
Requirements Management for Automotive Systems Development

Bernd Gebhard

BMW Group, Electric/ Electronics

Martin Rapp

Dept. of Computer Science, Munich University of Technology

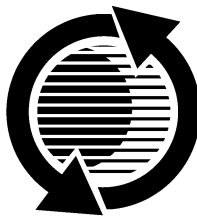
Reprinted From: In-Vehicle Software
(SP-1502)

The appearance of this ISSN code at the bottom of this page indicates SAE's consent that copies of the paper may be made for personal or internal use of specific clients. This consent is given on the condition, however, that the copier pay a \$7.00 per article copy fee through the Copyright Clearance Center, Inc. Operations Center, 222 Rosewood Drive, Danvers, MA 01923 for copying beyond that permitted by Sections 107 or 108 of the U.S. Copyright Law. This consent does not extend to other kinds of copying such as copying for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale.

SAE routinely stocks printed papers for a period of three years following date of publication. Direct your orders to SAE Customer Sales and Satisfaction Department.

Quantity reprint rates can be obtained from the Customer Sales and Satisfaction Department.

To request permission to reprint a technical paper or permission to use copyrighted SAE publications in other works, contact the SAE Publications Group.



GLOBAL MOBILITY DATABASE

All SAE papers, standards, and selected books are abstracted and indexed in the Global Mobility Database

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

ISSN 0148-7191

Copyright 2000 Society of Automotive Engineers, Inc.

Positions and opinions advanced in this paper are those of the author(s) and not necessarily those of SAE. The author is solely responsible for the content of the paper. A process is available by which discussions will be printed with the paper if it is published in SAE Transactions. For permission to publish this paper in full or in part, contact the SAE Publications Group.

Persons wishing to submit papers to be considered for presentation or publication through SAE should send the manuscript or a 300 word abstract of a proposed manuscript to: Secretary, Engineering Meetings Board, SAE.

Printed in USA

Requirements Management for Automotive Systems Development

Bernd Gebhard

BMW Group, Electric/ Electronics

Martin Rappl

Dept. of Computer Science, Munich University of Technology

Copyright © 2000 Society of Automotive Engineers, Inc.

ABSTRACT

Electric's and electronics' content in the high performance luxury segment's vehicles is continuously increasing. Dynamically controlled safety features, passenger comfort, information and entertainment and operational convenience are *the* field of permanent innovation. To guarantee an appropriate level of quality for this permanently innovated features requires robust design techniques (analysis, modeling, simulation, rapid prototyping and verification) and reliably controlled integration processes for components with significantly different life-cycles including information management and change control.

For these purposes BMW chose to utilize object-oriented analysis (OOA) methods and to define its automotive-specific subset of the Unified Modeling Language UML [OMG99]. The resulting object networks of OOA are then specified within an object-based specification (generation of executable models) process. Dedicated tools support a seamless development process.

Effective information management and change control are based on sound management of the projects' requirements. Thus, the information model of the developed products has to be embedded in the requirements model.

INTRODUCTION

Though innovative features are key potentials to competitive advantage, their merit will be limited, if quality, cost and time-to-market constraints are not met. Customers' wishes and requirements and the systems' constraints set up the entirety of functional and non-functional requirements, that have to be met as a prerequisite for successful products. To be able to meet these requirements one has to be sure to know them in their entirety, without ambiguities, incorrectness or contradictions, in an up-to date status. The requirements model evolves dur-

ing product development, i.e. requirements continuously change, are updated, deleted or new ones are created. To be able to control the „magic triangle“ of quality, cost and time constraints reliably, these changes have to be integrated into the current requirements model, their impact has to be analyzed and evaluated, trade-offs have to be carried out, decisions have to be made and to be documented.

This paper presents an approach to integrate the requirements model (functional and non-functional) with the results of OOA and object-based specification (functional, structural, behavioral view) in terms of process, methods and tools. The selected tools for the targeted process are Telelogic's ORCA for requirements capture and OOA, and ETAS's ASCET-SD (continuous systems) and Telelogic's SDT (discrete systems) to generate executable models. QSS's DOORS supports the requirements management process.

Sketching a prototypical run through a new development process in application to a running example presents a new methodology for the development of control units. The development of a control unit for interior light control within a car serves as an example for the demonstration of the methodology.

The paper is organized as follows. Section 2 describes the requirements engineering process for electronic control units, elicitation and refinement (OOA, specification) of the complete set of requirements. Essential steps of this process are illustrated by an example, the development of the interior lighting. Section 3 introduces the applicable basic concepts of requirements management and their deployment within the requirements engineering process. Finally, section 4 gives an outlook to the integration of the described processes, and concludes.

REQUIREMENTS ENGINEERING

The development of electronic control units in the field of automotive systems is split into a number of sequencing

phases, which are run through with respect of validation and verification activities to get control units, which are working correct and are fulfilling customer oriented functions. This section focuses the initial phase of the system development, the requirements engineering phase. Processes and notations for this phase, which are planned to get a standard at least for the BMW Group are presented. Aim of the initial development phase is the complete and unambiguous elicitation and description of all requirements, which belong to the whole network of control units.

In order to get a stable set of requirements, which serves as basis for further development activities (see Requirements Management section), the three activities elicitation, structuring and specification are proposed to control and refine the requirements. A first classification to precisely define, what requirements are separates functional and nonfunctional requirements [SBJ+98] both originated by user or system demands.

Functional requirements are directly connected with the behavior of the intended system and they can be represented by a functional description. In the field of automotive systems, the behavioral description has to be done with respect that electronic control units are reactive mostly hybrid (discrete and continuous) systems. The fact that functional requirements can be represented by a functional description, induces the logical view of a feature, which is associated with this functions. The set of functional requirements is therefore a set of features, where all elements are working in parallel and are interacting together.

Another advantage in treating functional requirements as features is the aspect of traceability. In the sequencing phases of the overall development process there exist a lot of models of the intended system, which are all represented in different languages with different levels of abstraction. If there is a missing mapping between requirements and specifications of the system, there is a great additional expense, which has to be invested to relocate the exact system functionality. Changes, as they occur every day, are very expensive to integrate. However features represent a logical encapsulation for requirements. These capsules survive the transition from the analysis model to the design model. Traceability of requirements is guaranteed.

Nonfunctional requirements are at least as important as functional requirements. Due to their nature, there exists no functional description for them. The only way to represent them is by natural text. To show the relevancy of nonfunctional requirements, later sections will sketch that nonfunctional requirements can effect the presence of functional requirements. In some sense nonfunctional requirements are constraints guaranteeing a proper work of the system. Therefore nonfunctional requirements define quality, reliability or portability aspects of the intended system. Quality attributes have to be elicited in the requirements phase and to be related to functional requirements.

Functional requirements, or better features, have to be represented in a notation, which can be pragmatically applied in the first development phase. These documents serve as a communication basis between different stakeholders. Therefore they have to be intuitively understood. Graphical or textual formalisms are well suited. For a comprehensive notation of features, different views on features have to be considered. These are structure, behavior and interaction communication.

For all of these views, the Unified Modeling Language (UML) [RAT99] offers appropriate means of notation. All diagrams of the UML have been standardized by the OMG [OMG99]. This is an important aspect for the integration of these notations in the development process of the BMW Group and it is also very important for tools to support the notations. All diagrams of the UML are abstract, containing a wealth of constructs to fit into nearly every application domain. For an optimal adaptation of the UML to the field of automotive systems, the UML was tailored to a subset, called automotive UML. For further information about this subset, please feel free and contact the authors of this article.

Figure 1 shows the three basic requirements engineering activities and correlates them with the different views on features. In the third column the used notations of automotive UML are listed.

Activity	View	Notation
Elicitation	Structure/ Inter-communication	Use Case/ MSCs
Structuring	Structure	Class Diagrams
Specification	Behavior	Statecharts

Figure 1.

Note: All diagrams, with respect to the standardization, are restricted to time discrete, state based systems. For modeling hybrid systems, the discrete means of notation have to be replaced by their hybrid variants (hyMSC[GKS99], hyCharts[ACH+95]).

In the sequel, the features for the interior light control are elaborated. Within the three requirements engineering activities elicitation, structuring and specification a model of functional requirements (features) is successively developed by using the notations of automotive UML. Because of the easy, well understood functionality, a first, rather simple model can be drawn as a statemachine with three states. All three states are directly dependent on the actual position of the interior light switch. In two cases, the interior light is switched on and off immediately, in the third case, the light is switched on and off in order to the position of the front doors. This basic functionality has been completed with features like get in assistance, get out assistance or emergency on. In addition to this, the features have been extended with timing and repetition protection to ensure a proper work of the

control unit. A behavioral model of this extended system includes about 30 states.

ACTIVITY 1: ELICITATION OF THE FEATURES OF THE INTERIOR LIGHT CONTROL – The notational medium in the first half of this activity is the Use Case diagram. Every feature, resp. use case in the terms of the UML, is added to existing, already elicited features. It doesn't matter if they are executed in mutual exclusion with other features or if any interaction conflicts with features executed in parallel exists. The enrichment of an existing use case diagram with new features is done under the assumption that a daemon is controlling the execution and is resolving any conflicts. This easy method of integrating new services implies that changes of functional requirements have to be applied top down in the initial development phase. The method of treating functional requirements as features eases the way of pushing these changes into models of the later phases. Figure 2 shows the elaborated features for the interior light control unit.

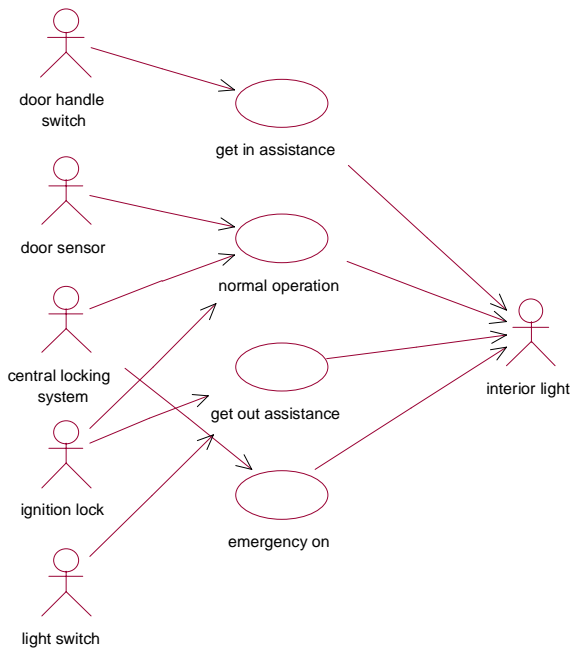


Figure 2. Features of the interior light controller

It is highly recommended to specify the elicited features textually, by using a structured text as suggested in [BO98] (figure 3). By keeping this structure during the textual specification, the potential actors, which are communicating with the system are identified. Each actor is capable to send out a set of signals, with an unique signature. The interface of the system is the result of the union of signatures, which are given by all I/O signals. The Use Case diagram is therefore equivalent with the context diagram, because of the precise interface description between system and environment.

USE CASE DESCRIPTION
Actors:
Initial Event:
Precondition:
Postcondition:
Description:
Exception:
Result:
Comment:

Figure 3. Textual Feature Specification

After the elicitation of features and the identification of actors, it is possible to specify exemplary scenarios of the use of a feature. The more formal MSC notation was used to model the communication interaction of the actors and the feature. Figure 4 shows an exemplary run of the feature get out assistance. Due to the fact that MSC specifications are not comprehensive, it is usual that there exist several scenarios for one feature.

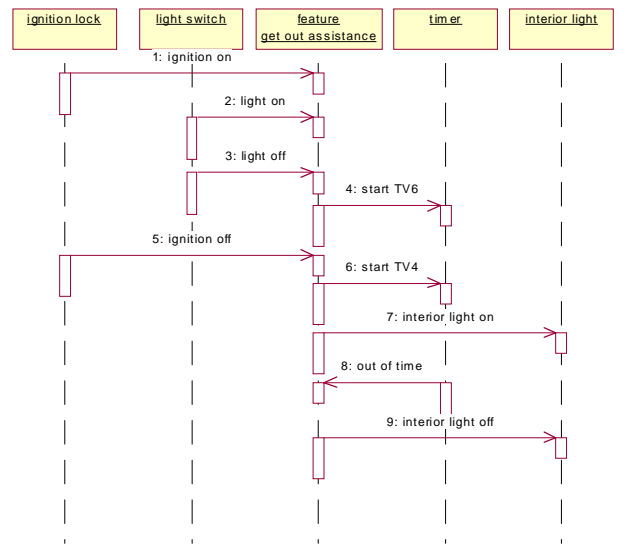


Figure 4. Exemplary use of the feature get out assistance

ACTIVITY 2: STRUCTURING – Aim of the second requirements engineering activity is to decompose the elicited features systematically to get small, manageable and reusable objects with low complexity. Further the relations among this objects should be worked out properly. The second activity carries out the transition from the Use Case diagram to the Class diagram. Aggregation and inheritance relations are used to achieve hierarchical decomposition of objects, resp. classes, or to exploit the reuse of objects within the same diagram. Whenever decomposition is done by using the aggregation relation, it is done under the premise that functional requirements are decomposed. System boundaries or other structural design information is not regarded.

ACTIVITY 3: SPECIFICATION – At least the behavior of the atomic classes have to be specified (with Statecharts). In the third activity no new knowledge is added to the requirements model. Behavioral diagrams are drawn by using all the information which was elaborated in the preceding phases. This information has only to be transformed from an informal, textual description into the formal statechart notation.

REQUIREMENTS MANAGEMENT

The information model of any project is represented by information of different maturity. Starting with vague wishes (market pull) or ideas (technology push) products have to be defined and developed. To efficiently handle changes, these different qualities of information have to be defined in their appropriate „levels“. Any information within the information model has to be allocated to the concerning level. The level reflect the increasing maturity of data and thus the development process.

REQUIREMENTS FLOW-DOWN – The only and most valuable information in projects' earliest phases are the customers' and users' needs and wishes or the developers' ideas. They state the **User Requirements** for a „product“. The User Requirements describe the WHATs of the desired product, the purpose of its existence. They are formulated independently of the later technical solution usually in the language of the customer or user.

User requirements have to be translated into the language of the developers. By this translation additional information is acquired, first decisions have to be made and the information matures. The requirements will be enriched with „technical“ information and thus represent a certain content of the subsequent solution. The **System Requirements** can be interpreted as a first abstract solution of the stated problem.

The most concrete information within the requirements model will be the **Architecture Design**, the description of the concrete solution. The Architecture Design already knows about the system's components and its partitioning in HW and SW. It can be seen as the HOW of the stated problem's solution. Figure 5 shows the requirements flow-down [QSS].

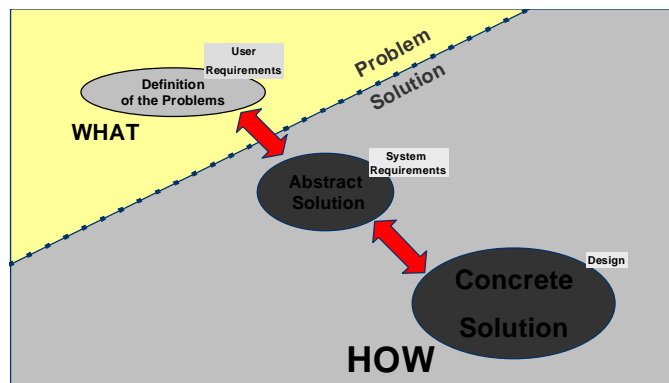


Figure 5. Requirements flow-down

TRACEABILITY – As information matures, the data model gets enriched and decisions have been made, decisions and data will be changed. It is a key to successful development to effectively control these changes. Therefore any information has to be allocated to its concerning level or layer. Information of different quality cannot be handled within the same level of abstraction. The elements within the layers (User Requirements, Systems Requirements, Architecture Design) have to be linked, i.e. their mutual implications have to be allocated and the layers have to be linked. I.e. the requirements model has to be enriched with context information like „Which part in the Architecture Design will cover which Requirement?“, „Which User Requirement is the reason for the statement of a System Requirement?“, „Which test verifies the completion of a System Requirement“, etc.. Thus the implications of changes of each element of the requirements model can be traced to all other elements. Figure 6 shows the principle of information tracing.

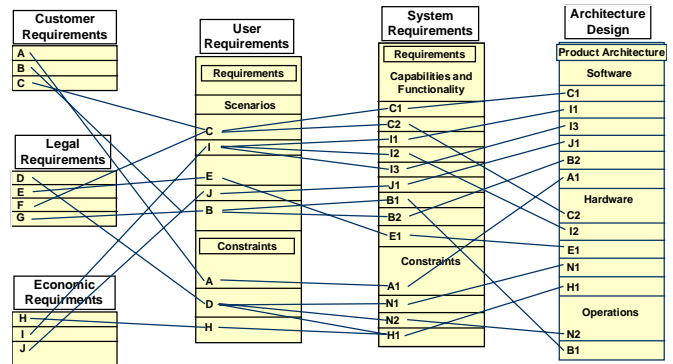


Figure 6. Traceability

DEVELOPMENT PROCESS – For the early phases of a complete development process the results of requirements engineering, i.e. the analysis model and specification model, and the requirements management have to be integrated into one information model methodologically. This means, the elements of both have to be set in a sensible context.

Due to the quantity of data, this can only be controlled by supporting tools. The goal to integrate the two information models into one requires the integration of the concerning engineering (analysis, specification) and project management (requirements management) tools. Within a project of the Bayerische Forschungsstiftung BMW, ETAS, QSS, Telelogic and the Munich University of Technology are defining the methods and developing the tool-chain for a seamless design process.

CONCLUSION

This paper sketches principles, how to put the development of electronic control units on a solid fundament of processes and notations in the early development phase. But if it is intended to use Requirements Engineering and Requirements Management techniques as a standard at the BMW Group, theory has to be supported by a

seamless tool chain. In respect to this, the tool vendors Telelogic, ETAS and QSS are working for an integrated tool suite, to support a seamless workflow management of the different models.. With the integration of three different development tools for Requirements Engineering, Requirements Management and System Design, we hope to be well prepared for future issues on the development methodology of embedded systems.

ACKNOWLEDGMENTS

This work has partially been funded by the Bayerische Forschungsstiftung (BayFor) within the ForSoft projects C3 and "Automotive".

REFERENCES

1. [BO98] Bernd Östreich: Objektorientierte Softwareentwicklung, Analyse und Design mit der Unified Modeling Language, Oldenbourg, München, 1998
2. [SBJ+98] Richard Stevens, Peter Brook, Ken Jackson, Stuart Arnold: Systems Engineering, Coping with complexity, Prentice Hall Europe, 1998
3. [RAT99] RATIONAL Software Corporation: The Unified Modeling Language V. 1.3., Semantics, 1999
4. [GKS99] Radu Grosu, Ingolf Krüger, Thomas Stauner: Hybrid Sequence Charts, Technische Universität München, TUM-I9914, 1999
5. [ACH+95] R. Alur, C. Courcoubetis, N. Hlabwachs, T.A. Henzinger, J. Sifakis: The algorithmic analysis of hybrid systems. Theoretical Computer Science, 138:3-34, 1995
6. [QSS] QSS: Keys to successful product development
7. [OMG99] Object Management Group: OMG Unified Modeling Language Specification (draft), Version 1.3 alphaR5, March 1999

CONTACT

The authors' e-mail addresses:

Bernd.GA.Gebhard@BMW.de
rappl@in.tum.de