A UML Meta Model for Specifying Functional Requirements of Mechatronic Components in Vehicles

Jörg Petersen, Torsten Bertram
Gerhard-Mercator-University, Dept. of Computer Science and Mechatronics, 47048 Duisburg, Germany
[joerg.petersen@uni-duisburg.de | bertram@mechatronik.uni-duisburg.de]

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

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

Abstract
The demand for innovative vehicle functions at reasonable costs leads to more and more complex vehicle functions as well as more and more interconnections between mechatronic components in vehicles. Therefore a detailed specification and semantic foundation of all (logical) functionalities is required. A formalized description improves the clearness of specifications, decreases contradictions, increases information density and supports early simulation cycles in the overall software development process. This paper presents a Unified Modeling Language (UML) meta model used for a mapping of automotive domain specific models into UML models added by constraints formalized via Object Constraint Language (OCL) expressions. The meta model and first domain models are implemented in a commercial tool, a prototype checker of OCL expression is realized in Java.

1. Motivation and Introduction
The demand for innovative vehicle functions resulting in more complex and interconnected system structures within constant product cycles requires automated and computer based development processes with first high level simulation possibilities in early design steps. Electronic control units (ECU) are the most frequent mechatronic applications and today are mostly realized as embedded systems i.e. implemented in software. The development of such mechatronic systems includes three core aspects: the development of the (control) functions itself and their realization in hardware and software. Communication relationships between components are realized by parts of all these aspects. Therefore a concurrent function, hardware and software development process has to be defined which includes co-engineering and simulation environments in early development phases to meet these demands.

In upper class vehicles already over 60 ECUs exist controlling and monitoring more than 170 functions covering all control tasks in the vehicle. Individual components in the vehicle, in particular sensors, actuators, communication hardware and ECUs are usually supplied by various manufacturers. A detailed specification will be essential to guarantee the functionality but also to ensure quality, reliability and safety. Therefore a structural domain architecture with respect to functionality called CARTRONIC® has been developed [Bertram et al. 1998]. A formalized description improves the clearness of specifications, decreases contradictions, increases the information density and supports early simulation cycles in the overall software development process. The UML as an international standard passed by the OMG [OMG 1999] includes meta modeling facilities as well as the OCL. The OCL allows the definition of constraints on object-oriented models to increase precision on this models. Discussed features in this section are foundations for an automated, computer based development process.

Actually several groups of car manufacturers and suppliers as well as universities are working on architecture descriptions in the automotive domain by means of UML, e.g. [AIT-WOODDES], [AUTOMOTIVE-UML], [FORSOFT], [Torre Flores et al. 2001]. This paper presents an approach to specify functional requirements using UML models based on a UML meta model together with constraints formalized by OCL expressions. Section 2 gives a short overview of the structural CARTRONIC® domain architecture with respect to functionality being base concept of the mapped UML models together with an example for traction control. A UML meta model for formalizing the mapping of CARTRONIC® models into UML models is presented in section 3. Section 4 closes with a short summary and some remarks on current work on an automatic checker for OCL constraints.

2. A Short Overview of the CARTRONIC® Domain Architecture
CARTRONIC® is a structuring concept for all control functions and systems in a vehicle. The concept comprises structuring and modeling rules and a modular, hierarchic and expandable structure which complies with this rules. It can be applied to different types of vehicles and ECU configurations, it is
intended to be open and neutral regarding automotive manufactures and suppliers, and it is especially useful in the analysis phase. Functions and systems of different origin, different automobile manufacturers and suppliers with standardized interfaces can be interconnected to a system compound.

A main idea of the structuring concept is the hierarchical decomposition and flow of orders. Summarizing, the following features of a function structure, developed in accordance with the structuring and modeling rules, can be listed:

- defined, consistent structuring and modeling rules on all levels of abstraction,
- hierarchical decomposition of the system structure,
- hierarchic flow of orders with each component being assigned to only one orderer,
- high level of individual responsibility for each component,
- control elements, sensors and estimators are equal information providers,
- encapsulation, so that each component is only as visible as necessary and as invisible as possible for other components,
- realization independent.

Figure 1: Traction control with a simplified gearshift process as a CARTRONIC® function structure

Components and communication relations between these components are main elements of the CARTRONIC® function structure. A component does not automatically represent a physical unit in the sense of a constructing element or an ECU, but has to be understood as a conglomerate of logical functions. 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 which principally might be subdivided into subsystems, sub-components, etc. Seen from inside, i.e. from the point of view of the subsystems or the sub-components, the original component is only an enclosure which integrates the entirety of all subsystems. Figure 1 shows a traction control as an existing example of a CARTRONIC® function structure modeled out of a global view of the powertrain functionalities.

Mainly, three different types of communication relations exist: orders, requests and inquiries. An order is characterized by the obligation to be executed by the receiving component. If an order can not be executed the receiving component has to give a response to the ordering component why this order could not be fulfilled. A request expresses the demand of a source component to the target component to initiate an action or realize a function. Nevertheless, in contrast to an order there is no obligation for the receiving component to fulfil a request. This communication relation is used e.g. for the realization of competing resource demands (energy or information). The inquiry is employed to get information needed for the execution of an order or a request. If a component is not able to provide the requested information, it can notify the inquiring component accordingly. Table 1 summarizes the CARTRONIC® elements with their graphical representations.

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Logical functional unit</td>
<td><img src="image1" alt="Component" /></td>
</tr>
<tr>
<td>System</td>
<td>A system consisting of several components respectively subsystems (view from inside to outside)</td>
<td><img src="image2" alt="System" /></td>
</tr>
<tr>
<td>Envelope</td>
<td>Detailed component delegating communications to inner components (view from outside to inside)</td>
<td><img src="image3" alt="Envelope" /></td>
</tr>
<tr>
<td>Rule type</td>
<td>Structuring and modeling rules</td>
<td><img src="image4" alt="Rule type" /></td>
</tr>
<tr>
<td>Order</td>
<td>Order to a component with the duty to execute a function</td>
<td><img src="image5" alt="Order" /></td>
</tr>
<tr>
<td>Request</td>
<td>Request to a component to execute a function with no obligation</td>
<td><img src="image6" alt="Request" /></td>
</tr>
<tr>
<td>Inquiry</td>
<td>Requirement for some pieces of information</td>
<td><img src="image7" alt="Inquiry" /></td>
</tr>
</tbody>
</table>

Table 1: CARTRONIC® modeling elements
3. UML Meta Model Based Mapping to CARTRONIC® UML models

Figure 2 shows the mapping of the CARTRONIC® function structure of figure 1 into the CARTRONIC® UML model. In the following examples of this mapping will be discussed and the meta classes of the CARTRONIC® UML meta model are presented.

CARTRONIC® components are modeled by object instantiations of UML classes realizing interface classes which encapsulate the internal behavior of a component from their external visible behavior which is specified in the CARTRONIC® function structure. For example, the CARTRONIC® component Engine in figure 1 is represented by the classes <<Interface>> Engine and the realization <<Variant>> EngineR1 in figure 2. Object instances like EngineO1:EngineR1 in figure 3 ultimately implemented in ECUs represent realizations of CARTRONIC® components especially used in UML behavior diagrams. Interface classes collect all operations used to specify services of a certain component. The interface encapsulates the functionality which may be realized in different ways. Therefore <<Variant>> EngineR1 or <<Variant>> EngineR2 may be different possible realizations of the <<Interface>> Engine.

Figure 3: Example of mapping CARTRONIC® components into UML model elements

In the meta model all elements are inherited from an abstract root meta class <<AbstractElement>> CartronicModelelement (figure 4). The stereotype <<AbstractElement>> in contrast to the stereotype <<Element>> symbolizes that meta classes with this stereotype are not allowed to be
instantiated in CARTRONIC® UML models and mainly serve in structuring the meta model. Conceptually, a CARTRONIC® UML model forms a structural architecture of components and connectors with respect to functionality. The meta model formalizes semantic relationships of these components and connectors. Two principally different types of meta classes are derived from the meta root, the abstract meta classes <<AbstractElement>> Componentbase and <<AbstractElement>> Connectorbase. From Componentbase three abstract classes <<AbstractElement>> CartronicInterface, <<AbstractElement>> CartronicVariant and <<AbstractElement>> Operation are inherited. <<Element>> Interface and <<Element>> Variant are again derived meta classes, which can be instantiated in CARTRONIC® UML models as shown in figure 2. At first view this doubling in hierarchy seems to be superfluous but will be useful for later extensions.

**Figure 4:** The CARTRONIC® UML meta model element hierarchy

The receiving component of a communication relation offers a functional service to the sending component. To represent this, UML operations are assigned to the UML interface classes representing a CARTRONIC® component, e.g. in figure 2 the interface of the component Powertrain offers the operations <<Order>> torque_go and <<Inquiry>> rot_speed (compare figure 1). The type of communication will be specified by stereotypes <<Order>>, <<Request>> and <<Inquiry>>. In the meta model the different operation types are represented by three different meta classes <<Element>> Order, <<Element>> Request and <<Element>> Inquiry inherited from the abstract meta class <<AbstractElement>> Operation (figure 4).

**Figure 5:** Each interface class contains at least one operation.

The aggregation between CartronicInterface and Operation shown in figure 5 specifies that each interface class requires at least one offered operation. This can be expressed very efficiently by the multiplicity expression 1..* (not written multiplicities are 1 by default) and using the structuring abstract meta classes CartronicInterface and Operation.

**Figure 6:** Defined relationships between interface and variant classes
Figure 6 summarizes allowed relationships between interface and variant classes. In contrast to the meta model of the UML with associations ends modeled separately from the association itself [OMG 1999, p. 2-15], the CARTRONIC® UML meta model will be minimized to binary relationships only; therefore no relationship ending classes are defined and the two role names origin and destination are used. Each CartronicInterface has one Realisation or more, each Realisation belongs to exactly one CartronicInterface with public role name origin (envelope). A Realisation connects a CartronicInterface to exactly one CartronicVariant by a directed binary association with public role name destination at the association end CartronicVariant.

UML composition relations are used for modeling the hierarchic assignment of sub-components to a superior component. For example, in figure 2 the refined component Powertrain from figure 1 maps into the UML classes <<Interface>> Powertrain realized by the <<Variant>> PowertrainR1 as an enclosure for the five subsystems of figure 1 as part-of compositions <<Interface>> Powertrain_coordinator, <<Interface>> Engine, <<Interface>> Transmission, <<Interface>> Converter_Clutch and <<Interface>> Gearshift_panel. A composition relation always connects realizing classes with interface classes of refined components. Therefore, in the meta model each CartronicVariant may have no (zero) Composition or an arbitrary number (multiplicity expression 0..*), vice versa each Composition starts from exactly one CartronicVariant with role name origin (envelope). At the other end of a Composition exactly one CartronicInterface is connected with role name destination (subsystem) and vice versa each CartronicInterface is part of exactly one Composition. The multiplicity 0..1 is used because exactly one exception exists at the root of the composition tree. Using the OCL an invariant expression [OMG 1999, Warmer et al.1999] may be used to express this restriction by building a symmetric difference set of all Interface classes against composed ones and counting classes in the model simultaneously:

\[
\text{Composition}
\]

\[
\begin{align*}
&((\text{self}.\text{allInstances}.\text{destination}\to) \\
&\text{symmetricDifference(Interface.allInstances)})\to\text{size} = 1) \\
&\text{and (self}.\text{allInstances}.\text{destination}\to\text{size} + 1 = \\
&\text{Interface.allInstances}\to\text{size})
\end{align*}
\]

Composition relationships also implicitly include the mechanism of delegation of operations respectively messages. To specify exactly which operation is delegated to a subordinated class, the UML composition is itemized by a constraint, e.g. <<Variant>> PowertrainR1 delegates the operation rot_speed to <<Interface>> Engine. During the refinement there has to be exactly one component as the target component for forwarded orders called entry component in the function structure. To model this property the UML composition is specified by the role EntryOrder described by an enumeration attribute of the meta class Composition.

A structural description of CARTRONIC® communications result in operations defined in UML interface classes. The realization class <<Variant>> EngineR1 in figure 2 for example needs to know the actual air_pressure from the <<Interface>> Environment_data. The directed binary association expresses that the sender class EngineR1 knows (calls) the receiver class Environment_data. Analogous to composition relationships an OCL constraint expression specifies which operations will be called. In the meta model each CartronicVariant is aggregated to the meta class Communication with multiplicity 0..* specifying that as much as needed communications from a component are allowed including none (figure 6). Vice versa the CartronicVariant is a unique sender for the Communication and has again role name origin. 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..*).

4. Summary, Current and Future Work

The demand for innovative vehicle functions at reasonable costs leads to more and more complex vehicle functions as well as more and more complex interconnections between mechatronic components. Therefore a detailed specification and semantic foundation of all (logical) functionalities will be required. Semiformal models assist in structuring and managing complexity. They have to be translated into computer readable specifications to support automated and computer based development processes. A UML meta model is presented which is used for a mapping of automotive domain specific CARTRONIC® function structures into UML models. Model and meta model are implemented in the commercial tool Rose [Rational].

Current work includes automated consistency checks of domain specific rules which up to now could be checked only manually. The new UML meta model allows the formulation of OCL expressions describing
modeling restrictions. For example, one of this rules specifies the requirement that each component has to receive at least one operation (call) of type Order. This can be formalized by an OCL expression as an invariant for the meta class Interface (expressing that in the set of all contained operations one has to be a of type Order):

\[
\text{Interface}
\text{self.operation->exists(op | op.oclIsTypeOf(Order) = true)}
\]

Another rule defines that all orders reaching a refined component have to be forwarded from the envelope to a uniquely defined component managing all incoming orders. Based on the meta model this can be expressed as an invariant for the meta class Variant counting occurrences of composition relationships:

\[
\text{Variant}
\text{self.composition->size>0 implies self.compositionsEntryOrder->size=1}
\]

whereby compositionsEntryOrder is a derived OCL attribute [Warmer et al. 1999, p. 68, 71] containing the OCL collection Set(Composition) of composition relationships with attribute value EntryOrder.

A prototype of the checker is implemented in the programming language Java, reused from the object oriented modeling concept OMOS for ECUs [Hermsen et al. 2000].

Since UML behavior diagrams only show "examples" of dynamic behavior, current work also includes a systematic collection of information which should be specified by such diagrams. Further requirements for an information exchange between such UML domain models and function modeling/simulating tools like MATLAB/Simulink [MathWorks] are investigated.

From a single CARTRONIC® UML model several different design models may be created. In practical applications design model restrictions depend on system and development process characteristics like hardware components, operating systems, bus systems, real-time demands, tools etc. Therefore further essential working points are the development of design patterns and the development of an efficient management of variants.

5. References


