

Specification of Functional Interfaces for Innovative Vehicle Functions based on a UML Meta Model

Jörg Petersen, Torsten Bertram
Gerhard-Mercator-University
Institute of Information Technology, Institute of Mechatronics and System Dynamics
47048 Duisburg, Germany
[joerg.petersen@uni-duisburg.de]

Pio Torre Flores, Andreas Lapp, Jürgen Schirmer, Dieter Kraft
Robert Bosch GmbH, FV/SLN
70442 Stuttgart, Germany

Wolfgang Hermesen
ASSET Automotive Systems and Engineering Technology GmbH, SF/EAS1
70442 Stuttgart, Germany

Abstract: Increasing demands concerning safety, economic impact, fuel consumption and comfort result in more and more complex vehicle functions realized by interconnected mechatronic system structures. The development of networked electronic control units (ECU) as the most frequent mechatronic applications contains three core aspects: the development of the (control) functions itself and their realization in hardware and software as embedded systems. A coordinated, systematic and concurrent function, hardware and software development process including co-engineering and simulation environments in early development phases requires a formalized description to improve the clearness of specifications, decrease contradictions and increase information density. A Unified Modeling Language (UML) meta model can be used for a mapping of automotive domain specific models into UML models with constraints formalized by Object Constraint Language (OCL) expressions. Specified functional interfaces together with a hierarchical decomposition of the system are basis for the system design and implementation supporting aspects like re-use, exchangeability, scalability and distributed development. Result is an open and modular system architecture called CARTRONIC. This paper gives an overview of the system architecture according to CARTRONIC and its main aspects and presents the UML meta model with the focus of a detailed description of the decomposition and the functional interfaces.

1 Motivation and Introduction

Increasing demands concerning safety, economic impact, fuel consumption and comfort result in more and more complex vehicle functions realized by interconnected mechatronic system structures. In order to fulfil these partly diametrical demands, the creation of a car wide web including the up to now usually independently working systems in a vehicle seems to be an appropriate method of resolution. Within this web embedded systems as realizations of electronic control units (ECU) are today most

frequent mechatronic applications. In upper class vehicles already over 80 ECUs control and monitor more than 170 control functions supported by approximately 450 electric servo motors. In all vehicles components stem from different automobile manufacturers and suppliers and are interconnected to an uncountable variety of system compositions. Therefore, the system and the function suppliers respectively have to guarantee that the function is able to fulfil the required specification, even in case of a distribution on several, interconnected control systems. Well carried out requirement specifications include all communication relationships between the individual components in the vehicle, in particular sensors, actuators, communication hardware and ECUs. A detailed specification improves the clearness of requirements, decreases contradictions, increases the information density and form well standardized interfaces within this web.

Functional interfaces in this context describe the ability of a component to interpret and realize communications between complex functional components. Functional components realize tasks, and the exchange of information ensures the execution of tasks being too complex for a single component. Therefore, implicitly a functional interface describes the activation of the function, the (continuing) realization of the underlying task or the generation of needed information. It is closely connected to the physical data required by the function to develop.

To ensure quality, reliability and safety a high integrated, computer-assisted, concurrent function, hardware and software development process has to be defined. Actually several groups of car manufacturers and suppliers as well as universities are working on architecture descriptions in the automotive domain, e.g. [AW01], [Be01], [Ho01], [To01]. In the automobile industry the Unified Modeling Language (UML) as an international standard passed by the Object Management Group [OMG99] is widely spread, last but not least cause of its support by diverse commercial tools. Even if it lacks semantic preciseness, the UML includes meta modeling facilities as well as the Object Constraint Language (OCL) suited to define constraints in object-oriented models to increase precision on this models. This justified the specification of functional requirements and interfaces by UML models based on a domain specific CARTRONIC UML meta model. Together with constraints formalized by OCL expressions this finally results in the definition of appropriately detailed functional interfaces.

In this paper a UML meta model for a structural automotive domain architecture with respect to the functional requirements [Be98] will be introduced specially focussed on functional interfaces. This CARTRONIC function architecture as one part of an open and modular CARTRONIC system architecture [La01] is based on a structuring concept for all control systems in a vehicle. It will be briefly introduced in section 2 together with an extract of a torque control use case as an example. Section 3 describes the mapping of CARTRONIC model elements of the function architecture into the UML meta model. In the first part the static hierarchic structure of components will be discussed, in the second one the allowed communications between components. The third part focuses on the specification of (real) time requirements for (control) function parameters. The simplified torque control from section 2 provides examples in each part. The paper closes with a short summary and some remarks on current work.

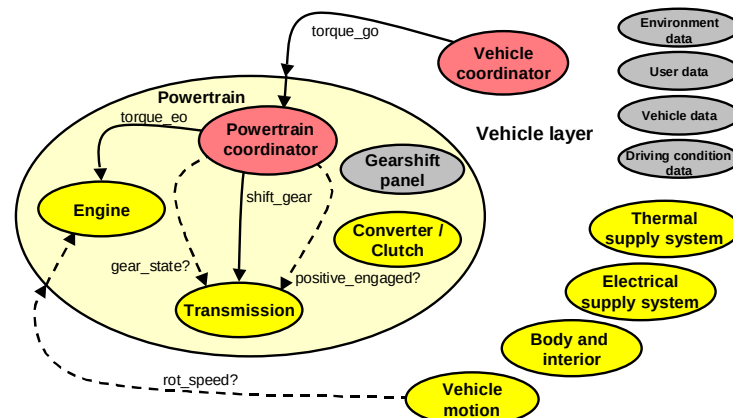


Fig. 1: An extract of torque control at a high level of abstraction as an example for a CARTRONIC function structure with refined components Vehicle layer and Powertrain.

2 The Automotive Domain Structuring Concept CARTRONIC

In the following CARTRONIC stands only for the aspects regarding the function architecture. Used for the structuring of tasks in the phase of the requirement analysis components and communication relations are the elements of the CARTRONIC function architecture. Components provide restricted, independently resolved tasks and exactly defined physical interfaces. They do not automatically represent a physical unit in the sense of a constructing element or a control unit, but have to be understood as a logical unit. Components can be subdivided into sub-components representing the concepts of abstraction and encapsulation. Seen from outside, i.e. from the point of view of neighbor components, a component of a function structure has to be interpreted as a system. Seen from inside, i.e. from the point of view of the sub-systems or the sub-components, the original component is only an enclosure integrating the entirety of their sub-systems.

The interaction of components is modeled by three different types of communication relations: order (with response), request and inquiry (with notice). An order is characterized by the obligation to be executed by the instructed component. If the order is not executed, the receiving component has to provide the orderer with a response why the order could not be fulfilled. A request describes the demand of a source component, so that the target component initiates an action or realizes a function. Nevertheless, in contrast to the order the request is not obliged to be fulfilled. This communication relation is used e.g. for the realization of competing resource demands (energy or information). The inquiry is used to get information needed for the fulfilment of an order. If a component is not able to provide the requested information it can notify the inquiring component accordingly.

Structuring and modeling rules restrict communication relations concerning the possible communication partner within the function architecture of the entire vehicle. A main

idea of the structuring concept is the hierarchical flow of orders. Always exactly one component exists in a refinement receiving all incoming orders and called 'entry component'. Communications are distinguished between components being realized on the same level of abstraction or on higher or lower ones and therefore restricted. The forwarding from one system into another in the respective refinement is controlled. Patterns combining components and communication relations describe the execution of special and repeatedly occurring tasks, e.g. electrical and mechanical as well as thermal energy management. Final result are modular, hierarchic and expandable function structures with a high level of responsibility of each individual component.

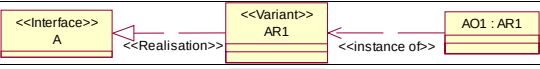
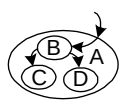
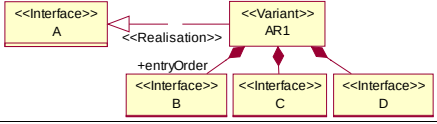
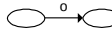
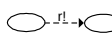
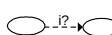
In figure 1 the Vehicle layer on the highest abstraction layer contains one coordinator (Vehicle coordinator), the five components Powertrain, Vehicle motion, Body and interior, Electrical supply system and Thermal supply system with mainly operational tasks, and four information providers Environment data, Vehicle data, User data and Driving conditions data. Since the complexity of component Powertrain is high, it is structured into a set of yet smaller even more manageable components. Interactions of all components are coordinated by Vehicle coordinator and Powertrain coordinator in the refined component Powertrain. An extract of simplified torque control is specified on an abstract level by the sub-components of the refined Powertrain, i.e. Engine, Transmission, Converter/Clutch, Gearshift panel and Powertrain coordinator. The vehicle propulsion demand is determined from the various inputs as for example the driver's request, cruise control or torque demands from other systems. The different propulsion demands are collected and managed accordingly to their priority by the Vehicle coordinator. The resulting torque demand is then coordinated with gearbox interventions and transformed into an engine torque demand by the Powertrain coordinator. The specified communication relationships are:

- the order torque_go (to provide a certain torque at the gear output) is given by the Vehicle coordinator to the Powertrain and forwarded to the 'entry component' Powertrain coordinator,
- the order torque_eo (to provide a certain torque at the engine output) from the Powertrain coordinator to the component Engine,
- the order shift_gear (to change the transmission gear) is given by the Powertrain coordinator to the component Transmission,
- the inquiries gear_state? (present gear state) and positive_engaged? (transmission positive engaged) from the Powertrain coordinator to Transmission,
- as well as the inquiry rot_speed? (present rotational speed) from the Vehicle motion to the Powertrain which is transferred to the responsible component Engine.

3 A UML Meta Model for CARTRONIC

Table 1 summarizes the CARTRONIC modeling elements with their graphical notation. Their mapping into UML together with the underlying CARTRONIC UML meta model will be described in the following.

Tab. 1: CARTRONIC notation versus UML notation of the modeling elements

CARTRONIC	Notation	UML
Component		
System		
Enclosure		
Order		Operation with stereotype <<Order>> o of <<Interface>> A
Request		Operation with stereotype <<Request>> r of <<Interface>> A
Inquiry		Operation with stereotype <<Inquiry>> i of <<Interface>> A
Rule Type		Meta model with meta classes, relationships between them, multiplicities, and OCL expressions

3.1 The Static Hierarchic Structure

A CARTRONIC component A is mapped to an abstract interface class named A encapsulating the internal behavior of a component from their external visible functionality (e.g. <<Interface>> A in table 1). Interface classes are used to collect all operations specifying services of a component. An interface A can be realized by one or more different classes AR1, AR2, etc., reflecting automobile variants. To describe dynamic behavior objects like AO1:AR1 of this variant classes can be used in UML behavior diagrams. At the end of the software development process most variant classes are instantiated by singleton objects in the memory of an embedded system (e.g. one A, one B, etc.).

If a CARTRONIC component A is detailed, e.g. in table 1 in components B, C, D, in the UML model the respective variant class AR1 has UML composition relations to the interfaces representing B, C and D. Therefore, the hierarchic assignment of sub-components to a superior component is given by composition relations always connecting realizing classes with interface classes representing refined components.

Looking into the meta model all elements are inherited from an abstract root meta class CartronicModelElement. Abstract meta classes with stereotype <<AbstractElement>> are only used for structuring the meta model. Meta classes with (meta) stereotype <<Element>> are instantiated in CARTRONIC UML models and result in class and connector stereotypes. All types of component classes are derived from the abstract meta class Componentbase (fig. 2). For later extension the meta classes Interface and Variant are derived from the abstract meta classes CartronicInterface and CartronicVariant. The other meta classes will be introduced in next subsections.

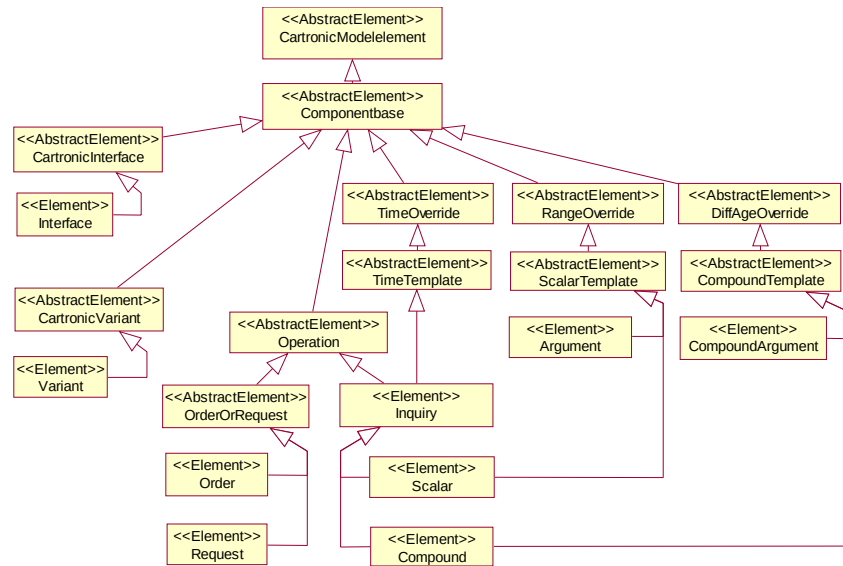


Fig. 2: UML meta classes for CARTRONIC components.

Connectors describe the semantic relationships between components. The meta classes Realisation and Composition are derived from the root meta class Connectorbase (fig. 3).

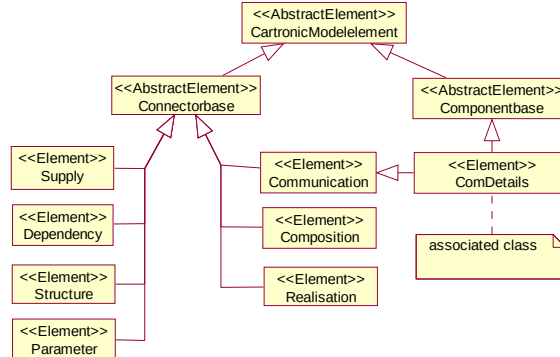


Fig. 3: Connector classes in the meta model.

Figure 4 summarizes allowed relationships between interface and variant classes; they are explained in more detail in [Pe01]. Figure 5 shows the refined component Powertrain from figure 1 mapped into the UML class <<Interface>> Powertrain realized by the <<Variant>> PowertrainR1. The five sub-systems of the component Powertrain, more exact the classes PowertrainCoordinator, Engine, Transmission, ConverterClutch and GearshiftPanel of type <<Interface>> are connected by (part-of) Compositions to the class <<Variant>> PowertrainR1.

A composition relation always connects a realizing class with an interface class of a sub-component.

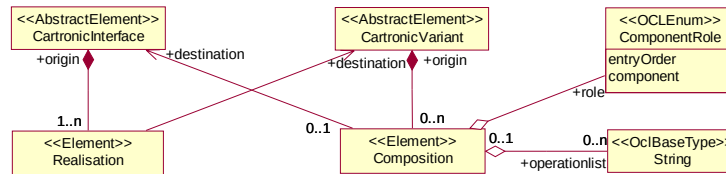


Fig. 4: Compositions and Realisations define the structure relationships of CARTRONIC models regarding CartronicInterface and CartronicVariant.

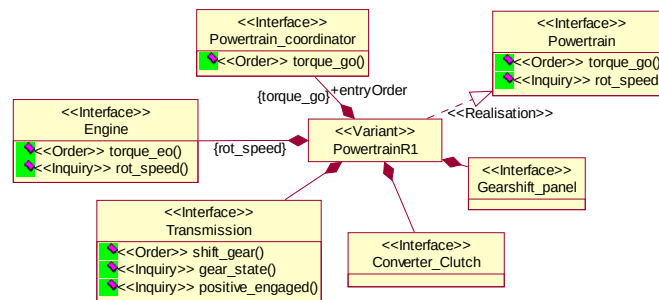


Fig. 5: The refined CARTRONIC component Powertrain from figure 1 modeled in UML.

3.2 The Communication Relationships

The receiving component of a CARTRONIC communication relation offers a functional service to the sending component. A UML operation in the UML interface class representing the receiving CARTRONIC component describes this service. The three different types of communication (tab. 1) are expressed by the stereotypes <<Order>>, <<Request>> or <<Inquiry>>, e.g. the UML interface of the component Powertrain in figure 1 offers the two operations <<Order>> torque_go and <<Inquiry>> rot_speed. The abstract meta class OrderOrRequest is used for a combined description of their parameters, and Operation in figure 2 generalizes all of them.

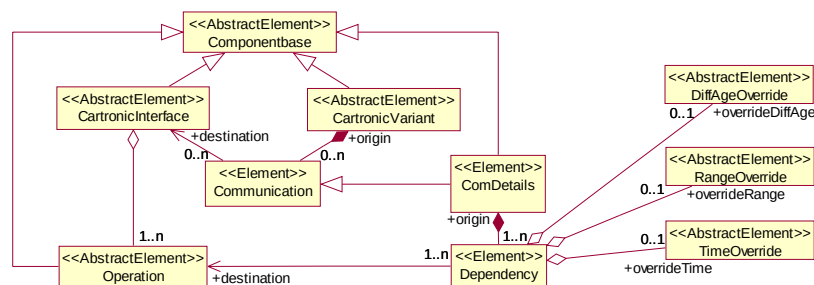


Fig. 6: Overview of CARTRONIC UML Communications.

Figure 6 shows, that each CARTRONIC operation is supplied by exactly one CartronicInterface, each interface class offers at least one operation. CartronicVariant is aggregated to the meta class Communication with multiplicity 0..n specifying that as much as required communications from a component are allowed (including none). Vice versa the CartronicVariant is an unique sender with role name origin for the Communication. A CartronicInterface is the unique receiver in the role destination of a Communication and each CartronicInterface may be recipient of arbitrarily many communication requests (multiplicity 0..n). The associated class <<Element>> ComDetails specifies, which of the offered functional services modeled by operations of the interface are used exactly, e.g. in figure 1 Vehicle coordinator only communicates torque_go to Powertrain. In later software development phases the associated classes have to be implemented completely by one of the (by means of Communications) connected classes or partly by both of them; this also includes the implementation of responses indicating not executed orders and notices analogous for inquiries.

3.3 Physical Data and Time Requirements

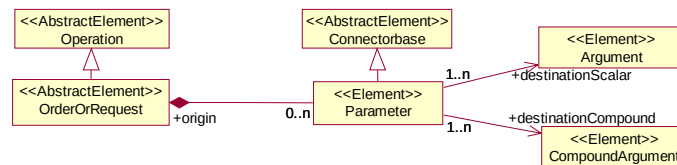


Fig. 7: CARTRONIC orders and requests have scalar and compound arguments as parameters.

Orders and requests have scalar or compound parameters (fig. 7). Each OrderOrRequest has (is connected to) 0..n Parameters, each Argument or CompoundArgument is used 1..n times. Argument and CompoundArgument are inherited from the abstract meta classes ScalarTemplate respectively CompoundTemplate (fig. 2). All parameters are physical data with the (from Componentbase inherited) properties name and description (fig. 8a). In the function development view ScalarTemplates have the properties unit, (base) type and the feasible range (fig. 8b). CompoundTemplates structure 2..n ScalarTemplates (fig. 8c). In later software development phases the responsible modules building structures of scalar data have to guarantee, that combined scalars are simultaneous. Using an underlying concept of relative time specifications the properties diffAge and diffAgeUnit specify the difference of their age in pairs not to be exceeded.

Inquiries are Scalars or Compounds (fig. 2); they do not have functional parameters. Their hybrid operational and data character as required and provided information is expressed by multiple inheritance of Inquiry and ScalarTemplate or CompoundTemplate. Additionally, Inquiry inherits from the abstract meta class TimeTemplate the two properties maxAge and maxAgeUnit (fig. 8d). This two

properties describe the (default) supplied actualization rate, i.e. which (real) time requirements the realizing class has to guarantee (rot_speed with 5 ms in fig. 9). If a requiring component needs a lower or stronger (real) time requirement, the default value specified in the supplied Scalar or Compound class can be overwritten by specifying in terms of the respective meta class ComDetails. The Dependency connecting the associated class ComDetails with Operation in figure 6 allows to override maxAge and range for Scalars (e.g. VehicleMotionR1_to_Engine in fig. 9 requires 1 ms actualization rate of rot_speed) and maxAge and diffAge for Compounds (presupposed that age units normally stay in the same order of magnitude).

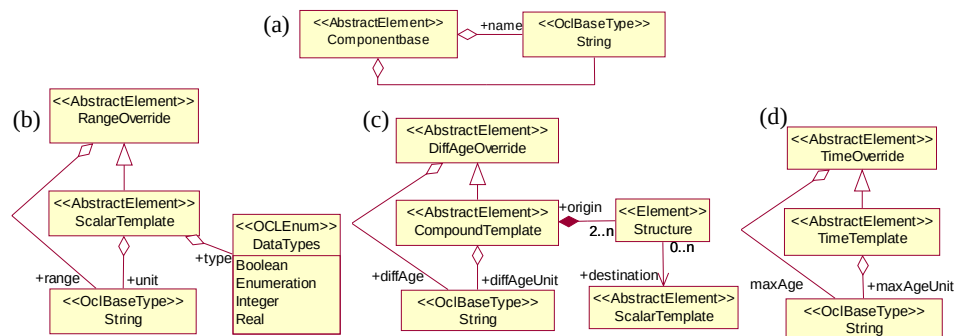


Fig. 8: Physical data properties required for (control) function development.

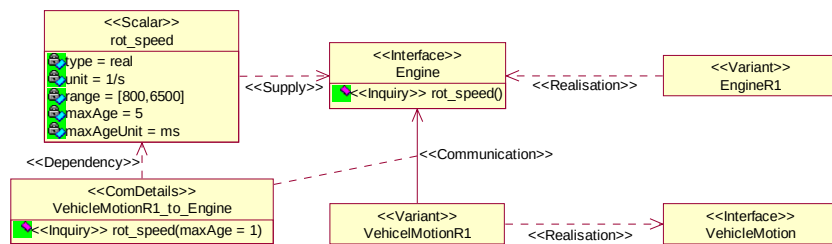


Fig. 9: <<Interface>> Engine supplies <<Scalar>> rot_speed as <<Inquiry>> rot_speed with a guaranteed actualization rate of 5 ms.

4 Summary, Current and Future Work

In the automotive domain CARTRONIC is intended to be an open and neutral concept regarding automotive manufactures and suppliers and it is especially useful in first function analysis phases. A formalized and computer supported model of specified functional requirements can be achieved by mapping CARTRONIC components and communication relationships into UML. Specifying a model for the automotive domain allows a computer supported, coordinated system development by independent distributed teams. Choosing UML gives the benefit of being tool independent, additionally defined OCL constraints checked outside the tools give the possibility of first automatically done consistence checks.

The presented work is implemented in a commercial UML tool. Independent from this tool an OCL-Checker works on CARTRONIC XML files exported from CARTRONIC UML models. This prototype is written in Java; it is extended from an older version [He00].

Current work includes evaluating a second UML tool, refinement of the UML meta model and the associated XML schema, vice versa structure and information exchange to a specialized function development and simulation tool as well as next steps in the software development process. From a single CARTRONIC UML model several different design models may be created depending on several realization possibilities for provided component interfaces. Designing software components, the interface classes may result in export interfaces, the collection of all communication relationships to such an interface class may lead to the import interface of the software component [La01]. Overwritten requirements in the associated classes can be used for calculating overall real time requirements, implementing associated classes and related compound inquiries may result in grouping information packets with respect to the underlying bus system.

References

- [AW01] AIT-WOODDES. <http://wooddes.intranet.gr/>.
- [Be01] Beeck, M. v. d.; Braun, P.; Rappl, M.; Schröder, Chr.: Modellbasierte Softwareentwicklung für automobilspezifische Steuergeräte. VDI Tagung Elektronik im KFZ, Baden-Baden, 2001.
- [Be98] Bertram, T.; Bitzer, R.; Mayer, R.; Volkart, A.: CARTRONIC - An open architecture for networking the control systems of an automobile. SAE International Congress and Exposition. Detroit/Michigan U.S.A., 1998.
- [He00] Hermesen, W.; Neumann, K. J.: Application of the Object-Oriented Modeling Concept OMOS for Signal Conditioning of Vehicle Control Units. SAE International Congress and Exposition, Detroit/Michigan U.S.A., 2001.
- [Ho01] Hofmann, P.; Fasolt, J.; Geretschlager, P.; Sakretz, R.; Wohlgemuth, F.: Automotive UML - eine neue objektorientierte Entwicklungstechnik. Published in 'Elektronik Sonderheft: Automotive', 2001, pp. 88-95.
- [La01] Lapp, A.; Torre Flores, P.; Schirmer, J.; Kraft, D.; Hermesen, W.; Bertram, T.; Petersen, J.: Softwareentwicklung für Steuergeräte im Systemverbund – Von der CARTRONIC-Domänenstruktur zum Steuergerätecode. VDI Tagung Elektronik im KFZ, Baden-Baden, 2001.
- [OMG99] Object Management Group (OMG): OMG Unified Modeling Language Specification V1.3, <http://www.omg.org/UML/>, 1999.
- [Pe01] Petersen, J.; Bertram, T.; Lapp, A.; Torre Flores, P.; Kraft, D.; Schirmer, J.; Hermesen, W.: A UML Meta Model for Specifying Functional Requirements of Mechatronic Components in Vehicles. Object-Oriented Modeling of Embedded Realtime Systems (OMER) 2 Workshop, Herrsching (Ammersee), 2001.
- [To01] Torre Flores, P.; Lapp, A.; Hermesen, W.; Schirmer, J.; Walther, M.; Bertram, T.; Petersen, J.: Integration of the ordering concept for vehicle control systems CARTRONIC into the software development process using UML modeling methods. SAE International Congress and Exposition, Detroit/Michigan U.S.A., 2001.