Technical Foundations for the Development of Automotive Embedded Systems

Jörn Schlegel
Hasso-Plattner-Institut an der Universität Potsdam
joern.schlegel@hpi.uni-potsdam.de

ABSTRACT

Building an automobile today is a very complex process. Various computer hardware and software components have to work together to ensure a safe and comfortable drive. The customers can choose from a wide range of customizing options and automobile models to finally buy *their* automobile. How this effects the number of features in an automobile is described in *Section 1*, while how this increases the requirements for production is described in *Section 2* and *Section 3* of this report.

For the manufacturers of automobiles this means, they have to handle an already large and still growing number microcontrollers and software components. While a growing number of features in automobiles allow them to strengthen their brand and sell more automobiles, a growing number of hardware components means growing weight and production cost. Both means automobiles become more expensive and less of them can be sold. Another problem is to integrate new features successfully, regarding hardware and software interoperability with already existing systems. This is shown in more detail in *Section 4* and *Section 5* of this report.

To overcome these problems either the used computer hardware has to be unified or the used software has to be standardized to allow faster integration. Unifying hardware means using generalized hardware which is at the moment to expensive to use it for all systems in a car. Standardizing software currently is a much more promising approach and therefore is pursued by the Automotive Open System Architecture Consortium, with already promising results. *Section 6* of this report clarifies what the Automotive Open System Architecture Consortium achieved so far.

Keywords

Automobile, Software, AUTOSAR, Embedded Systems

1. INTRODUCTION

Automobiles are supposed to be lightweight, safe, and comfortable. Being lightweight is important while fossil fuels become more and more expensive. Because every 100kg an automobile weighs less, mean a reduction of fuel consumption by 0.5 litres [1]. This reduction of fuel consumption would lead to downsized motors and less CO_2 emission. These factors lead to a reduced total cost of ownership.

Making an automobile safer while reducing its body weight at the same time makes it necessary to use different materials for the bodywork and to use additional safety measures. These safety measures maybe passive systems as advanced designs of parts like the fascia and pedals to reduce the probability of injuries to driver and passengers. Used measures in the design include a rounded shape of the front of the glove compartment and predefined breaking points, which cause the parts to give way when knees or feet of the driver or the front passenger push against them in case of a crash.

Active safety systems include measures, which become active before or during a crash. Measures, which are active before a crash occurs, are for example seatbelt reminders, speed limitation devices, lane departure warning systems, and electronic stability control (ESC) systems. These systems help to avoid crashes by keeping the automobile controllable and preventing dangerous automobile movements as skidding or leaving a lane unintended or reminding drivers to avoid potentially dangerous situations as driving too fast or without wearing a seatbelt.

Safety measures becoming active during a crash include airbags, seatbelt load limiters and active seats and head restraints. These systems are designed to reduce the risk of injury to automobile occupants. Airbags provide a soft, elastic surface, which reduces the impact force when an occupant hits hard parts of the automobile or prevents that impact completely. Seatbelt load limiters release a small amount of excess belt webbing in a serious crash to prevent the seatbelt from injuring the restraint person by applying too much force while reducing the person's inertial speed. Active seats and head restraints will, during a crash, move into a position in which the seated person is in an optimal position towards the airbags and the head restraint to minimize injuries by whiplash.

All active safety systems require the installation of additional hardware and often additional software as well. These additional parts include the acting parts and components to communicate with other systems of the automobile. Additional parts mean more weight, which leads to a higher fuel consumption. However, most customers and the German legislature demand safety and low fuel consumption.

Another goal contradictory to reduced automobile weight and less fuel consumption is the growing desire for more comfortable automobiles. Comfort features in automobiles include power windows, air conditioning, sound systems, video systems,

communication systems, and heated seats. All of these systems need hardware, which means more weight, and consume energy. The more complex systems need software as well.

The increased weight and energy needs lead to higher fuel consumption and so is contradictive to the need for lighter less consuming automobiles. Safety is another goal entertainment and communication systems are contradictory to. These systems can distract the driver and so make additional safety systems necessary.

A good indicator for more equipment in automobiles is the increase in weight of an average family automobile between 1993 and 2005 by 30% [2]. This increase in weight took place despite a decrease in weight of bodywork and engine. The weight gained is not only to be attributed to a growing number of mechanical parts but also to a growing number of electrical parts.

The average share of electronics in total vehicle value will rise from 20% in 2004 to 35% in 2010. The software share in this share of electronics will rise from 20% in 2004 to 38% in 2010. This means the total share of software in vehicle value will rise to 13% in 2010. This means software plays a role of growing importance in building an automobile.

Figure 1. An automobile cockpit built in 1978 (left) contains much less electronic than one built in 2008 (right).

Most of the components marked in the right picture are connected to at least one other component.

Taken from [10]

As a result, the mastery of software development processes is becoming crucial for automobile manufacturers. An important step towards this goal is software reuse. It would mean reduced development cost and time and better quality of software and thus better failure avoidance. The use of standardized software modules which either encapsulate basic or specialized functionalities is a requirement to achieve this goal. Nevertheless, the software mostly used today is very heterogeneous, because different suppliers use different software specifications and standards. Consequently, automobile manufacturers have to go to great lengths to integrate software of various suppliers into their automobiles successfully.

2. INCREASING NETWORKING

The increased amount of advanced electronic systems in latter-day automobiles does not only lead to a larger number of electronic control units but also to a higher total cable length used in each single automobile.

This increase in cable length stems from a growing need for communication between once separated components of the electronic systems used in automobiles. For example, the velocity of an automobile was only displayed on the speedometer in earlier days. Today the velocity can be evaluated by the electronic stabilization control system or the cruise control to keep the automobile driving in the direction and with the velocity, the driver intended. However, it can be also evaluated by the volume control of the radio or the power windows to ensure more comfort by closing the windows and adjusting the volume of the radio when driving at higher speed. *Figure 1* illustrates the increasing number of networking electronic components during the last thirty years.

The increased need for hardware as cables, electronic control units and the software used to control them can lead to up to 1800 meters of cable with a total weight of 30 kilograms and one gigabyte of software installed on over 70 electronic control units in a single automobile of the BMW 3 Series built in 2005.

Adding even more complexity to the networking hardware in a single automobile, different types of bus systems are deployed for different purposes. The most often used bus types are the Controller Area Network (CAN), the Local Interconnect Network (LIN) and the Media Oriented System Transport (MOST).

The Controller Area Network is an asynchronous, serial fieldbus system. Its error detection and confinement capabilities along with a high data rate of up to one Megabit per second make a bus system that can handle communication between safety critical real time applications. A typical use is communication concerning the engine or the transmission.

Where the flexibility and bandwidth of the Area Control Network is not needed, the cheaper Local Interconnect Network (LIN) is

used. The Local Interconnect Network is a serial fieldbus. It is based upon a time triggered single master / several slaves concept. A typical use is the networking in a single component like a door or a seat.

The Media Oriented System Transport (MOST) bus covers another area of application. It is based on an optical fibre bearer, which allows far higher data transfer rates than other bus technologies used in automobiles. The serial data transmission is used to transport media data like audio or video data or to connect multimedia devices.

An approach to reduce the quantity of computer hardware built into an automobile is the unification of the functions currently spread over several bus systems to allow the use of homogeneous hardware components. One prominent approach is the FlexRay System. It is developed to provide a higher data transfer rate, better failure safety and real-time ability. These qualities become necessary as more driver assistance systems are built into each new generation of automobiles. Flexray is a deterministic serial fault-tolerant fieldbus system, which is developed by FlexRay Consortium.

Another approach is being researched by BMW. This approach uses the Internet Protocol to handle all communication tasks generated in an automobile. The integration of WLAN and Ethernet for in and out of vehicle communication would be easy with this approach and is one incentive for the research. The number of bus lines and control units each could be reduced to five with this approach. There would be one control unit and one bus for each the powertrain, chassis, driver assistance, infotainment and comfort. [3]

This homogeneity of hardware comes at a price. The reduction of cables, control units, and thus weight requires the use of very advanced and flexible high-tech systems. These systems have a higher per unit cost than specialised components, which are used today. Deploying those flexible systems for simple systems like air conditioning has no advantages justifying the higher costs necessary. This is the mean reason why the construction of automobiles containing only unified computer hardware is not feasible without skyrocketing costs.

3. INCREASING VARIETY

Another reason for an increasing number of different computer hardware built by automobile manufacturers and suppliers is the increasing variety of automobile models offered by manufacturers. 1978 Audi offered three models thirty years later they offered 34 models.

Each of these models can be customized with several options. These options include diverse comfort and yes/no options like having a navigation system or a radio, an air conditioning or maybe a moonroof. However, they also include varieties for lots of the components built into an automobile. For example, the customer can choose the size of the rims and the engine, he can choose between different suspensions and transmissions. The type of radio and seats, the colour of the exterior and interior of the automobile, several assistance systems and safety equipment can be selected.

The basis to which these choices are added is formed by core components like the chassis and the body. This method is called customized mass production. All components are mass-produced and in the process of building and assembling the components, only small, standardized alterations have to be made. This keeps the production of the product cheap and allows the customer to customize the product he buys to a certain degree.

The introduction of highly customized mass production to today's automobile industry has led to the production of only a few identical automobiles per year.

Customized mass production of automobiles generates the need for similarly produced software. This is important in order to safe costs in software development. Each mandatory or optional hardware component controlled by software has to have an equalling software module in order to ensure problem-free integration and operation of the necessary software. A possible means to achieve this goal are Software Product Lines (SPL). A Software Product Line is a set of different software products all originating from the same basic software. Each product is a little different, customized to different requirements. Software products to be installed on similar hardware components, in terms of function and requirements towards the structure of software should be of the same product line. This ensures that software can be developed and deployed fitting the needs of each customized automobile model.

4. INCREASING COMPLEXITY

The complexity of building an automobile does not only rise because more different models and varieties are offered, but also because more computer hardware is built into each automobile. More computer hardware components do not only mean more communication effort and thus more cables, but also more parts which have to be developed, tested, built into an automobile and then can be damaged. Computer hardware parts being damaged are expensive to repair and often cannot be repaired at all but have to be replaced.

It is common practice that suppliers deliver their software on controller hardware. This means every function added to an automobile means an electronic control unit is added to that automobile. Even though more electronic control units mean more functions and more functions in each automobile equal to more sold automobiles, more hardware components mean less profit for automobile manufacturers. This is because each component built into an automobile has a constant unit cost added to the cost of development. This unit cost is the main reason why automobile manufacturers try to reduce the number of electronic control units and cables and the total cable length while they also try to increase the number of functions.

More functionalities can be added by software, given that the necessary sensors and actuators, as well as the required communication hardware is already present in the automobile. This means that two automobiles identical in hardware can have different features and characteristics. For example, two identical engines can provide different power output and consume different amounts of fuel if they are controlled by different software.

These new functionalities help to strengthen the brand image and therefore are heavily used in advertisements. If a new functionality can be implemented by software only without deploying new hardware components, the implementation of this feature would have no unit cost, but development cost only. This would be a huge advantage concerning not only the costs, but also the reduction of weight and the flexibility of customizability.

Features only requiring new software and using already existing hardware also have the advantage, that they can be developed and exchanged or updated faster than features hardcoded onto specialized hardware components. Today this is partly used in chip tuning. Here software controlled parameters for the engine performance are altered to achieve faster acceleration or a higher top speed. In the future error prone software parts could be exchanged, revised software components could be deployed as an update, or completely new features could be deployed without exchanging hardware components, just like updating a personal computer or a cellular phone today.

To effectively find and exchange faulty software two requirements exist. Firstly, the software has to be organized in functional modules. Each module should encapsulate one functionality, which can be replaced without altering other modules. Secondly, these modules have to be developed in a model driven approach. This would then enable model based fault tracing. The actual data of a vehicle could be compared to the expected standard data generated by the model, this would allow to find hardware and software errors faster.

5. HETEROGENEOUS SOFTWARE

The software used in today's automobiles is highly heterogeneous. This is the results of various suppliers developing software for the same automobile. Different developers use different software standards for diverging requirements.

They develop software for real-time systems, like airbags, which have to open ten to forty milliseconds after the detection of a crash, or the calculation of the fuel/air mixture, which has to be calculated twice per engine stroke. These systems have to guarantee very short response times and need to employ failure safety mechanisms.

On the other hand, suppliers develop non-real-time systems like control units for the fuel gauge or the CD changer. These systems are not safety critical and therefore do not have to meet as strict requirements regarding time and failure safety.

Furthermore, different suppliers develop different software components for different hardware. This includes different types of electronic control units as well as different types of buses.

This is a huge problem for the automobile manufacturers because they have to integrate all the software, which the suppliers usually deliver already installed on hardware, e.g. on electronic control units, into a single working system. Considering the amount of functionalities this is no trivial task and requires a lot of time and money for testing. A solution to the problem would be the unification of the computer hardware used. FlexRay offers one attempt at a solution by unifying the functionalities of different bus systems currently used.

Taking this thought a step further would mean to unify the hardware of electronic control units as well. This would lead to more flexible personal computer like hardware. The advantages would be the simplification of dynamic resource allocation and thus an improved hardware redundancy to ensure failure safety. If there were only identical electronic control units, software currently not in use could be swapped with software likely to be used out of their memory. Furthermore, if one electronic control unit running safety critical functionalities malfunctioned, the software could be loaded into another electronic control unit, ensuring a safe drive until the malfunctioning hardware can be replaced.

This solution to the problem of heterogeneous computer hardware and thus software is very unlikely in the near future. The main reason for this is that a few of the flexible systems needed to implement this solution cost far more than a lot of less flexible systems used today. Therefore, another solution is necessary. Because software has no unit cost, the idea to reduce complexity by using standardized software suggests itself. A prominent standardized automotive software architecture is the Automotive Open System Architecture (AUTOSAR).

6. AUTOSAR

The Automotive Open System Architecture provides common basic system functions, a modular design, standardized interfaces, and a good scalability within and across different product lines. To standardize common basic system functions is important to automobile manufacturers, because these functions do not contain innovations and so do not sell automobiles. Furthermore AUTOSAR eases the exchange and reuse of software functionalities by standardizing them. This greatly reduces time and costs for testing, development, and integrating those components. *Figure 2* shows an overview example.

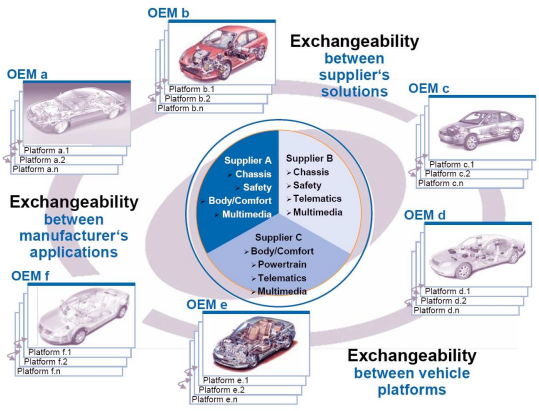

Figure 2. AUTOSAR manages complexity by exchangeability and reuse of software components.

Taken from [12]

Modular design means that clearly separated functional units are encapsulated. One advantage of this is, that functionalities can be swapped, if a better suited implementation was available. Another advantage is that functionality can be developed hardware independent up to a certain point and then the same functionality can be implemented for different sets of hardware.

Standardized interfaces make sure that components developed by different parties can communicate with each other, because the appropriate interface specifications are available to everyone. This has the advantage that single components can be replaced without rising the need to replace or modify other components.

Scalability within and across different product lines means for automobile manufacturers that they can use the same procedures to generate software no matter how many diverse implementations are needed. They can then use these software components for different models and just apply the steps and modules necessary to customize a specific product.

The Automotive Open System Architecture applies a software architecture to each electronic control unit, which is outlined in this section. It consists of AUTOSAR Software, the AUTOSAR Runtime Environment (RTE), and Basic Software. [8] An overview of the AUTOSAR ECU software architecture is depicted in *Figure 3*.

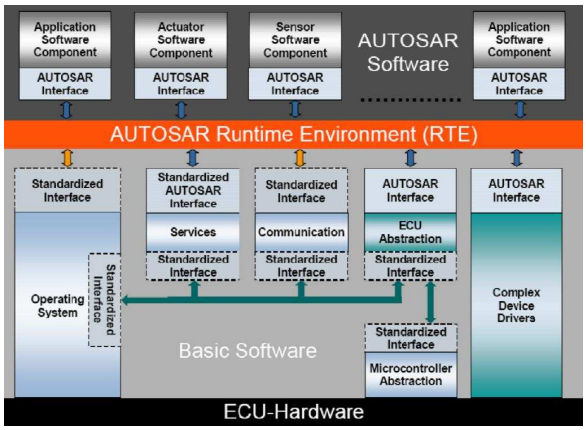

Figure 3. AUTOSAR ECU Software Architecture.

Taken from [11]

AUTOSAR Software contains all Software Components mapped to the electronic control unit. Each Software Component is atomic and encapsulates one software functionality. This means that each component can be exchanged without having to alter any other component. Software Components are connected to each other by ports. These come in two variants each consisting of a providing and a requiring port.

The first variant defines a set of operations that can be invoked. It works similar to a client/server interface. The providing port defines the operation available and triggers it without showing any implementation details to the outside. The requiring port on the other hand defines which operation he needs to be executed and triggers a request.

The second variant allows data-oriented communication. It works like a sender/receiver interface. The providing port sends data to the requiring port.

Specialized kinds of a Software Component are sensors and actuators. Those need to run on an electronic control unit, physically connected to the sensor or actuator hardware and are highly dependent on the function of the sensor or actuator.

In general, the implementation of Software Components is independent from the type electronic control unit it is mapped to. This allows a hardware independent programming of software functionalities and so contributes to the reuse of software. This is important to allow a wide range of automobile models all using the same software for identical functionalities.

The implementation is also independent from the location of other Software Components this Software Component has to interact with and the type of networking technology used to connect interacting Software Components, if they are on different hardware components. On advantage of these facts is that interacting functionalities can be stored on the same or on different electronic control units. This maybe important, if these functionalities were used in two different automobile models using a different hardware layout. In one model, two interacting functionalities could be stored on the same electronic control unit, whereas in the other model the same two functionalities are stored on different electronic control units.

The description of a Software Component contains operations and data element provided or required by the implemented functionality and information regarding the specific implementation, as for example which version is implemented. It also contains requirements on the infrastructure and resources needed by the Software Component. This contains for example, information about minimum data transfer rates and reaction times of connected hardware.

The AUTOSAR Runtime Environment handles the communication between Software Components, regardless if they are on the same or on different electronic control units. It is different for every electronic control unit due to different communication needs of different hard- and software. The configuration takes care of the actual communication paths, and has to be done for each different system. This is important to ensure that Software Components can be placed on different electronic control units, to satisfy the requirements of different hardware layouts. [9]

The Virtual Functional Bus (VFB) is the concept behind the Runtime Environment. It is a hardware and mapping independent means of virtual system integration. The software integration in much earlier design phases than is usual in today's development processes is one of its major advantages. Thus it allows earlier testing which safes a lot of money.

AUTOSAR Basic Software handles services, communication, the operating system, microcontroller abstraction, and electronic controller unit specific components. All those components include standardized interfaces to ensure interoperability. The services include diagnostic protocols and memory management. Communication contains input/output and communication management and frameworks like CAN and LIN, whereas AUTOSAR does not support MOST. The operating system is

based on the OSEK OS. The access to hardware is routed through the Microcontroller Abstraction Layer (MCAL). This avoids direct access to microcontroller registers from higher-level software. Thus, hardware independence for higher-level software is ensured. Electronic controller unit specific components consist of abstraction components, decoupling the software from the underlying hardware, and complex device drivers. Complex device drivers allow to access hardware directly, particularly for resource critical applications.

Using the information provided by the used Software Components, the Basic Software, Operating System, and Runtime Environment are configured. This means that only the components necessary to fulfil the requirements of the functionalities implemented in the Software Components are added to the system. By this means, the size of the whole system is kept to a minimum. For example, the SystemDesk RTE-Generator can be used to generate the code needed to integrate Software Components. [4], [7]

First, it will check the consistency of the used software architecture. If the architecture is consistent, the Basic Software, Operating System, communication stack [5], and RTE will be configured according to the information specified in the Software Components. After that, the actual Runtime Environment will be generated, determining the exact communication paths between the Software Components. [6]

7. SUMMARY AND CONCLUSION

The development of the automobile industry has led to more functionality available in today's automobiles. This functionality includes safety, driver assistance, and comfort systems. Most of these systems require additional computer hardware and software, which suppliers often deliver as computer hardware components including software for a very specific function within an automobile.

More electrical components delivered by different suppliers pose a growing difficulty for automobile manufacturers. This problem results from various requirements the components have, making the integration of all components into a working system difficult and time consuming. Another aspect of this problem is the weight and cost of an automobile growing with each micro controller and cable build into it. More weight leads to more fuel consumption, and higher production costs mean automobiles that are more expensive. Both effects lead to less sold automobiles.

The manufacturers cannot reduce the number of functionalities offered to the customers, because functionalities sell automobiles. Some functions are even declared mandatory by the lawmaker. Besides the growing number of features per automobile, today's customers demand individuality for the products they buy. This led to customized mass production further complicating the process of building automobiles. Furthermore, not only building a single automobile has become more complicated, but today a wide range of different automobile models are offered to the customers to satisfy their need for automobiles tailored to their liking.

One idea to reduce this complexity is to unify the hardware used, so that hardware components could be plugged together without having to consider lacking interoperability. This idea is not feasible today, because flexible hardware, as for example FlexRay, costs a lot more than specialized components. This is true even though the number of required specialized computer hardware components is larger, than the number of more general components required providing the same functions would be.

Another idea to reduce the cost of integrating components into a complex system is standardizing the used software. The Automotive Open System Architecture (AUTOSAR) provides a standard how software functionality should be specified. The main advantage is saving time and cost in system integration, because the basic software and communication structure is configured and added based on the requirements of the software implementing automobile functionality. This frees the automobile manufacturers from working on the basic software and leaves them with more money and time to invest in developing selling features. *Figure 4* shows more concepts the AUTOSAR provides.

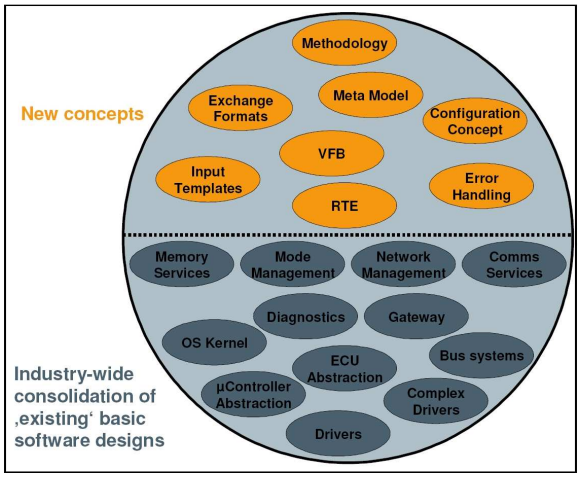

Figure 4. Technical scope of the AUTOSAR standard.

Taken from [13]

AUTOSAR provides the chance to create a mass market for automotive basic software. This mass market would exist, if a large number of companies used AUTOSAR, and would result in a significant in prices for automotive software and thus in the manufacturing costs for automobiles.

REFERENCES

[1] R. Kötz, Ph. Dietrich, M. Hahn, F. Büchi. Paul Scherrer Institute, Villingen, Schweiz. „Supercaps – Eigenschaften und Fahrzeuganwendungen“. VDI-Berichte Nr. 1874, 2005, S. 175 – 188.

[2] Oliver S. Kaiser, Dr. Heinz Eickenbusch, Dr. Vera Grimm, Dr. Dr. Axel Zweck. Zukünftige Technologien Consulting, VDI Technologiezentrum GmbH, Düsseldorf. „Zukunft des Autos“. Zukünftige Technologien Nr. 75, Januar 2008, ISSN 1436-5928.

[3] Wolfgang Pester. “Kabelbäume im Auto vor dem Abholzen”. In: VDI-Nachrichten 16.11.2007

[4] Dr. rer.-nat. Stichling. „Autosar Run-Time-Environment effizient generieren”. In: AUTOMOBIL-ELEKTRONIK, Oktober 2007.

[5] Gosda, Johannes. AUTOSAR Communication Stack. 2009.

[6] Hebig, Regina. AUTOSAR Methodology & Templates.2009.

[7] Wätzold, Sebastian. Modeling and Development of AUTOSAR using SystemDesk. 2009.

[8] Warschofsky, Robert. AUTOSAR Software Architecture. 2009.

[9] Naumann, Nico. AUTOSAR Runtime Environment and Virtual Function Bus. 2009.

[10] Stephan Reichelt. I/AEV-2, FlexRay und AUTOSAR. 14. November 2007.

[11] Moessinger, Juergen. AUTOSAR – The Standard for Global Cooperation in Automotive SW Development. May, 2008.

[12] Moessinger, Juergen. AUTOSAR – The Standard for Global Cooperation in Automotive SW Development. July, 2008.

[13] Harald Heinecke, Jürgen Bielefeld, Klaus-Peter Schnelle, Nico Maldener, Helmut Fennel, Oliver Weis, Thomas Weber, Jens Ruh, Lennart Lundh, Tomas Sandén, Peter Heitkämper, Robert Rimkus, Jean Leflour, Alain Gilberg, Ulrich Virnich, Stefan Voget, Kenji Nishikawa, Kazuhiro Kajio, Thomas Scharnhorst, Bernd Kunkel. AUTOSAR – Current results and preparations for exploitation. 7th EUROFORUM conference ‘Software in the vehicle’ 3-4 May 2006, Stuttgart, Germany.