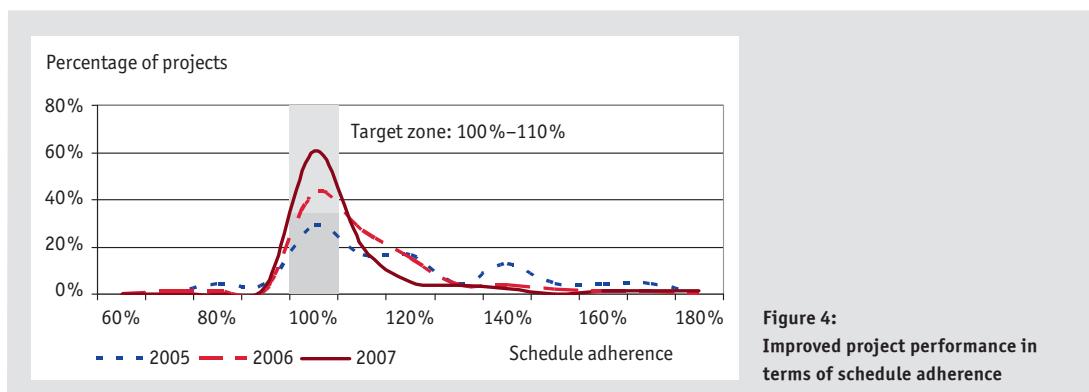
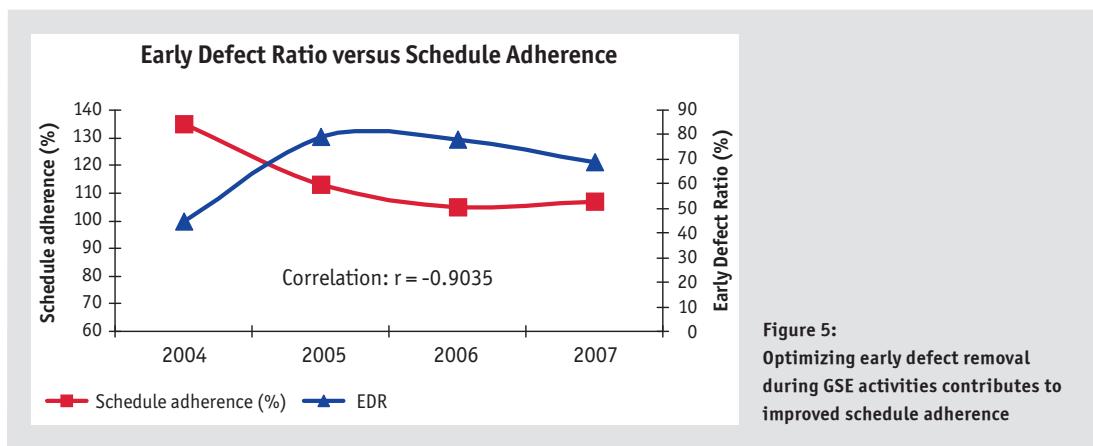


Figure 4:
Improved project performance in terms of schedule adherence



team claims that the US-American specification was not good enough, while the integration tester in India kicks back the product because Mexico again delivered insufficient code quality.

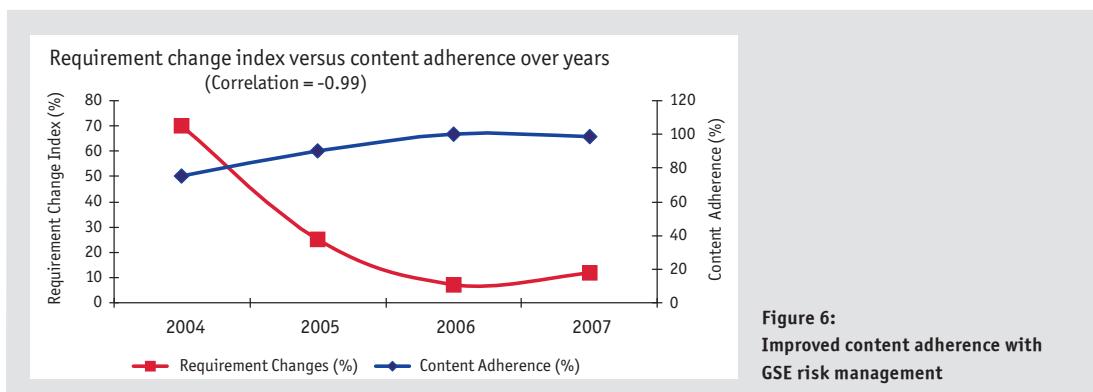
The major risk mitigation to such repeated rework and increasing mistrust is to force quality gates on work product level. A work product is only accepted if it has the right quality level. Incoming work products are inspected at least in samples to check consistency and quality. Service level agreements and responsibility assignments reduce the “ping pong” effect because at the least it is clear who should do better.

We look to “Early Defect Removal” (EDR) as the measurement of defects found before start of test compared to total estimated defects and compare it to upfront-defined threshold [3]. It will provide warning signals so that corrective actions can be initiated, well before the product becomes due for delivery to customer. Having worked with this concept over many years in different companies, we observe a strong negative correlation of -0.9035 between mean EDR and mean schedule adherence for a set of around fifty

projects in the timeframe of 2004–2007. **Fig. 5** shows this trend over four year period which indicates that EDR is indeed an advance indicator to reflect on quality and thus schedule variance of the product.

3.2 Presence

Instability with overly high change rate. Frequent changes create extra cost. Often being present in different markets with individual engineering teams means that each of the teams first of all looks to needs of the local market. When products and features are assembled, inconsistencies appear which cause late requirements changes. Global development amplifies such requirements engineering weaknesses that have in most companies long been present but could be camouflaged due to collocated teams. If specifications are insufficient, a remote team will either misunderstand or not accept them.



As a mitigation, GSE demands more reliable requirements and change requests. We recommend enforcing a rigid roadmapping process that provides sufficiently early insight into feature evolution and release planning. Teams will appreciate it with more anticipation and design for change. Global development demands more communication than co-located development. Specifications and documents must be carefully reviewed, because engineers on the other side trust to what is written. Establish for all distributed work packages a baseline and configuration management based on defined and proven quality entry criteria. The expectations and deliverables in terms of effort, deadline, duration and quality levels are to be clearly documented and agreed.

We observed that the number of changes to the requirements are highly (negatively) correlated to the content adherence, defined as percentage of features delivered at the end of the project to that of original required features at the start of the project. The correlation coefficient which we use as an early risk indicator and thus lead indicator for risk mitigation actions is -0.99 for a set of around fifty projects in the years of 2004–2007. A possible way to mitigate the risk of uncontrolled requirement changes is to closely monitor the requirement change index (number of changes to the features divided by total number of features required at the start of project, expressed as percentage). **Fig. 6** shows how content adherence is related to requirement change index over years.

Inadequate IPR management. Intellectual property rights are a key success factor in software development. Mostly software is not patented and copyrights are not enforced equally in all regions of the world. Further risks are related to improper use of external software (e.g., OSS) and careless handling of confidential information (e.g., leaving contracts at printers, etc.).

As mitigation, make sure that the intellectual property is well-secured. Divide key assets into pieces and provide only fragments to each global team. Share according to strategic relevance. Reinforce copyright protection for external sources. A GPL-protected component included with your product might force you to fully disclose all the software of the product. This can be handled both on

policy and architectural levels. Most relevant however is that your teams get trained in copyrights and specifically on the traps related to open source software. Install and enforce effective policies for confidentiality, copyright protection and intellectual property handling and train all software engineers and managers on it. Rigorously punish wrong-doing and unprofessional behaviors in this critical domain.

At the same time do not underestimate the potential of new teams to explore the innovation. The first step of innovation is not knowing what is the right (read legacy) way of doing things and the distributed teams have an inherent, hidden potential to innovate. Creating awareness on how to identify IPR and potential patentable ideas, on-line forums to share ideas, voluntary moderators to guide raw ideas in to potential patents is something not to be ignored.

Fig. 7 shows how distributed teams have generated the patent proposals after awareness training and workshops are conducted in 2006 and 2007, respectively. The correlation between training and workshop timing and the number of idea generation can be seen as a quite strong, indicating the huge effect such awareness has on protecting IPR in GSE.

3.3 Talent

Staff turnover. This is a specific risk especially in Asian countries due to abundant job opportunities in the respective economies. It is a generic risk whenever GSE has no clear integration with an organization's overall engineering strategy and career paths, such as having a nearshore maintenance organization within a software company.

Regarding attrition we have to apply two parallel mitigation strategies. First, it is clear that attrition in certain places of the world is higher than, say in western Europe. So we have to cope with it and prepare to learn and live and deal with attrition in advance. This means advanced planning of buffers, long term retention mechanisms like loyalty bonus etc. Buffers could be foreseen if engineers' unavailability exceeds certain thresholds. Note

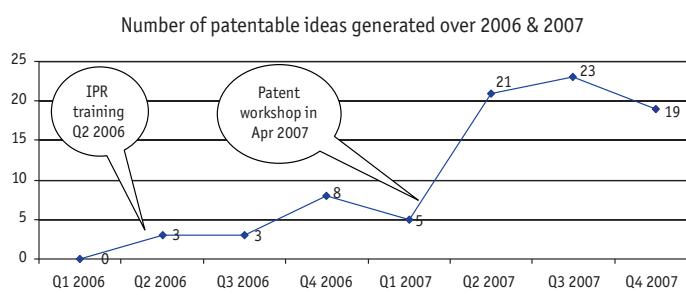


Figure 7:
Impact of IPR training on patent generation and thus IPR protection

though that such buffers immediately impact the bottom line and should be carefully pooled to serve several projects.

Second we should measure attrition and its impact factors in order to control and limit staff turnover. We recommend conducting periodic employee engagement studies from which we can learn and improve the working environment, which shall limit attrition to manageable levels. Based on surveys we then look into specific incentives to keep people, even in times where stock options are not the preferred instrument. For instance, international career paths and excellent individual development skills reduce attrition.

Insufficient competencies. This is a risk in each development project; however it is amplified by the bigger dependencies given the globally distributed team combined with less visibility on resource planning and skills availability.

For mitigation we recommend assuring global competence management and resource planning (e.g., with a multiproject management tool such as Primavera or similar) and a skills management on the level of detailed technical skills necessary for the projects. Note that competence management is not the same as above mentioned attrition management. It is however a necessary condition to reduce attrition.

We recommend managing competency needs in parallel to technical and project roadmapping. It is a strategic task in the hands of engineering management, not HR. Organizations spend huge effort in training employees but often does not correlate whether the training and improved skills reflect the business needs. In our case study we have converted the overall team's competency and skills (C&S) range in to a single measurable number by linking available C&S and normalizing with required business needs. This single index is monitored for improvement. A direct consequence of skill management is enhanced retention of employees. We observed a strong negative correlation between skill index computed during

one year and that of team's lead attrition, i.e., attrition of the teams in next year. **Fig. 8** shows the strong influence of team skill index measured at end of 2006 with that of full year attrition in 2007. The correlation factor within a community of close to 1000 engineers is -0.85.

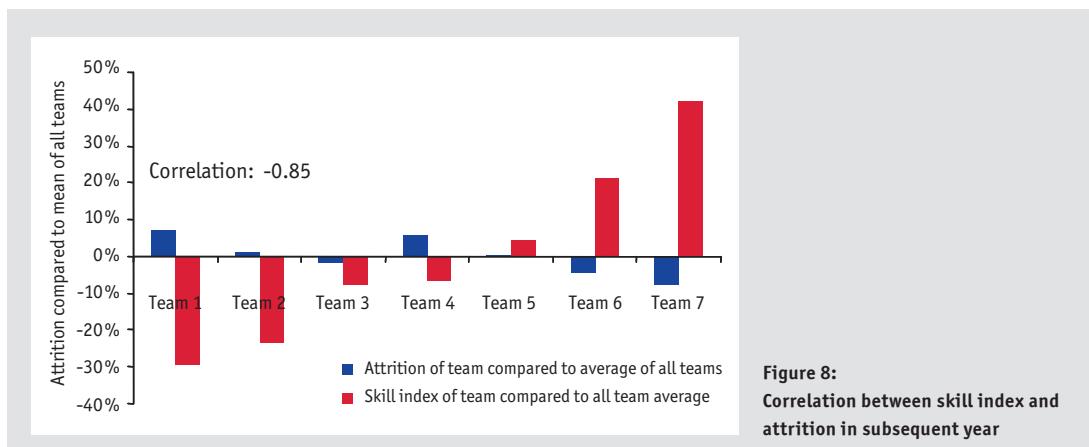
Wage and cost inflation. With the global fight for software engineering talent, wage inflation is a major GSE risk. For instance, salaries in India have increased by 14% pa between 2004 and 2007. The annual increase in most Asian countries is around 10% [1,2].

The primary mitigation is to carefully consider which regions to utilize and to make a profound business case and cash flow planning taking into consideration expected wage increases in different regions of the world. Where external suppliers are involved, evaluate upfront their business models and past cost evolution. Determine upfront which supplier size fits best to your own company size and structure. A big supplier with a small enterprise won't fit, because the SME will not have the chance to make corrections once the contract is settled. Evaluate offers during the tender of a supplier contract with dedicated estimation tools such as QSM SLIM to compare and judge feasibility [3].

3.4 Flexibility

Poor supplier services. One frequent risk with third party suppliers is not meeting the expectations in terms of quality and delivery schedule. SLAs won't help because once this point is reached, even escalation will not help much, because it will take too long and the product or service quality is already hampered.

Therefore the primary mechanism for risk mitigation related to insufficient supplier services is to carefully evaluate one's own processes and those of potential suppliers – before engaging into GSE. As a client you should always consider the golden rule of supplier management: You pay for what you get. Don't get trapped into



contracts that look “cheap” and later bring tons of extra cost due to weak processes and insufficient delivery quality. Preconditions of any successful supplier management are good processes on both sides, i.e., for the client and the supplier. Insufficient client processes cannot be externalized. They will not scale up from a single site to several sites. Often those low-maturity processes can be handled in localized development without many overheads due to co-located teams, but will fail with globalization. From our experiences we recommend having a CMMI (development and/or acquisition) maturity level 3 on both sides – and being applicable for all impacted engineering and sourcing processes.

When still in preparation mode, negotiate for a fixed project cost, where price is fixed and linked to deliverables with specific quality targets, often including penalty clauses. A fixed price project will make the supplier proactive for performance as payment is linked to quality deliveries and not time spent. Though generally fixed price is at higher cost compared to time and material at the contract negotiation stage, our lesson learnt is at the end of the project, it turns out to be generally 10 to 15% less costly than comparable time and material projects.

Supplier management includes clarification on non-disclosure and related agreements before starting negotiation. Establish clear acceptance and liability rules following contracting and legal schemes of your headquarter base. For maintenance projects they also include clear SLA in terms of response time, solution time, percentage of return failures etc. Set up clear and measurable service level agreements. Ensure that this SLA contains all that matters for you in the contract. Insist on periodic reporting according to the SLA. From the beginning define thresholds that establish when and how insufficient performance will be escalated. Measure supplier capability or demand such measurement based on industry standards, such as CMMI. Relate value you receive from suppliers to the risk and cost of the delivered services or components. Implement contract evaluation after each single project.

Consider sufficient time and budget (resources) for training the supplier on your processes. A very strong training tool is the Scrum process with short team meetings every day where recent results and next steps are briefly reviewed. Any uncertainty should be brought up in such reviews, which should take not more than 15–30 minutes and can be conducted even per telephone conference across sites. Build a supplier program management to handle the necessary review and decision processes. Agree with your supplier review and acceptance processes to assure the right quality level. Installing such processes after the contract signature will create the perception of policing the supplier. You can ask third parties in case of questions or needs for escalation.

Lock-in. With GSE supplier competition on a global market, external suppliers often start with rather low rates and once the projects are sufficiently large clients might be forced to lock-in with them due to progress of product development and knowledge

transition. In the least we may have to face increasing cost inflation.

The primary risk mitigation is to have multiple partners and distribute critical knowledge on two sources. Each one shall know that we have a choice to make and that will make the external suppliers to remain competitive. To improve efficiency and reduce effects of lock-in, global teams must use the same tools, methods and processes. It is worth the extra money for tools licenses, although in a low cost country the additional load on engineering cost can be 10–20% for the necessary design tools. Our recommendation is to avoid supplier-specific and ad hoc tools as they won’t scale up and can bring substantial issues if backups cannot be restored or contents are corrupted. Process improvements and best practices gained over years of experience in one engineering team need to be replicated quickly, in other engineering teams. Common processes and tools across engineering teams will benefit quick spread of lessons learnt/defect prevention actions across teams.

Lock-in goes beyond suppliers. Do not forget about risks related to certain regions of the world, where you might currently be locked-in. We also recommend maintaining flexibility in where you work and with which supplier. Instabilities can be caused by political turmoil as well as earthquakes, civil war or terrorist attacks. Don’t put all your global development into one single site. Consider distributed hosting of infrastructure and backups. Periodically test the restore mechanisms to a different new site.

Distance and culture clashes. Globally distributed software development is highly impacted by work organization and effective work split. Working in a globally distributed project means overheads for planning and managing people. It means language and cultural barriers [13]. It creates jealousy between the more expensive engineers being afraid of losing their jobs, while forced to train their much cheaper counterparts. The barriers to such harmonization and cooperation are not to be underestimated. They range from language barriers to time zone barriers to incompatible technology infrastructures to heterogeneous product line cultures and not-invented-here syndromes. An obvious barrier is the individual profit and loss responsibility that in tough times means primarily focusing on current quarter results and not investing in future infrastructures. Incumbents perceive providing visibility a risk, because they become accountable and more subject to internal competition.

As risk mitigation we recommend collaboration and communication. Collaborate across disciplines, cultures, time, distance, organizations. Communication starts before the GSE project is kicked off. Fears, hopes, barriers must be articulated. Assess your organization carefully on such distance and culture risks. This demands a fully new skill set, currently not taught at universities (e.g., managerial, teaming, sharing without loosing) [3]. Cultural sensitization, periodic workshop between clients and suppliers and networking between various teams has been the effective risk

mitigation strategy. Provide space for engineers to share engineers' emotions with team leaders openly. Establish early warning systems to "smell" upcoming barriers and fears.

Collaboration also means effective and efficient tools support. The exchange of information between sites must be carefully planned. The closer tasks and software components are linked, the more need for good data communication. Tasks with high overlap should not be done with too much time distance. Especially with a high work time overlap, online collaboration has high demands on fast, reliable quality of service for video, engineering tools and online collaboration. A change management tool is not enough because engineering demands collaboration on content and knowledge. Plain supplier management platforms as they are offered today for handling online market places and tenders are also insufficient due to their limitations in sharing engineering information. You will need rules and workflow support for documentation, design reviews, change management boards, etc. We recommend installing workflow management and online accessible project, work product and process information to assure proper knowledge management.

Conclusions

Over the past years GSE – both captive and with external suppliers – has grown at a rapid pace. GSE today is part of the software engineering business and discipline as is testing or project management. It is not anymore nice to have for cost reduction, but a need to have for sustainable growth and competitiveness. Independent of their size, software organizations have recognized GSE as a key driver to become more **efficient**, achieve **presence** in world-wide markets, have access to **talent** according to actual needs and still have the **flexibility** necessary to cope with changing demands and rapid technology changes. These four drivers determine the off-shore strategy of software and IT companies.

GSE amplifies typical software project and product related risks, such as project delivery failures and insufficient quality. Worse yet, it creates new risks, such as inadequate IPR management or lock-in situations with suppliers. These risks must be identified in due time and have to be considered together with the GSE strategy and its operational implementation.

This article highlights the **top-ten GSE related risks** as we have identified them over the past decade in a multitude of GSE projects and situations covering four continents. They are not specific to an industry or company size, but rather to the underlying life-cycle processes and management practices. Based on Alcatel-Lucent's experiences and enhanced with consulting work in several other globally acting companies, we show impacts of these risks and also how they can be effectively mitigated. For this paper we have studied the project results over years and analyzed the correlation between risk mitigation actions and project deliveries. This paper is a result of empirical study of historical project data together. Whereas 7 out of 10 experienced risks in this paper are common to already published research, 3 new risks are presented. We have verified the validity of risk mitigation actions by establishing a strong correlation with expected end results of projects. Our experience clearly demonstrated that applying the discussed risk mitigation methods, has resulted in improved delivery schedules as outlined in **Fig. 4 and 5**.

As a conclusion and for practical usage the risks and our recommended mitigation actions are summarized:

Validity of risks. Not all of the above risks are suggested mitigation actions may be applicable to all organizations. Wage and cost escalation will not be an issue for growing teams, as generally new recruits are at a less cost than existing average and per head cost will come down, even if wage cost is going up. Professional training like certification of project managers increases the risk of attrition due to better sellable skill level in market! Long term retention methods for attrition management will itself contribute to other

Risk	Mitigation Actions
Project delivery failures	<ul style="list-style-type: none"> Professionally train all project managers. Apply best practices from the CMMI (DEV + ACQ) frameworks and target maturity level 3. Maintain an organization risk repository. Use lessons learned and root cause analysis reports from previous projects to avoid repetition of problems.
Insufficient quality	<ul style="list-style-type: none"> Implement and systematically follow quality gates at work product level. Establish and use quality indicators. Monitor and use early defect ratio as a warning sign of insufficient specification and code quality.
Distance & cultural clashes	<ul style="list-style-type: none"> Establish open communication across multiple channels. Use workflow management and online tools. Have periodic workshops with teams. Apply online team-building if visits won't work.
Staff turnover	<ul style="list-style-type: none"> Learn to deal with staff turnover by means of pooled buffers. Establish long-term retention models. Periodically conduct employee engagement surveys.

Risk	Mitigation Actions
Poor supplier services	<ul style="list-style-type: none"> Establish a fixed price contract scheme with agreed supplier management and escalation processes. Evolve towards a partner model with the supplier. Train suppliers on required processes. During the ramp-up period, carefully train supplier management on escalation procedures and your own required quality level. Escalate carefully and step-wise and avoid the SLA hammer. Rigorously highlight insufficient quality, delays or lack of visibility.
Instability with overly high change rate	<ul style="list-style-type: none"> Review and sign-off of all requirements. Monitor and control the requirements change index.
Insufficient competencies	Establish and maintain a global competency and skill management process across the entire GSE organization
Wage and cost inflation	<ul style="list-style-type: none"> Distribute work across regions and anticipate wage increases. Carefully protect against supplier lock-in. Evaluate your own and suppliers' business models over future years.
Lock-in with supplier	<ul style="list-style-type: none"> Work with multiple partners. Distribute critical knowledge Establish common processes and tools. Maintain back-up and recovery mechanisms.
Inadequate IPR management	<ul style="list-style-type: none"> Systematically train engineering and management on IPR. Establish and rigorously apply a strong policy on IPR protection. Encourage innovation on all GSE sites and promote patents.

risk of wage escalation. Similarly, the strong correlation observed between skill development and attrition might not be an universal phenomena, or there might be other overlying attributes impacting attrition more strongly. We recommend that organizations make an internal analysis to fine tune their approach.

As a general rule for risk identification in a specific GSE environment, we recommend setting up undesired scenarios, evaluate their probability to occur and decide for some 10–20 of those scenarios to take dedicated mitigation action. A majority is mitigated inside the global development project (e.g., common tools), while only a few must be part of the corporate risk strategy (e.g., handling supplier defection). Organizations should not worry about the number of 10–20 scenarios. They repeat in each of the organization's respective GSE projects and will build a kind of checklist with dedicated and organization-specific mitigation strategies that are reused in each new project.

Our empirical research provides data from a longitudinal study across close to hundred telecommunication projects and products of different size and managed either in captive or distributed mode. We have showed with this data that specifically early indicators, such as requirements change rate, early defect removal and skill level, key risk related to project performance and attrition can be effectively mitigated.

Risk mitigation does not come for free. It depends on process maturity, technology complexity and managerial competences additional cost for mitigation must be foreseen. Each of the following risk mitigation activities accounts for a 5–10 % increase to project cost [5,6,11]:

- > IT infrastructure, global tools licenses.
- > Distributed project management and progress control.
- > Coordination and interface management.
- > Liability coverage, legal support.
- > Training, knowledge management, communication.
- > Supplier and contract management.

This being said, initial saving potentials will be substantially reduced if organizations are immature in their processes and in GSE management.

Fragmented tasks handled in several sites combined with inadequate engineering and management processes not only ruin the GSE business case but are to our experience the number-one-reason for cancelled GSE engagements across industries.

Organizations on CMMI maturity level 1 or 2 should not expect that global software engineering would yield much benefits. Instead it will reveal major deficiencies in processes and workflow, which create all type of difficulties, such as insufficient quality, delays, additional cost, cancelled offshoring contracts, demotivated workforce in both places (previous and new), and many more. The only viable alternative for such low-maturity organizations is to ramp-up the own processes before proceeding with GSE.

Looking to these risk patterns, there is two problems with GSE: Getting started and keeping going. However, with the needs, the rewards and the mitigation patterns that we have showed here, you will translate risks to chances and opportunities which is what they should be seen.

Translation of a German publication in Automobil Elektronik, 5/2011

References

- [1] Aspray, W., F. Mayadas, M.Y. Vardi (ed.): Globalization and Offshoring of Software – A Report of the ACM Job Migration Task Force, Association for Computing Machinery, 2006, online at www.acm.org/globalizationreport/
- [2] Duke University and Booz Allen Hamilton: Next-Generation Offshoring: The Globalization of Innovation, 2007, <https://offshoring.fuqua.duke.edu/report.jsp>
- [3] Ebert, C.: Global Software Engineering. IEEE Computer Society Books, Los Alamitos, USA, 2006. http://www.computer.org/portal/pages/ieeecs/ReadyNotes/ebert_abstract.html
- [4] Key Highlights of the IT-BPO sector performance in FY 2007-08 (of India). NASSCOM report. www.nasscom.org.
- [5] Ebert, C. and R. Dumke: Software Measurement. Springer, Heidelberg, New York, 2007.
- [6] IDC: The Early Termination of Outsourcing Contracts, 2007, <http://www.idc.com/getdoc.jsp?containerId=CA11507>
- [7] Michael Corbett & Associates: The [annual] Strategic Outsourcing Study. 2006, Can be ordered at: www.corbettassociates.com
- [8] Herbsleb, J.D. et al: Distance, Dependencies, and Delay in a Global Collaboration. Proc. ACM Conf on Computer-Supported Cooperative Work, ACM Press, NY, 2000.
- [9] Cramton, C. D. and S. S. Webber: Relationships among geographic dispersion, team processes, and effectiveness in software development work teams. Journal of Business Research, vol. 58, pp. 758-765, 2005.
- [10] Krishna, S., Sahay, S., and Walsham, G. Managing cross-cultural issues in global software outsourcing. Communications of the ACM, Vol. 47, No. 4, pp. 62-66, 2004.
- [11] Corporate Executive Board: Managing the Global R&D Function. Case profiles in governing and coordinating dispersed R&D. Research & Technology Council, June 2004.
- [12] Ebert, C. and P. DeNeve: Surviving Global Software Development, IEEE Software, Vol. 18, No. 2, pp. 62-69, Apr. 2001.
- [13] O'Hara-Devereaux, M. and H. Johansen: Global work: Bridging distance, culture and time. San Francisco, CA: Jossey_Bass, 1994.
- [14] Boehm, B.W. Software Risk Management: Principles and Practices. IEEE Software 8, 1 (1991), 32-41.

Company profile

Vector Consulting Services is the leading consulting company to improve technical product development. Numerous clients from automotive, aerospace, medical, ICT and transport embark on our professional solutions. Objective-driven improvements combined with pragmatic implementation are what we stand for. Functional safety, process improvement, efficiency increase – our know-how provides our customers with measurable competitive advantages. The consulting branch of the globally active Vector Group, we support our clients worldwide with powerful consulting solutions that cover the entire product life-cycle and its related processes and tools.

www.vector.com/consulting

Dr. Christof Ebert

is managing director and partner at Vector Consulting Services GmbH. He is helping clients worldwide to improve technical product development and to manage organizational changes. Prior to that, he held engineering and management positions for fifteen years in telecommunication, IT, aerospace and transportation. As a business consultant and author of several books he has influenced numerous companies. Dr. Ebert lectures at the University of Stuttgart, serves on the editorial committees of several journals and is a SEI authorized CMMI Instructor.
Contact him at christof.ebert@vector.com

Bvs Krishna Murthy

Alcatel-Lucent India
94/95 Thiruvika Ind. Estate, Guind,
600032 Chennai, INDIA
Bvs.Krishnamurthy@alcatel-lucent.com

Namo Narayan Jha

Alcatel-Lucent India
94/95 Thiruvika Ind. Estate, Guind,
600032 Chennai, INDIA
N_N.Jha@alcatel-lucent.com



►► Objective-Driven Improvement

You set the targets.
We ensure you hit them.

Vector Consulting Services is the preferred partner for improving technical product development.

Solutions for you:

- > Functional safety
- > System, HW and SW engineering
- > Crisis and interim management
- > Distributed development
- > Processes, models, tools
- > Efficiency improvement
- > Organizational change management

Your goals are our goals.

Consulting to us means result-driven implementation together with our clients.

Your success is our own success. This is the yardstick we want to be measured.

► More information: www.vector.com/consulting

More Information!



Visit our Website for:

- > News
- > Products
- > Demo Software
- > Support
- > Workshops
- > Contact Addresses

www.vector.com

