A Method for Model Based Automotive Software Development

Martin Rohdin, Lars Ljungberg and Ulrik Eklund
Department of Electrical Engineering – Systems Integration
Volvo Car Corporation
SE - 405 31 Gothenburg, Sweden
vcc2.mrohdin@memo.volvo.se

Abstract

The distributed electrical system in the latest Volvo models, S80 and New V70, put new demands on software development methods. Increased demands for higher flexibility in the physical system requires higher flexibility in distributing functionality and more complex functionality requires higher quality of requirement specifications.

In the MESC method for model based automotive software development, function development is separated from the system development, which helps parallel work and improves reusability.

An object oriented function model provides means to analyse and describe functionality in a structured way and serves as an input to the development of a design model which can be validated against functional requirements.

In the system design phase, logical objects from the model are allocated on the physical system architecture.

1 Background

Developing the electrical system for the new Volvo platform, used in the S80 and the New V70 models, was the starting-point of a new technical era at Volvo Car Corporation. Earlier the electrical systems were based on one unique control unit per function and a very limited amount of shared information. The new system introduced is based on a set of electronic control units (ECUs) interconnected via a multiplexing CAN network where virtually all vehicle information can be shared between functions [1] [2].

The new system provides not only the means for new and more complex functions but also an increase in flexibility where functionality may be moved from one ECU to another. The number of ECUs may be changed even in late development phases. This leads to new demands on development methodology in both the function analysis and design phases.

The project organisation used when developing earlier systems was based on one organisational unit per sub-system (ECU plus peripheral components). Introducing the new technology and the new methodology has led to the need for an organisation based on the functions of the car rather than physical components.

2 Driving forces for a new development methodology

The main reasons for changing the development methodology from the previous one, adapted to “one function per ECU” technology to one adapted to the new possibilities introduced by the new technology are:

It is believed that the functionality of a premium car will play a greater role in a customer’s choice of brand. It will become more important that functionality is brand-unique which will increase the need for in-house development and knowledge of functionality.

Customers will associate brand with functionality, which demands that functionality is inherited from model to model regardless of system solution. This requires that functional requirement specifications are reused, i.e. the functionality is expressed in a non-system specific manner.

More complex functionality requires greater skills in communicating, expressing and handling functional requirements.

A large amount of sharing of resources and information in the system requires that the system and its functionality are handled from a complete system point of view where all functional dependencies are visible and all functional responsibilities are visualised.

Increased demands on shortened development lead times requires more parallel development activities and reusability of already developed functionality.

A more flexible system solution requires the functionality to be easily redistributed.

An increasing amount of functionality, even safety-critical, will be implemented in software, which requires higher quality of software requirement specifications.

3 MESC – Model based Electrical System Concept

To meet these drivers an automotive software development methodology called MESC (Model based Electrical System Concept) is being developed at Volvo Car Corporation

The parallel development challenge is met by separating the activity of developing the system solution from the activity of developing the functionality. These activities are performed in parallel by different teams. This separation means also that the functionality to a great extent can be expressed without the notion of a system solution leading to functional descriptions that are reusable for different system solutions and in different car projects.

In the MESC method the complete vehicle functionality is expressed using an object model based on a subset of the Unified Modelling Language UML [3]. The object model divides functional responsibility on associated logical objects which provides the means to overview the complete functionality and functional relations, to distribute the responsibilities on different system solutions and to divide and organise development responsibilities.

Using UML standard diagrams such as use case diagrams, class diagrams and sequence diagrams it becomes easier to express and communicate functional requirements on all organisational levels from product planning via system development to implementation and testing. Better understanding of requirements on all levels immediately results in higher software quality and both fewer and ”smaller” iterations.

MESC also aims to model functionality on a detailed design level using tools, which provides virtual analysis (simulation) capabilities. The design effort it self helps increasing the understanding of the functional requirements and the simulation helps visualising the functionality for both product planners and software suppliers.

This far MESC is aimed to cover the analysis and design phases but work is being made to extend MESC also to cover the implementation phase where a software platform including a set of platform components, e.g. a “virtual engine” for executing statecharts, has been developed. Some work has also been made to cover the system test phase in which one approach is to replace the system ECUs by PCs in the early development phases and execute the distributed applications on those, using a statechart simulator.

4 Functional modelling

4.1 Function areas

The functional requirements on the complete car is captured in one functional model which, for easier handling, is divided in a set of interacting Function Areas (FA), seen in Figure 1, each describing a subset of the complete functionality; e.g. Central Locking, Exterior Lighting and Alarm.

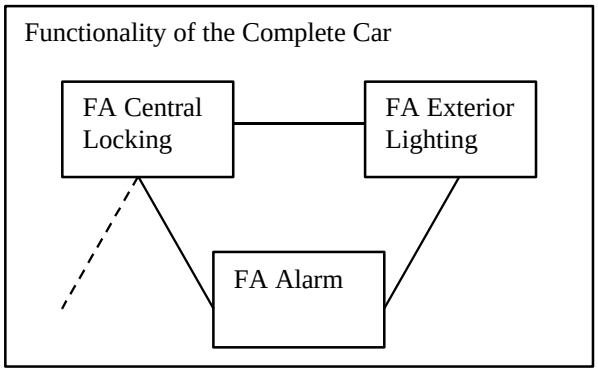

Figure 1: Function areas are separated and interact to fulfil the functionality of the complete car.

4.2 Defining use cases

Traditionally functional behaviour and requirements are described on a high abstraction level and divided into different sub-systems. This approach has been further developed in the MESC method in which we, for each function area, identify use cases and actors from the customer requirements. Use cases highlight what functionality we want from a function area on a high abstraction level. Actors, representing persons, physical environment etc, show which users are customers of the use cases. One use case diagram, which shows relations between actors and use cases, is drawn per function area, see Figure 2. This use case analysis has shown to be beneficial especially for new and unfamiliar functionality.

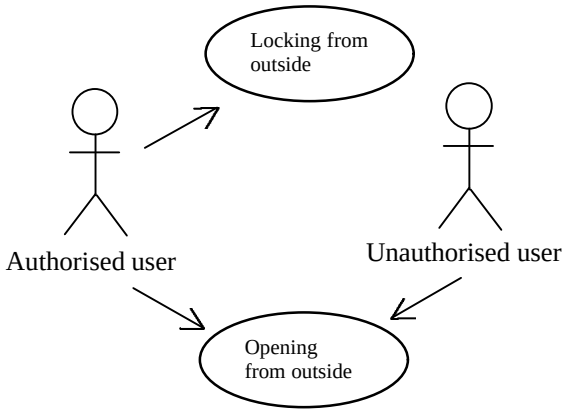

Figure 2: Use case diagram for FA Central Locking.

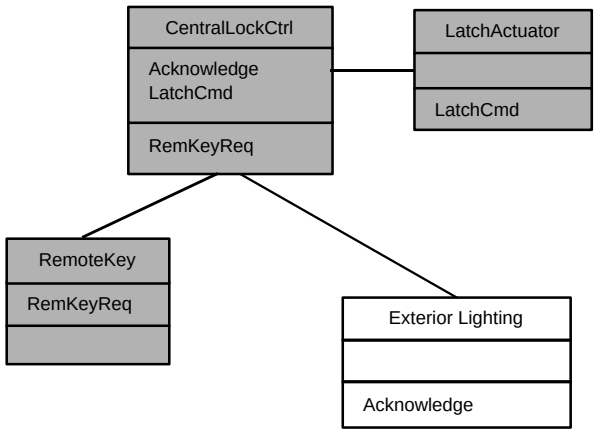

Figure 3: Part of class diagram for FA Central Locking. Grey classes are internal classes and transparent classes are external to FA Central Locking.

4.3 Logical objects and associations

Within the function areas the functional behaviour are divided on classes, that all have a defined responsibility. When identifying classes and dividing behaviour on them the aim should be that those classes (instanciated as logical objects) should be possible to distribute on different system architectures. This activity is based on incoming requirements, experience and system policies.

Classes have associations to other classes both internal and external to the function area. Such associations are shown by using the simple UML association line (bidirectional navigatable association) between classes. In this way a class diagram is drawn per function area. Figure 3 shows a part of the class diagram for FA Central Locking.

Traditionally hardware diagrams representing components and signals have been used at Volvo. Thus we try to interpret classes as "functional components" and associations between classes as message flows or signals between objects. To make this work we use class operations as input messages and attributes as output messages. Further we use only a subset of the UML syntax for class diagrams and we do not use aggregation and generalisation. This is because implementation in most cases is done in C (not C++) and that our software platform does not support these constructs.

This way we introduce "object influenced" semantics and we prepare the organisation for fully using object orientation and UML in the future.

4.4 Capturing behavioural requirements

The class diagram and classes provide the function area structure and responsibility respectively, and use cases provide the overall functionality. Details in the use cases are described using scenarios based on message flows between instances (objects) of classes. Such message sequence diagrams (MSCs) describe not only interactions between internal objects but also between internal and external objects participating in providing the function area functionality. In Figure 4 a MSC for the "Locking from Outside" use case is shown.

Functional timing requirements can be described in the MSCs, e.g. a message must be sent within some time limit or a message is sent periodically with a defined period. The timing aspects that are best described in the MSCs are however the complete reaction time from an actors input stimuli to the last message in the sequence is received.

Often some prerequisites must be fulfilled in a MSC for a message to be sent from one object to another. This is described in the MSC by using UML notes anchored to the message.

For some function areas we will apply formal verification techniques, e.g. for safety critical functions. I.e., if the MSCs are formally written with a complete set of prerequisites it will be possible to formally verify that the MSCs. are e.g. non-conflicting.

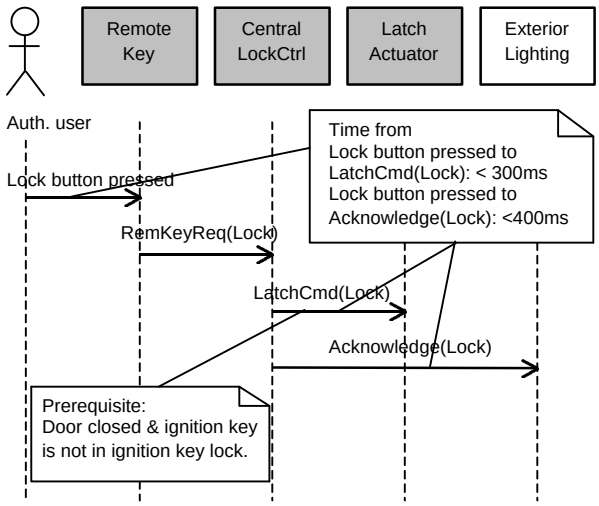

Figure 4: MSC for scenario: "Locking from outside".

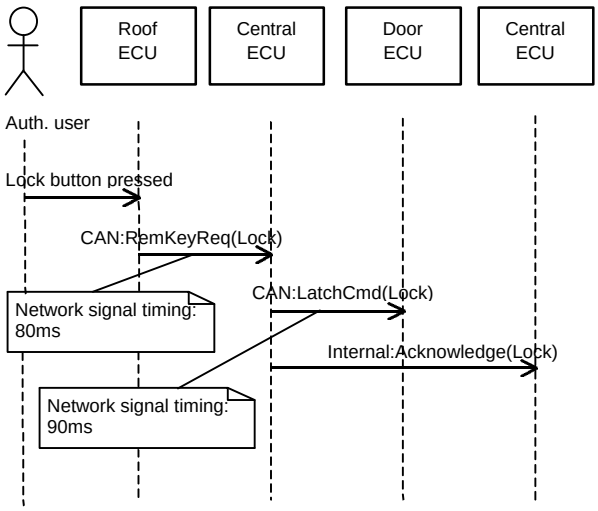

Figure 5: MSC for the system timing.

5 Design and validation

Some complex functions may need a deeper understanding than what the function modelling can provide. For such functions we make executable design models where the function model structure and behaviour forms the basis. Logical behaviour is modelled using statecharts in any tool providing simulation capabilities through animated GUIs. This helps us in validating the behaviour captured in the function model and product planners may actually see the results from their functional requirements before they are implemented on hardware. The executable design also helps the suppliers in understanding the design specifications. This approach drastically shortens feedback loops.

6 Function distribution – System design

In the system design phase the function model is allocated on the physical system architecture i.e. all instances of the function area classes are allocated on the sub-system ECUs which implies that some function area messages become network signals and others become ECU-internal signals.

The timing requirements from the functional MSCs are broken down to system timing diagrams describing requirements of evaluation time in ECUs, propagation times on networks etc, see Figure 5. The VOLCANO CAN protocol [2] assures that the network signal requirements are met by deadline monotonic analysis, but the timing allocation activity still requires a good knowledge of real time system design.

7 Future work

So far the MESC method is in use in the analysis and design phases but work is being made to further develop the implementation model (software platform) and the test phase. We expect tools for automatic code generation to mature further and rely on some kind of detailed design, why we already try to adopt design rules that would result in efficient and lean auto-generated code.

8 Acknowledgements

We would like to thank the consulting engineers at Combitech Systems AB, Gothenburg office, for valuable support and ideas on both object oriented techniques and methodology.

References

[1] Kent Melin, Volvo S80: Electrical system of the future, *Volvo Technology Report*, 1, 1998

[2] L. Casparsson, A. Rajnak, K. Tindell and P. Malmberg, Volcano a revolution in on-board communications, *Volvo Technology Report*, 1, 1998

[3] Grady Booch, Ivar Jacobson and James Rumbaugh, *The Unified Modeling Language User Guide*, Addison Wesley Publishing Company, 1998