A Model-Based Approach for Automotive Software Development

Peter Braun, Martin Rappl
Institut für Informatik, TU München
Boltzmannstr. 3
85748 Garching b. München, Germany

Abstract: Integrated model-based specification techniques facilitate the definition of seamless development processes for electronic control units (ECUs) including support for domain specific issues such as management of signals, the integration of isolated logical functions or the deployment of functions to distributed networks of ECUs. A fundamental prerequisite of such approaches is the existence of an adequate modeling notation tailored to the specific needs of the application domain together with a precise definition of its syntax and its semantics. However, although these constituents are necessary, they are not sufficient for guaranteeing an efficient development process of ECU networks. In addition, methodical support which guides the application of the modeling notation must be an integral part of a model-based approach.

Therefore we propose the introduction of a so-called 'system model' which comprises all of these constituents. A major part of this system model constitutes the Automotive Modeling Language (AML), an architecture centric modeling language. The system model further comprises specifically tailored modeling notations derived from the Unified Modeling Language (UML) or the engineering tool ASCET-SD or general applicable structuring mechanisms like abstraction levels which support the definition of an AML relevant well-structured development process.

1 Introduction

Within the automotive industry model-based specification techniques are becoming more and more popular allowing the complete, the consistent, and the unambiguous specification of software and hardware parts of automotive specific networks of control units. In this context model-based approaches provide methodical support to manage the integration of logical functions and the deployment of functions to distributed networks of ECUs. In addition well founded models are the source for all kind of analysis, validation, and verification activities.

A prerequisite for the design of a model-based specification methodology is a precise knowledge of the architecture of the targeted system class. In our opinion a good architecture centric language reflects issues of automotive embedded systems as modeling concepts in terms of an automotive specific ontology. Thereupon rests the construction of a system model by precisely defining relations between model elements, their classification within abstraction levels and their embedding in a development process. During

modeling all information is stored in an integrated and consistent model. To cope with the complexity of this model a system of domain specific abstraction levels provides an appropriate structuring mechanism for the specification of networks of control units on different technical levels.

The presented work is related to the work the OMG [OMG99], the U2 Partner group [Gro01], and the pUML group [CEK+00] are carrying out. In contrast to their approach we believe, that a more rigorous mathematical theory is necessary for a realistic model-based development process. Nevertheless the representation of model elements is done by specifically adapted notations from the UML 1.3 [OMG99], the ASCET-SD modeling language [WWWb], and textual specification techniques which stem from the tool DOORS [WWWc].

The paper introduces the most important features of the Automotive Modeling Language (AML). Afterwards notions of the automotive specific ontology are sketched constituting major AML modeling concepts. In the final section we draw our conclusions and provide an outlook to future work.

The work presented in this paper represent results of the research project FORSOFT Automotive - Requirements Engineering for embedded systems [WWWa]. The partners of this project are the Technische Universität München, the tool providers Telelogic and ETAS, the car manufacturers BMW and Adam Opel, as well as the suppliers Robert Bosch and ZF Friedrichshafen.

2 AML Features in a Nutshell

To get a first impression of the AML, we summarize characteristic language features offered by the AML for modeling distributed embedded systems.

Requirements Classification. One essence of an architectural language is to provide a fixed vocabulary (or ontology) for talking about architectural issues. The AML introduces an ontology which is well suited for systems in the automotive domain. The requirements classification rests upon this ontology. The requirements classification itself is used as a tool to manage the transition from informal, few structured requirements to model aligned, structured requirements. With this classification in mind, the different architectural entities of a system can be identified.

Abstraction Levels define restrictive views on the system model to structure and filter information. Each abstraction level is based upon the more abstract levels. So in a more technical level access to the information contained in a more abstract level is permitted. All of these views show the system on a uniform technical level. At each abstraction level semantic properties are considered which are characteristic for the corresponding technical level. In the development process, the transition from one abstraction level to another abstraction level means to restrict the design space by finding a solution for a specified problem

Formation of Variants. This modeling concept allows to specialize architectural elements according to the context the element is used in. From a methodological point of view the relation between elements and their variants allows to manage complexity by abstracting specialized details to general needed model information. In contrast to the concept of inheritance in object orientation building variants from model elements just means to select specific subelements from the available set of subelements.

Architecture Specific Modeling Concepts. Known ADLs offer recurring modeling concepts for representing certain architectural aspects of a system. Apart from elderly Module Interconnection Languages (MILs), architectural concepts offered by ADLs can be comprised by following equation: "ADL=Components+Ports+Connectors+ Styles" [RS00, BSW94, BDD+93]. In the AML these concepts are applied to different kinds of architectural elements contained in the system model.

Semantic Domain. Each modeling concept of the AML can be represented by various notations. Notations may be textually, tabularly or graphically aligned. Especially the graphical notations, also known as box and line drawings yield profit by the mapping to the AML. Each construct of the notation can be expressed by a corresponding AML modeling concept and therefore inherits its semantics. Within the project Automotive we define mappings of parts of the AML to the UML, to the ASCET-SD [WWWb], and to textual representations. These mappings also define the transformation of models conforming to AML in DOORS, The UML Suite, and ASCET-SD.

3 Towards an Automotive Specific Architecture

Current research in the field of embedded automotive systems reveals the importance of automotive specific concepts in terms of an expert system architecture. As long as commonly accepted abstractions, understood in terms of reusable ontological entities, are not found, we consider domain specificity as desirable. In fact, collaborating in a domain specific manner might well be the only way to identify generally applicable abstractions.

The AML comprises notions which are well known in the automotive domain such as signals, functions, electronic control units, real-time operating systems, communication infrastructure, and processors for assembling the automotive embedded systems architecture. Each of these notions constitutes a fragment of the architectural model at a distinct level of abstraction.

In the sequel we list all notions of this ontology with respect to their classification within a system of AML relevant abstraction levels. We informally describe their semantics and their use as modeling concepts. In addition the AML offers general applicable modeling concepts such as hierarchical structuring, instantiation, formation of variants, formation of configurations and model composition for each presented ontology.

Signals. The abstraction level signals contains model information about the system with the lowest amount of technical details. The core modeling concepts at this stage

are signals and actions. Signals are elementary entities which can be exchanged between actors, sensors, and control units. Each signal can be measured or computed from a physical context. For the construction of architectural models, the model-based management of all signals occurring in a car is essential since their number goes far beyond ten thousand. Furthermore actions allow the modification of signal configurations with respect to a managed set of operations. Both concepts together provide in addition to an ordering mechanism enough modeling power to describe scenarios. Scenarios are ordered sequences of actions which are necessary to achieve a determined goal in a certain context.

Functions constitute basic building blocks at a high level of abstraction independently of later used implementation techniques or target languages. Particularly functions are considered which behave as abstractions of later used control units, actors, sensors, or the environment. Each function is provided with an interface stating the required and the offered signals. Those interfaces are used to model in- and out-ports of functions. For reuseability reasons functions prohibit the access to local signals by putting a scope on them. Therefore communication has to be handled explicitly via signal passing between ports.

One essential model content of architectural models is the explicit representation of signal dependencies between different control functions. Since functions respectively their instances are potentially distributable units that can be deployed to different control units, the consistent and the complete capture of model information in terms of signal dependencies between functions supports the analysis of functional networks and further the collaboration of distributed development teams.

Logical Architecture. The logical systems architecture is determined by the specification of logical partitions where fragments of the functional network are deployed to. These logical partitions characterize potential control units (in AML terminology ”functional clusters”), actors, sensors and the environment. At this stage the uniform treatment of the overall system is broken up to a set of independent subsystems working interactively together.

Comprehensive experience from the development of electronic control units reveals that a clear separation between the logical system architecture level and the technical system architecture level is very helpful when it comes to the partitioning of functions on ECUs. On the logical architecture level only a subset of partitioning criteria is applied in order to achieve a clear view of the functional structure - without identifying the set of functions which constitutes an ECU. However, finally the complete set of partitioning criteria (e.g. also those which consider geometric requirements) has to be applied.

Technical Architecture. On the one hand the technical architecture level is determined by the finalization of the responsibility of each control unit by the application of the full set of partitioning criterias given in terms of technical, economical, quality, and political constraints. On the other hand it is determined by the model-based connection of functions and logical clusters to models of the technical infrastructure (processors, real-time operating systems, and communication infrastructure).

Implementation. At the implementation level the realization of the model in hard- and software is regarded. Altogether this level takes up an exceptional role in the system of abstraction levels since no further information is added to the model. Code generation and the installation of hardware goes far beyond the realization of first prototypes which could be generated from the models gained above. Whereas there are many examples for a successful application of simulation and code generation facilities for testing and for rapid prototyping, code generation often fails to fulfill domain specific constraints. Therefore the generated code has to be manually optimized with respect to code size and execution time. These optimization steps are achieved at this level.

4 Conclusion

Motivated by the aim to meet the challenges of developing complex networks of heavily interacting ECUs, we have developed a system model comprising all necessary constituents for a model-based software development in the automotive domain. In this short paper we have presented major modeling concepts of the AML along with their classification within a system of abstraction levels.

Whereas the AML establishes the basis for an adequate modeling of software in the automotive domain - especially for ECU networks - the system of abstraction levels additionally provides means for structuring the development process according to different domain-specific categories. Our application of these concepts to a common realistic example, a window lifting control system, reveals the benefits of applying our approach.

Future work will cover the following two directions:

First of all, we plan to complete the automotive specific system model: On the one hand this comprises the formal and the complete definition of up to now informally and insufficient described parts. For example consistency dependencies between two adjacent abstraction levels have to be defined in an unambiguous way. On the other hand the concrete development process has to be defined based on the system of abstraction levels by providing rules and heuristics, how to use these levels.

Second, the tool supported transformation from non-executable models to executable, target-dependent code will be explored in order to achieve the long-term objective of a seamless, complete software development process for automotive applications.

5 Acknowledgments

We thank Michael von der Beeck, Jianjun Deng, Ulrich Freund, Bratislav Miric, Bernhard Schätz, and Christian Schröder for helpful discussions and for many comments on draft versions of this paper. We are much obliged to our colleagues of the project Automotive for many fruitful discussions and we thank Manfred Broy for directing this research. This

work has been partially funded by the Bayerische Forschungsstiftung (BayFor) within the Forschungsverbund für Software Engineering II (FORSOFT II).

References

[And99] Jesper Andersson. Die UML echtzeitfähig machen mit der formalen Sprache SDL. *OBJEKTspektrum*, (3), 1999.

[BDD+93] M. Broy, F. Dederich, C. Dendorfer, M. Fuchs, T. Gritzner, and R. Weber. The Design of Distributed Systems, An Introduction to FOCUS - Revised Version. Technical Report TUM-I9202, Technische Univerität München, 1993.

[BR00] Peter Braun and Martin Rappl. Model based Systems Engineering - A Unified Approach using UML. In *Systems Engineering - A Key to Competitive Advantage for All Industries Proceedings of the 2nd European Systems Engineering Conference (EuSEC 2000)*. Herbert Utz Verlag GmbH, 2000.

[BRS00] Peter Braun, Martin Rappl, and Jörg Schäuffele. Softwareentwicklungen für Steuergerätenetzwerke - Eine Methodik für die frühe Phase. In *VDI-Berichte*, number 1547, page 265 ff. VDI, 2000.

[BSW94] Garth Gullekson Bran Selic and Paul T. Ward. *Real-Time Object Oriented Modeling*. John Wiley, 1994.

[CEK+00] T. Clark, A.S. Evans, S. Kent, S. Brodsky, and S. Cook. A Feasibility Study in Rearchitecting UML as a Family of Languages using a Precise OO Meta-Modeling Approach. Technical report, pUML Group and IBM, 2000. Available at http://www.puml.org.

[Gro01] U2 Partner Group, editor. *Unified Modeling Language 2.0 Proposal*. U2 Partner Group (http://www.u2-partners.org), 2001. Initial Submission to OMG RFP ad/00-92-02.

[OMG99] OMG, editor. *OMG Unified Modeling Language Specification*. Object Management Group, http://www.omg.org, March 1999. Version 1.3 alpha R5.

[RS00] Bernhard Rumpe and Andy Schürr. UML + ROOM as a Standard ADL? In *Engineering of Complex Computer Systems, ICECCS'99 Proceedings*. IEEE Computer Society, 2000.

[vdBBRS01] Michael von der Beeck, Peter Braun, Martin Rappl, and Christian Schröder. Modellbasierte Softwareentwicklung für automobilspezifische Steuergerätenetzwerke. In *VDI-Berichte*, number 1646, page 293 ff., 2001.

[vdBBRS02] Michael von der Beeck, Peter Braun, Martin Rappl, and Christian Schröder. Automotive Software Development: A Model Based Approach. In *SAE World Congress 2002*. Society of Automotive Engineers, Inc., 2002.

[WWWa] Homepage Automotive (FORSOFT). http://www.forsoft.de/automotive/.

[WWWb] Homepage ETAS GmbH. http://www.etas.de/.

[WWWc] Homepage Telelogic AB. http://www.telelogic.de/.