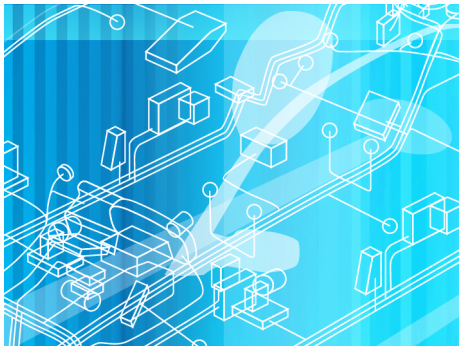
Fraunhofer Institut Software- und Systemtechnik

Rough Draft of VEIA Reference Process

Martin Große-Rhode, Simon Euringer,
Ekkart Kleinod, Stefan Mann



ISST-Bericht 80/07
January 2007

Editor: Fraunhofer-Gesellschaft e. V.
Institute for Software and Systems Engineering
Director: Prof. Dr. Jakob Rehof
Berlin branch of the institute: Mollstrasse 1
10178 Berlin
Germany
Dortmund branch of the institute: Joseph-von-Fraunhofer-Strasse 20
44227 Dortmund
Germany

ISSN 0943-1624

The VEIA Project

The project "Distributed Development and Integration of Automotive Product Lines" (German acronym, "VEIA"), is supported by the Federal Ministry of Education and Research within the framework of "Software Engineering 2006" research offensive under the support title "01ISF15A".

The project partners are:

- BMW Group
- Fraunhofer ISST
- PROSTEP IMP GmbH
- TU München

Authors

This document was written by

- Martin Große-Rhode (Fraunhofer ISST)
- Simon Euringer (BMW Group)
- Ekkart Kleinod (Fraunhofer ISST)
- Stefan Mann (Fraunhofer ISST)

Fraunhofer Institut Software- und Systemtechnik

Contents

1 Introduction

1.1 Purpose of the reference process in VEIA

The VEIA[1] project has set as its goal the formulation of a methodology for the distributed development and integration of automotive systems that is oriented toward the concrete demands of industrial development processes and has practical applications. This is to be achieved on the basis of the concept of product-line engineering.

In the first six-month project phase, the general structure of a reference process will be established whose primary function is to concretize the requirements of the four subprojects and to coordinate the project partners' activities.

The results of the project are foreseen as comprising methods, notations, and tools to be used to implement the activities of the reference process and produce the artifacts. Outside the framework of the project, the matching of the reference process and the requirements of the application domain is also a validation and evaluation of the project results.

1.2 The Reference Process

In order to achieve the project goals, a realistic and pragmatic approach in relation to the processes to be supported is necessary. Therefore, it is of particular importance to forge a link between the software and systems development processes known from the domain of research and product establishment processes for electrical/electronic (E/E) systems that exist in the application domain.

With the aid of a process model, artifacts that are developed successively in the engineering process are to be identified and their causal interdependencies

[1] The acronym VEIA stands for the joint project supported by the German Ministry of Education and Research "Distributed Development and Integration of Automotive Product Lines" being undertaken by the Fraunhofer ISST, Technische Universität München, BMW Group, and PROSTEP IMP, GmbH Website: http://veia.isst.fraunhofer.de/

described. It is they, in turn, that describe the engineering process, together with the relevant activities and roles—producing, evaluating, and maintaining the artifacts.

Processes that accompany and support will not be defined in the VEIA reference process.

The present document outlines the rough draft of the VEIA reference process, the following aspects of which are to be scrutinized:

- the artifacts in the automotive E/E engineering process and their causal relationships.
- Models to describe the artifacts, and
- the process phases based upon them with a focus on the general methodological and organizational framing conditions.

1.3 The Document Structure

Section 2 introduces the reference processes, the artifacts are justified, and their relationships to the other artifacts are presented.

In Section 3, the process is described as a chronological sequence of phases. The artifacts are dealt with and put into chronological order based on to their origin and/or use.

Section 4 also deals the artifact view, illustrating them using the example of Condition Based Service (CBS).

Section 5 briefly re-summarizes the contents of the document and gives a cursory overview of the follow-up activities in VEIA.

At the end of the document are chapters on “Notation” and “Acronyms”, whose goal is to aid in clarifying possible issues related to the graphics and technical terminology.

2 Artifacts in the Engineering Process

The application domain of automotive E/E systems requires a system-oriented approach in which hardware and software are developed simultaneously, and both processes are integrated into the long-term development process of the entire vehicle line. Possible interaction between these processes must be able to be observed at every point in time. It must be possible to identify and analyze solutions that are at variance with each other in order to be able to make decisions early enough in the process to allow follow-up.

In addition to the development activities, the required analysis activities must also be described explicitly. Among these are the various types of architecture evaluation that make clear which kinds of architecture descriptions are necessary, at which point they are needed, and the degree to which they can help optimize the project development.

The draft design of the VEIA reference process defines the requirements for the artifacts that are set as interim results in the process. They aid in the coordination among the partners and stakeholders as well as in quality assurance. The planned development and analysis activities will be described as will the information required for this and that which is actually provided. This connection is created by causal interdependences among the artifacts. The time matrix in which these activities will be implemented is to be defined by process phases (see Section 3.)

Section 2.1 gives an overview of the development artifacts discussed in this chapter. In Sections 2.2 to 2.6 the individual development artifacts and their mutual interactions are discussed in detail. Determining these artifacts is basic element of further project work in regard to support for product line development through methods, notations, tools and analytical procedures. In Section 2.7, the artifact relationships will be systematized in a summary, based upon which, in Section 2.8, activities for construction of the artifacts with their basic sequence are described. Modeling techniques will be summarized discussed in Section 2.9.

2.1 Overview of the Artifacts

Artifacts are interim results in the engineering process that are used in discussions and coordination: goals to be pursued with the systems under development; their concretization in the form of requirements; and the solutions based upon them

with the fundamental design decisions.

The development artifacts can be categorized as follows:

Products to describe the specific systems to be developed and their specific characteristics (features).

The area comprises the definition of the product line range in the form of product features and product names. Automobile product lines are usually defined in this area.

Functions to describe the functionality of the systems to be implemented.

By means of the artifacts, this area is roughly subdivided into the abstraction levels of requirements (description of the services), logical designs (networking of functions) and implementation designs (software architecture and description of the software components). In this regard, it corresponds to the structure of known software development processes.

Infrastructure to describe the technical resources to be used for the implementation of the functionality.

This area comprises the description of the technical architecture of the entire system and the specification of the individual controlling devices.

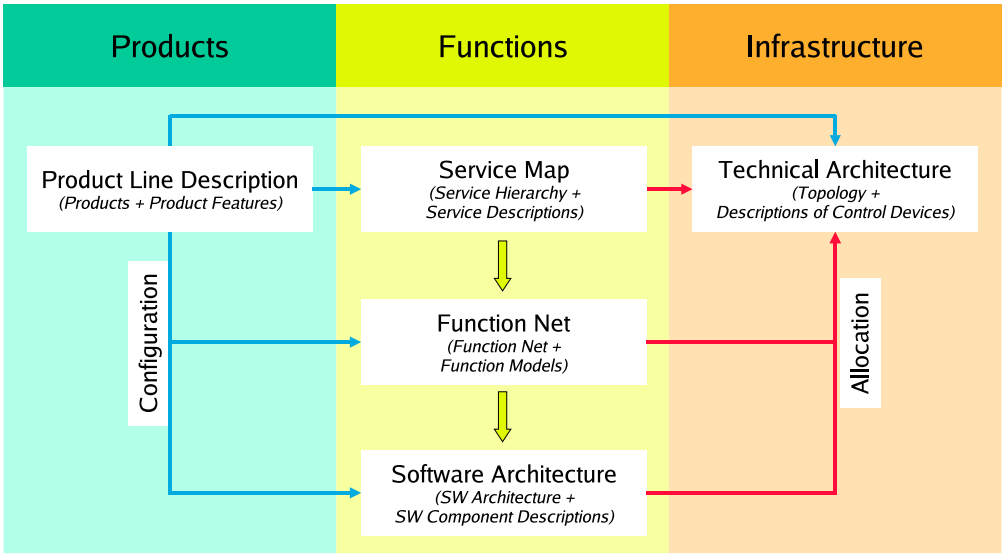
The concrete form of the artifacts is oriented toward methodic instructions to arrive step by step and precisely from "What should be developed?" to "How can this best be developed and implemented?".

For example, the artifacts in the area of "Functions" comprise groups of functional requirements (in the form of services) to establish the functionality from usage viewpoint (the black-box viewpoint of the system); design decisions in the form of function nets for the determination of the information to be gathered and processed; as well as the determination of software architectures as information-technical implementation of the functionality.

The following development artifacts are to be considered in the VEIA reference process for automotive E/E systems (see Figure 1):

- Product-line description (Section 2.2),
- Service Map (Section 2.3),
- Function Net (Section 2.4),
- Software Architecture (Section 2.5),
- Technical Architecture (Section 2.6),
- Configuration Models (Section 2.7.2) and

Figure 1 Artifacts and their relations in the VEIA Reference Process.



– Distribution Models (Section 2.7.3).

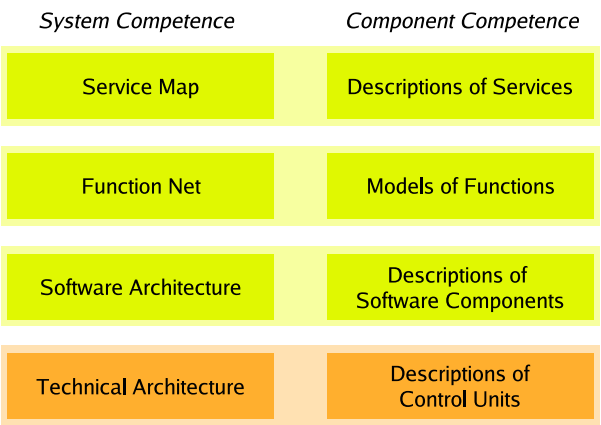
As shown in Figure 1 these artifacts are classified in the previously outlined categories and also set in relation with each other because they are either based upon each other or they have to be mutually consistent.

Unlike system development processes, in which the design of solutions—particularly software and infrastructures—always take place subsequent to the analysis of the requirements (both functional and non-functional), the VEIA reference process has several input points.

The first steps in the process can already include requirements analyses and infrastructure design along side the identification of system function requirements (services). This can be done because technical architectures (control unit topology, communication technologies, etc.) are often developed comprehensively for projects and product lines. The area "product" represents the expansion of the development process for individual systems to families of systems. This will define the range of the product line from the product line engineering viewpoint.

Orthogonal to the abstraction levels represented by the artifact groups, the distinction between the system and component level is significant and must be taken into consideration in the reference process (see Figure 2). Specific competences (in regard to responsibilities, technical abilities, etc.) are required at the compo-

Figure 2 Artifacts: System vs. Component Dimension.



nent level for the construction of the corresponding artifacts (individual service descriptions, functionality models, software component descriptions, and control unit descriptions). They also reflect a corresponding distribution of tasks in the engineering process.

A distinction is made between the system and component levels in all artifact groups, but the distributions are not congruent since no general 1:1 relationship can be established among services, functions, software components and control units.

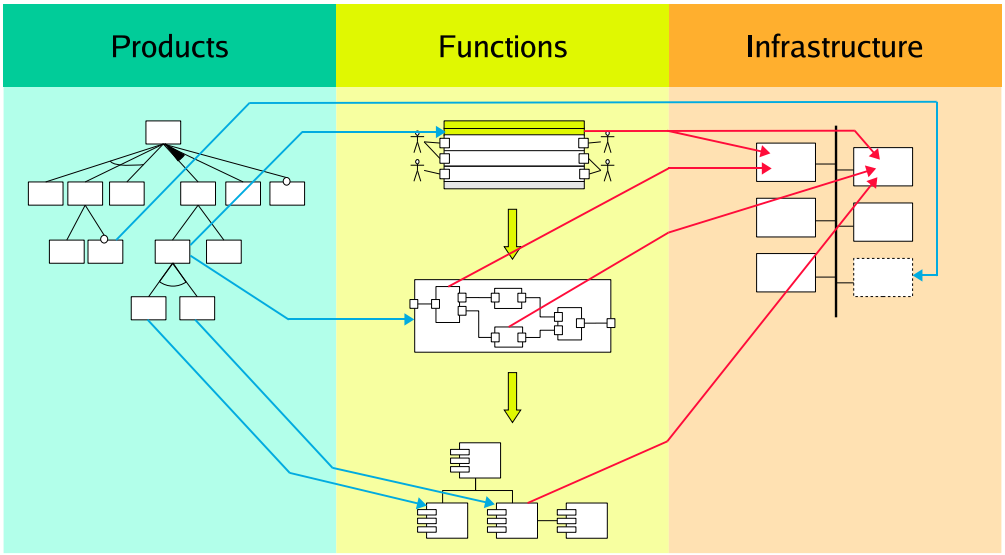
In Figure 3, the examples of the development artifacts to be discussed below are shown.

2.2 Product Line Description

The product line description covers the product features that are offered within the entire product line. It presents those features that are externally visible—from the view of clients or marketing viewpoint—in the system under development. The features can be related to both services and functions and to infrastructures (cf. Section 2.7.2).

The product line description in the "product" area is generally a vehicle product line description. Its variances and features determine the range of the vehicle product line. Going hand in hand with this, the variances for the other artifacts

Figure 3 Artifacts and their relationships in the VEIA reference process (schematic illustration).



(services, functions, software, and hardware) are given and are by and large taken over. However, every artifact has its own variance that does not come from the vehicle product line. For example, a separate software product line can arise in the software architecture and a separate control unit product line can emerge from the technical architecture, but these would be administered within the models.

The assignment of features to product types and groups or to equipment packages as well as the distinction between standard and special equipment options are based on the analysis of the relationships between the product line features.

The product line description serves as a basis for the configuration of the development artifacts for the areas "functions" and "infrastructure", in order to be able to identify and differentiate the models of individual products.

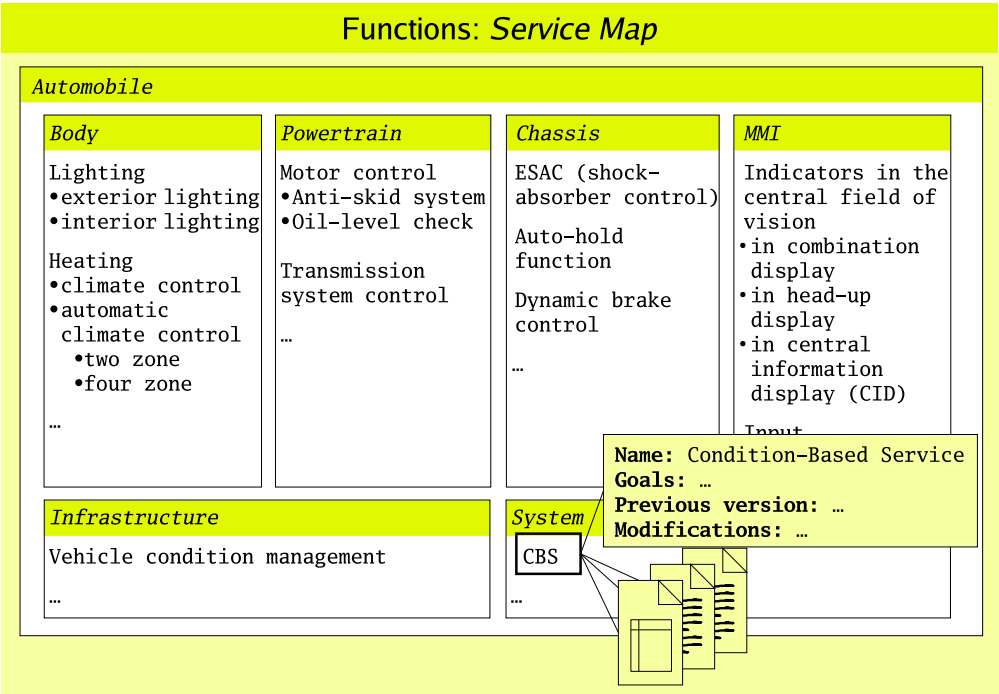
2.3 Service Map and Services

From a methodic viewpoint, it is necessary to analyze and consolidate the (textual) functional requirements, which have been formulated elsewhere. This is done through services, which describe the functionalities of the E/E systems to be developed that are visible from "outside". In this process, only the (idealized) core

functionality is described at first. The completion in regard to other aspects, such as error handling and diagnostic capability, is separated from this step (in the sense of the aspect orientation).

The formulation of the services takes place in a structure that is prescribed in terms of content and organizational structure as is regulated today, e.g., through body, interior, chassis, powertrain (KIFA), AUTOSAR areas, or the BMW functional organizational structure. The higher-level structure corresponds to the system level; on the "component level", the formulation of requirements takes place for the area-specific services through the stakeholder of this area. A visual representation of the service map schematic can be seen in Figure 4.

Figure 4 Service Map (Diagram).



At the outset, services are informal, described in suitable, unrestricted formats, for example with free texts or textual descriptions in DOORS, with the aid of Use Cases, through control loops or graphic sketches. The heterogeneity of the descriptions in the overall architecture is the reason for the term "service map". The heterogeneous items of information are affixed conceptually at the corresponding nodes in the hierarchically structured service map.

A service is a function of a vehicle that is offered to the user, e.g. `Condition`

Based Service (CBS), two or four zone automatic climate control, subservices such as motor control with anti-skid control and oil-level control, electronic shock absorber control (EDC), and various types of displays in the driver's central field of vision. The term "user" includes target groups who use the functionality or come into contact with it in one way or another including, e.g. driver, passengers, and service stations or automotive workshops.

Services are hierarchically structured, and various hierarchy relationships are to be distinguished, for example:

- Functional refinement of services into subservices, for example: lighting is divided into the subservices exterior lighting and interior lighting,
- Alternative services: for example for the service automatic climate control there are the alternatives of two-zones and four-zones.

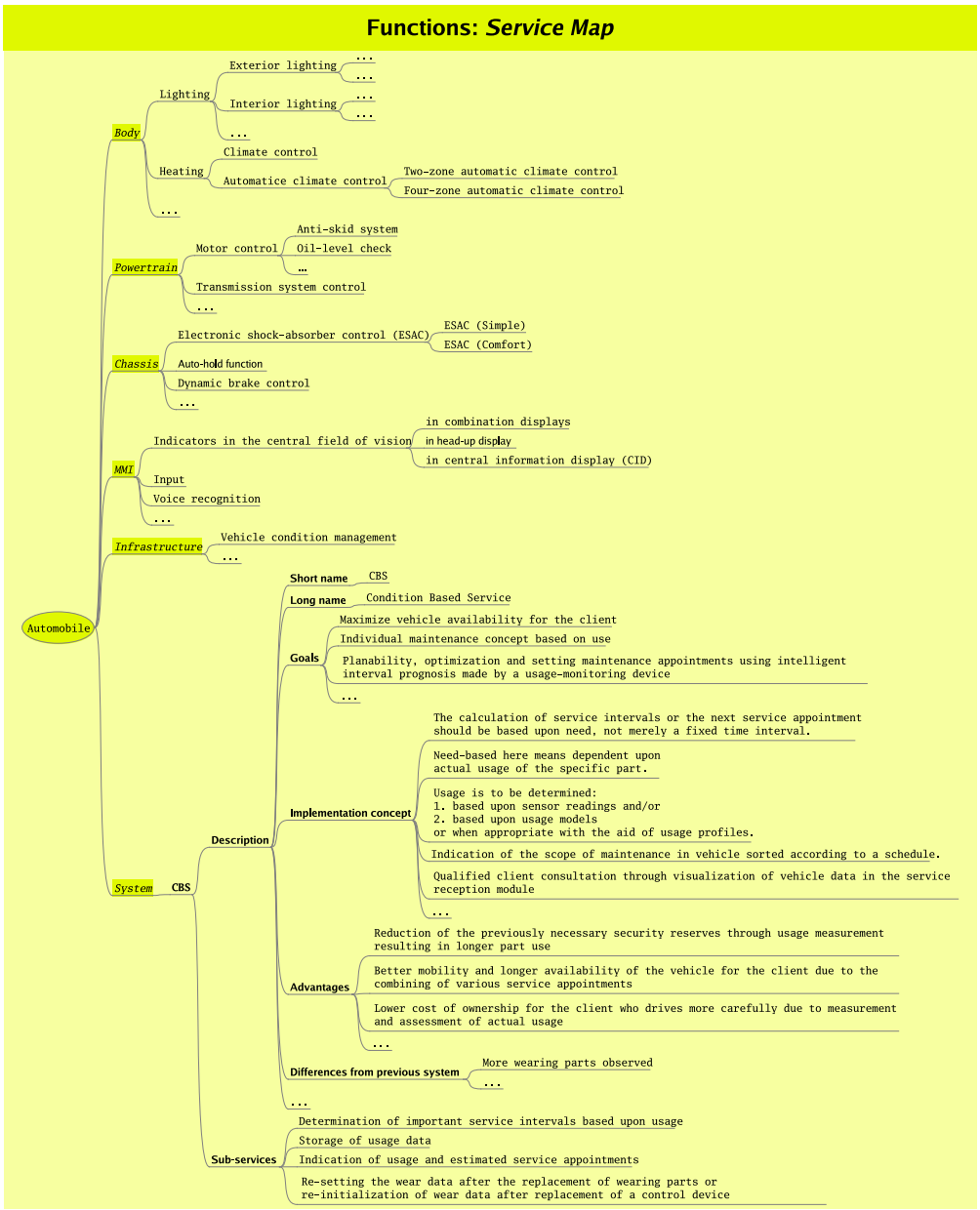
In the course of the project, a further task will be to identify these and other possible and meaningful hierarchy relationships in the service map and to support them methodically. In Figure 5, examples of hierarchically ordered services are represented as a mindmap but without an explicit distinction among the hierarchy relationships.

A service (i.e. every node in the service hierarchy) must be described exactly. Initially, a description schema that also includes informal parts and allows them to be managed is suitable. As the process progresses these aspects will be complemented through more formal descriptions (in accordance with the service model, cf. [GHHJ06], see also [BS01, SS03, KSTW04, Bro05, KMM05]) or also—in some cases—replaced. For example the usual short and long names of a service, the reason for the service (the goal the service is to achieve), as well as a short (textual) description. Further information in any form is possible.

The goal of the description of services in the service map is their formalized description as (partial) functions, i.e., that have not yet been specified precisely for all possible inputs. The advantage of this type of formal description is the possibility for better (tool supported) analysis of the assessment of the requirement; for example, whether and how services influence each other, whether there are contradictions in the service descriptions, and whether the full service range can be economically realized using the planned infrastructure.

The formal service description contains the specification of all inputs and outputs of the service as well as the (partial) description of how the output values (instructions for the actuating elements) can be determined from the (sensor-based) input values. The service description, as previously stated, can be refined by means of lower-level

Figure 5 Service Map (Illustration as a Mindmap).



services whose inputs and outputs, in contrast to those of the functions in the function net, are at the system level. In this sense, the service description is a formalized form of use case description (in accordance with UML Use Cases). The term "the services" and their formalization are described in detail in [GHHJ06].

The service map is used for the definition of products, that is, the formulation of functional requirements and not the specification of their realization: implementation details that extend beyond the general service description should not be registered here. The service map is "complete" when "all" services have been defined with a service description and all relevant information concerning the structure nodes and/or the services has been committed to writing. Conflicts should be resolved and there should be no more contradictions.

2.4 Function Net

For the realization of the services, functions are designed from which the services can be put together. With the design of the function net, common components of services are identified and relationships between the services ascertained.

Functions describe flows of information and behavior but no soft- or hardware dependent realizations. Unlike services, though, they also represent internal system functions and information flows. The function net at the system level describes the logical interfaces of the functions and the information flows between them. In this way it assists with the virtual integration, i.e., assuring the integration capability of the components to be implemented as well as system-wide optimization.

Individual areas and/or individual functions in the function net are represented by function models that are developed by the integration units with the relevant technical competence. Customization of the default interfaces specified by the function net and the functional interfaces that emerge from the function modeling occurs iteratively.

A function model describes a function in its structure and behavior, and every function model can, in turn, be constructed hierarchically from further functions as has already been described in MOSES [Fra03a, Fra05a, Kle06].

The interfaces for every function are determined with the aid of input and output ports in the form of signals (and/or operation signatures). With the aid of behavior modeling, the input/output relation between function input and function can be specified and (ideally) also simulated. The simulation can be carried out in phases: if "all" function models can be simulated, the function net can likewise be

simulated. Based upon this, signal flow analyses can be performed and possible sources of error detected.

2.5 Software Architecture

Through the software architecture, the implementation of the function net and its functions is described by software components. The system-wide software architecture is defined at the highest level; the system-wide software architecture is the model of all software components and their interrelationships. Architectural styles applied here are, for example, "master-slave" and "client-server" relationships between software components, among others.

The descriptions of the software components are derived from function models and maintained in a software component model. The structural and behavioral details necessary for implementation are added. In AUTOSAR, this corresponds to the filling of the Software Component Description Template. The grouping of the software component functions generally takes place in accordance with other criteria than in the functional decomposition in function models, which means that the hierarchical structuring of software components does not have to be congruent with the hierarchical structuring in the function net.

The relationships between software components (e.g., use relationships such as procedure calls, common use of global variables, etc.) are maintained with the aid of software architecture descriptions. In this way, the software architecture—similar to the function net for functions—contributes to the integration process, i.e., software component specifications or their implementations can thus be checked to ascertain whether they are compatible with the software architecture instructions.

The software architecture describes the structure of the application software. The basic software is part of a control unit and is, therefore, seen as part of the technical architecture. The goal is to design the software architecture of the applications software independent of the technical architecture, though in some cases this will not be possible due to specific hardware requirements.

A "software component" is the denotation for the smallest unit deployed in the software. Several software components can be installed on a single control unit, but the installation of an individual software component cannot be distributed among several control units. Multiple installations of a software component are, however, possible (on one or several control units simultaneously). The allocation

of a software component to several control units can thus be viewed as multiple instances of this component.

2.6 Technical Architecture

The technology decisions regarding the communication buses and other signal carriers—including the assignment of the control units and decisions about the structure of the communications networks and gateway technologies—are made with the help of technical architectural models (hardware topology models and control unit models).

In this process, a control unit model describes a control unit, including a virtual device; i.e., a control unit model encompasses a subsystem. The options offered by the AUTOSAR ECU Resource Template constitute the basic sources of the description, which means the technical resources of the control units, such as potential computing power, storage space or periphery accessibility, are described.

The control unit models must adequately fulfill the requirements of the technical system architecture, which means that the interfaces and inner structure of the control units to be developed are specified as far as necessary for their integration into the overall system design. As in the design of function nets and functions, the interface description of the control units occurs iteratively with the specification of the technical architecture (determination of bus interfaces, etc.). The modeling of the inner structure (for example, making commitments regarding processors, storage, etc.) not only aids in the estimation of service and costs, etc., but also in the software planning.

In the AUTOSAR context, the basic software forms part of the resources; it will be generated in accordance with the application requirements and the distribution of the application software components. In this respect, the description (designation and configuration) of the basic software can be assigned to the control unit description. The goal of AUTOSAR is to be able to generate the appropriate basic software.

The VEIA reference process begins to support the design of the hardware topology in the initial phase of the system and/or product line development, thus keeping it separate to a degree from the functional development. Architecture assessments that are independent of the actual use of the infrastructure—such as statistical through-put times of signals, etc.—can already be carried out early in the process. For assessments that are function dependent—such as execution times, e.g., with

HW/SW co-simulation or cost considerations—distribution models are used that are developed in various degrees of refinement.

The distribution of the functions on the hardware resources is represented by the corresponding distribution models (Section 2.7.3). Basic modeling concepts for technical architectures as well as distribution models have already been considered in MOSES [Fra03b, Fra04a, Fra04b, Fra05b, Fra05a, Kle06].

2.7 Artifact Relationships

As previously discussed, causal relationships exist among the development artifacts of the VEIA Reference Process. These can be subdivided as follows:

- Refinement and/or realization relationships,
- Configuration relationships, and
- Distribution relationships.

They are the result of the purpose that the individual artifacts in the development process have to fulfill. Through formalization and use of such relationships, the degree of inconsistencies among the individual artifacts can be checked, and it can be determined whether any aspects of the system that need to be considered have been "forgotten". The use of these relationships represents an important aid in ascertaining the (virtual) integration capability, and it serves as an early safeguard of test and integration activities in the right branch of the V model.

Refinement and realization relationships are vertical in nature since they establish a connection among artifacts at different levels of abstraction or different phases of development. In contrast, configuration and distribution relationships are horizontal in that they link artifacts from different system viewpoints on potentially the same level of abstraction or at the same phase of development.

In Figure 1 the relationships are listed: the refinement relationships between the models of the functional area; the configuration relationships between the product line description and the other models and, finally, the distribution relationships between the functional-artifacts and the technical architecture.

2.7.1 Refinement and Realization Relationships

Potential n:m relationships exist between the service map and function net as well as between the function net and software architecture. These relationships indicate which services are to be "realized" through which functions. Analogous to this, there are also relationships between functions and software components. The n:m relationship is due to the fact that the structuring and decomposition of services, functions, and software components must satisfy various criteria. Both these criteria and their relationships will be examined in further VEIA work packages and their use methodically investigated.

2.7.2 Configuration Relationships

Besides the establishment and description of variance in product line models, it is important to "resolve" variance that is inherently represented in the individual artifacts. On the one hand this will specify when and in which situations which variant is to be chosen, and (based upon it) on the other hand, it will make it possible to derive the individual products (and product models). The configuration (annotation) of varying artifacts enables this to be done.

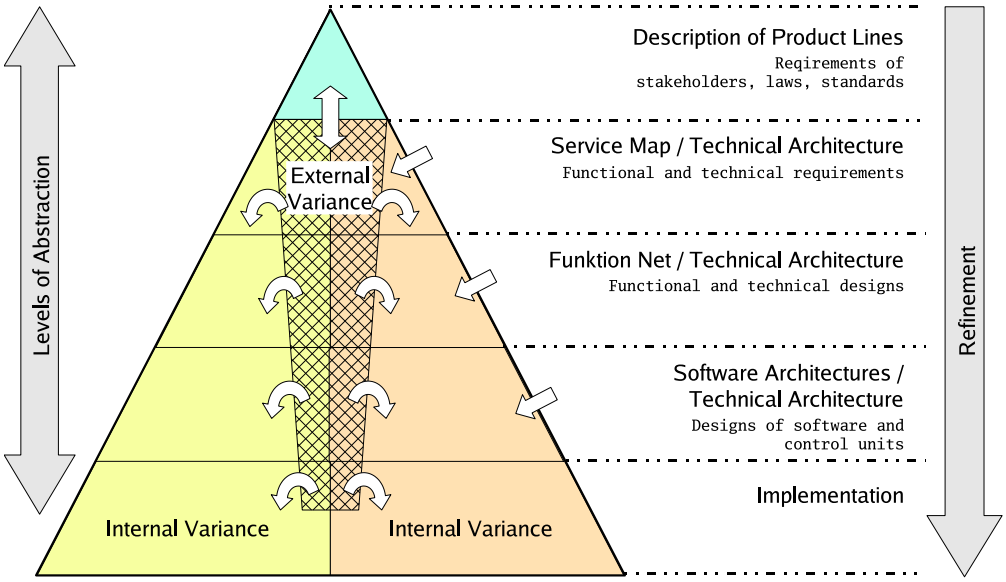
The external variance is identified in the product line description. The relationships between the product line description and the other development artifacts are established through corresponding configuration models. In this way, it must be clarified, for example, which functional variants in the function net are to "realize" or reflect specific feature variants in the product line description.

Configuration relationships do not have to be 1:1, but can also be of a more complex nature, meaning that feature combinations can be responsible for a functional variant. A first configuration model has already been indicated in MOSES, cf. [Fra04b, Fra05b, Fra05a, Kle06].

In Figure 6, the basic relationships between variance on the various levels of abstraction are represented. The illustration is borrowed from the work [PBL05, p. 71f.] and has been adapted here with reference to the development artifacts and levels of abstraction proposed in this document.

The illustration shows that as the level of abstraction decreases, the complexity of variance increases. This occurs because, on the one hand, the refinement and degree of detail increase (a lower level of abstraction "reflects" the variance introduced at the higher level of abstraction) and because, on the other

Figure 6 Variability Pyramid, based upon [PBL05].



hand, additional "internal" variance is added (incoming external arrows) since at each corresponding level of abstraction, additional factors have to be taken into consideration.

The variance that is made externally visible (external variance) that is defined in the product line description generally pertains to several levels of abstraction: guidelines for functionalities and their software implementation are established, as are guidelines for the technical architecture and its realization.

2.7.3 Distribution Relationships

Functions (and their implementation in software) must be able to be implemented on the infrastructure. The infrastructure elements are thus constraining resources: if, for example, no communication link is planned at the hardware level between control units, no communication can take place, even if the functions distributed among the control units that are not linked are able to communicate with each other.

Distribution relationships exist between the service map and technical architecture at the highest level of abstraction, as refinements for this purpose between both

the function net and technical architecture, as well as between software and technical architecture. The distribution of software on the technical architecture is also called software allocation.

Basic relationships between function net, software, and technical architecture, with consideration given to product line factors, have already been defined in MOSES, cf. [Fra04b, Fra05b, Fra05a, Kle06] under the term "Partitioning".

2.8 Activities for the Design, Evaluation and Maintenance of the Development Artifacts

The development artifacts described previously have to be constructed and maintained with methodological support and be evaluated as an aspect of quality assurance in the development process.

In Figure 7, the schematic from Figure 1 is expanded in terms of corresponding, basic activities for the construction of the artifacts using a methodic procedure and considering the causal dependences. The activities are represented by gray boxes with round corners.

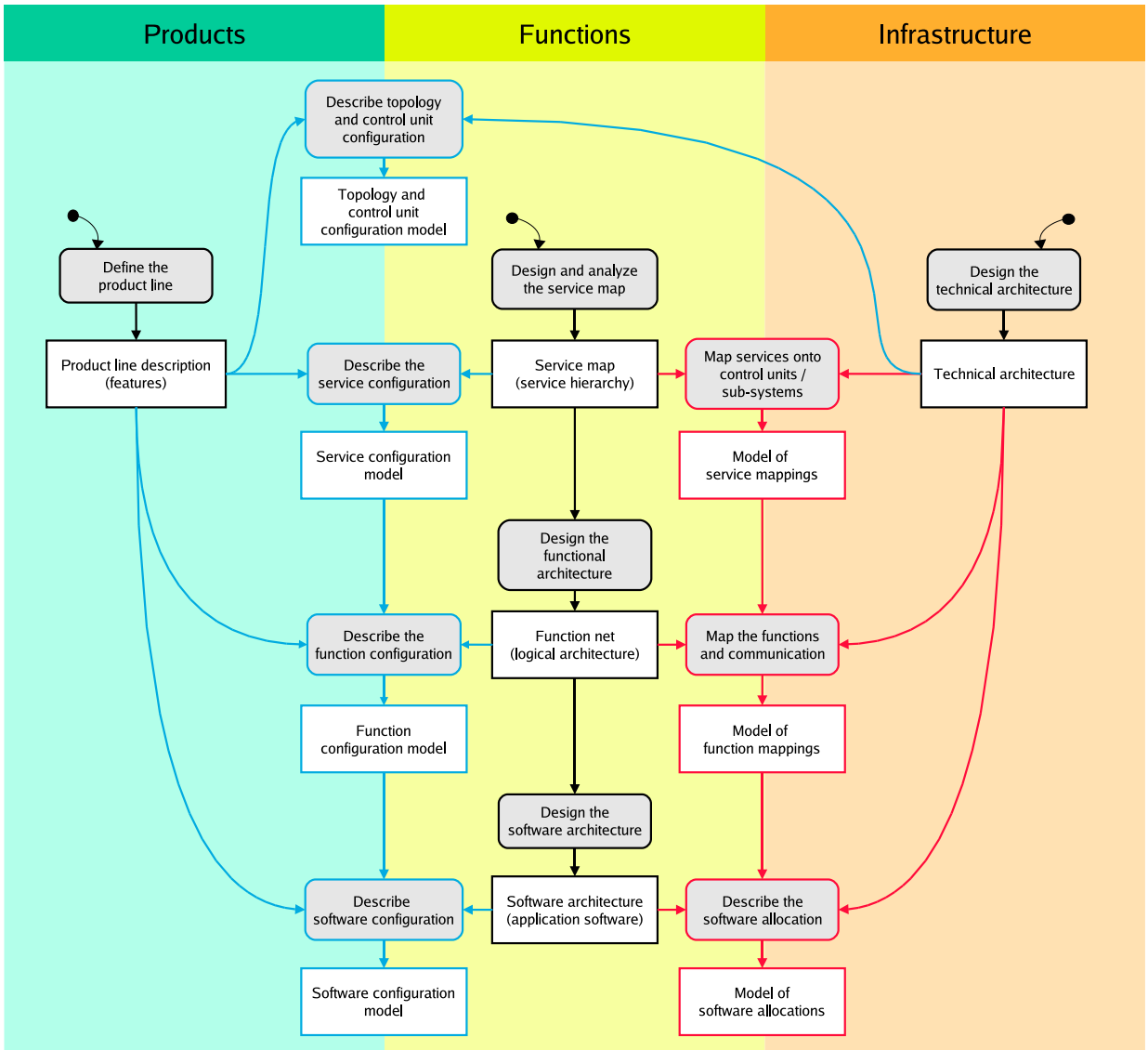
2.8.1 Preliminary Notes

Although a causal sequence of the artifacts is presented here, it must be kept in mind that this sequence is to be understood in such a way that a dependant artifact is only completely finished when its causal predecessors are finished. Whether interim versions are to be constructed in order to linearize the process or to design the process in a way that is manageable in general is the task of the description of the process phases in Section 3.

The description of the artifacts pertains only to the artifacts at the system level (cf. Figure 2), which is to facilitate the reading. It is clear, for example, that the construction of the service map progresses so that the descriptions of services are designed and are integrated into the map, and then further descriptions follow, which—in turn—have to be integrated, etc. Decisions made are revised and adapted in the process until the service map is fixed as an artifact. These iterations are not described further.

It is also clear that modifications of dependant artifacts can also affect their causal predecessors. This normal aspect of the development process is given no special

Figure 7 Artifacts: Causal relations and activities for the construction of the development artifacts.



attention. Rather it is viewed as a task of the model administration and the creations of various versions of the model, i.e., as change management. The reason for this and the previous simplification is that without them, retroactivity would have to be charted, which would reduce the readability and make the comprehension more difficult, without providing any additional information.

A further limitation is that reuse has not yet been discussed. It is clear that all available artifacts are reused if possible. The extent to which the artifacts are systematically kept (the domain approach of Model-based Systems Engineering (MOSES)) is either clarified in MOSES or has yet to be specified. This description would, however, be overloaded with this.

The artifacts are chosen so that each bears unique information, meaning the models are not redundant. Specific information occurs redundantly within processes, e.g. use cases of the service map contain the same information as service descriptions, but in varying degrees of formality. How this redundancy is dealt with is task of the (still to be written) model administration; that is, it is basically a tool question.

2.8.2 Basic Procedure

Step 1: Product Line Description, Service Map, Technical Architecture

The starting point of the modeling is the service map. It is the reference for all subsequent models. It is constructed based upon the requirements of all vehicle stakeholders, from which the use cases of a service are established. This makes it possible to describe the interfaces of the service as well as its behavior. The formalization of the use cases engenders the service descriptions. The services are integrated and synchronized. The service map encompasses the variance of the vehicle project at the service level.

During the formulation of the service map, the vehicle, the vehicle product line, and the production series are specified in detail. The vehicles to be built, the options to be offered, etc., are specified. This information is put in the service map, the product line description (the feature model). The service map and feature model are linked by the configuration model. The description of the production series and the feature model cannot be finished before the service map since the service map is the system's reference description.

The topology is also designed parallel to the service map. The control units to

be built and the quantity are determined, and, subsequently, how these control units are to be networked is decided. In this way, requirements related to flash times, bus load, communication speed, etc., can be taken into consideration. The services in the service map are distributed on the topology, determining the specific service that affects a specific control unit.

With the formulation of the topology and feature model, the relationship between these two models in the topology and control unit configuration models is also described.

Step 2: Function Net

Now the functionality architecture can be designed. To do this, the services are taken, analyzed, and modeled as functions. The functions are then refined, restructured, and summarized based upon logical factors. In this process, the functions are always synchronized with the entire function net so that a function net that can handle simulations is created. The variance is taken over from the services and adapted; in some circumstances, it is expanded or reduced since it is possible that variance in the function net differs from that in the service map.

Via the functional configuration, a relationship is established between the variance and the feature model.

The functions are also distributed onto the topology based upon the distribution of services, but the distribution of the functions must not violate the distribution of the corresponding services; it cannot expand them. That is, a function that implements a service can only be distributed onto a control unit onto which the service has also been distributed.

Step 3: Software Architecture

When the function net is complete, the software architecture can be finalized. To do this, the function models are implemented as software components. Logical linkages have to be considered, and signals and operations must be carried over into the software. The software architecture has to be designed so that the software does not become too big to be allocated yet is not too small to increase the allocation effort—and the concomitant flash effort—unnecessarily.

In principle, the software components describe only application software independently of the topology, but as indicated above, this is not always possible, for example, if information is divided up or the basic software requires a specific software design. In these cases the allocation is done in order to be able to design the software. The goal should be to reduce these cases to a minimum

in order to increase the reusability. In practice, software components can be described independently from the basic software through the use of adaptable code generators.

The variance of the function net is the starting point for the variance of the software components. At this point, too, modifications are possible, but if changes are made, the relationship between software components and features must be adhered to through the software configuration.

The allocation of the software components onto the topology is the last step. The allocation model specifies the software components that run on particular control units, a process in which the information catalog of the topology as well as the basic software are to be considered. The allocation model, too, must not exceed the limits of the function distribution model, there being restrictions analogous to those in the distribution of functions. Furthermore, software components can only be allocated to a single control unit.

With the construction of the software architecture and the allocation model, the topology, which up to this point had to be left open due to its dependence on the software architecture, can be completed. In regard to the hardware/software co-design, which is in the background of the considerations, changes in the software architecture also influence also the basic hardware.

Now all the models are complete, bringing the modeling process to an end. The models that have been constructed are used now to move along the implementation and integration, as well as to generate test cases and scenarios.

2.9 Modeling Techniques

Essential modeling techniques that can be employed to describe the artifacts of the reference process have already been considered in the project MOSES [Fra03a, Fra03b, Fra04a, Fra04b, Fra05b, Kle06]. In this project, distinctions were made among basic requirements, logical architectures, and hardware and software architectures, as well as the distribution of the logical architecture on the software and hardware architectures. The modeling of product lines and domain models were also supported in MOSES, and basis concepts for distributed modeling as well as for the multiple use and re-use of partial models were formulated.

The Department of Software & Systems Engineering at TU München has undertaken research on the concept of services as a formalization of use-case modeling, cf. [BS01, SS03, KSTW04, Bro05, KMM05]. Within the framework of VEIA, this

concept will be adapted to product-line development of automotive E/E systems, see [GHHJ06].

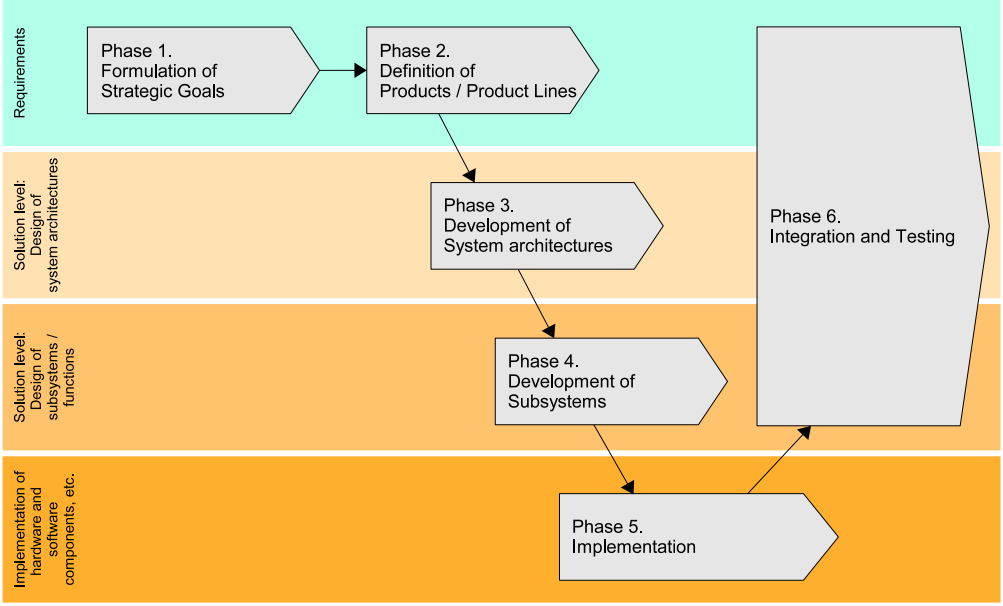
Among other things, AUTOSAR makes description techniques for software architectures and components as well as for technical architectures (hardware topologies and control unit descriptions) available. Product-line uses, though, are not supported explicitly, making the relevant adaptations are necessary.

Based on the requirements derived from the artifacts, the UML notations will be used appropriately: Behavior models such as state charts or activity diagrams can easily be imbedded in the relevant places. Class diagrams are suitable a means to describe the realization of software components. However, the are neither useful for service and functionality modeling nor for architecture descriptions.

3 Process Phases

The process phases discussed in this chapter provide a (rough) temporal sequence for the development artifacts of the VEIA reference process from Section 2 with consideration given to the causal dependencies and based upon methodological and organizational factors. An overview of the process phases considered is represented in Figure 8.

Figure 8 The Phases of the VEIA Reference Process.



The reference process is initially given for a vehicle project or product line; this is referred to as "the Project" in the following sections. The goal of a project of this type is the making of *products* (automobiles, automobile types) in a *product line* (automobile projects).

Product lines can generally be considered at various levels. In the automotive field, at least the following levels exists (Figure 9):

- Vehicle product lines to establish which motor vehicle types are put on a common basis,
- Control unit product lines (identical or similar parts)—control unit suppliers in particular work at this level—and

– Software component product lines.

Figure 9 Product Lines at Various Levels and Their Relationships.

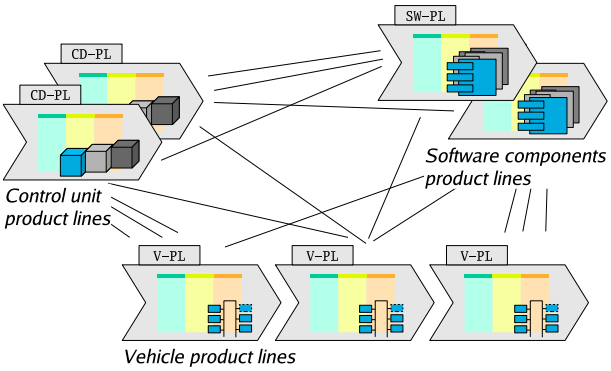


Figure 9 shows the relationships between these product lines. Software component product lines can be constructed in a way that they can be deployed in different vehicle product lines. The system function *CBS* (see Section 4) seeks to develop a generic solution, if possible as a software component product line, which can be used in several vehicle product lines. Similarly, control unit suppliers also have an interest that their control unit product line can be used for several vehicle product lines (including different OEMs).[2] The OEM, on the other hand, is interested in constructing identical and similar parts so that control unit product lines and software component product lines can be deployed to the greatest degree possible in the vehicles.

In the following, the individual process phases for the development of product lines in the automotive E/E context are explained; each phase is outlined according to the following scheme:

– abstract,
– goal,
– non-goal,
– artifacts and concepts, and
– examples.

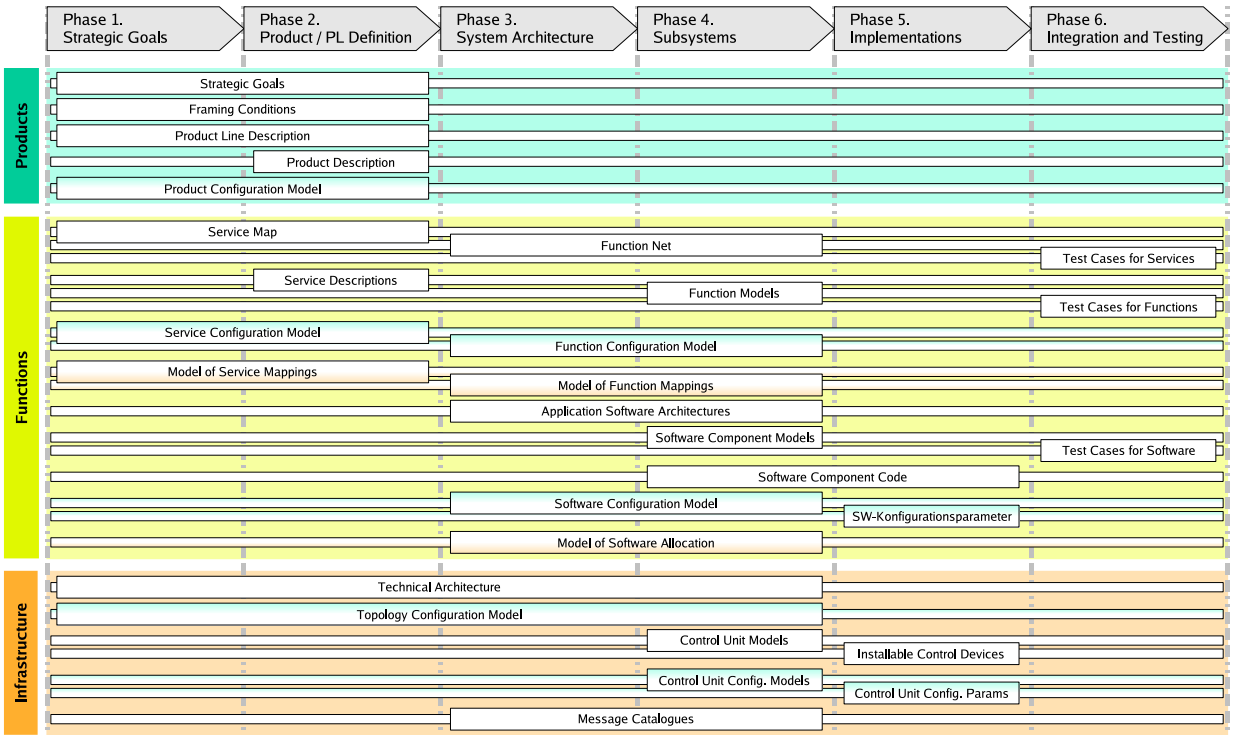
Individual artifacts and process phases acquire greater significance than others

[2] In VEIA, suppliers' use of product lines is not investigated.

depending on the level on which a product line is to be developed. Nonetheless, aspects of the product lines to be supported must be noted and considered on the other levels.

Figure 10 assigns the development artifacts to the respective process phases described in the following section. The classification indicates in which phase an artifact is primarily worked on; then the artifact is represented more broadly. The thin rows indicate how long the artifacts are valid. In VEIA, all artifacts maintain their validity until the end of the development and can then be updated. This updating, which occurs as a reaction to modifications in other artifacts in all phases, is not represented in the illustration as it was in the artifact description for the sake of clarity. It is, however, present and must be brought into play by means of corresponding change management processes.

Figure 10 The VEIA Reference Process: Assignment of the Artifacts to the Process Phases.



If the assignment of an artifact is distributed over several phases, the artifact goes through several "phases of development". For example, the technical architecture in phase 1 serves as a structured collection of requirements for the technical architecture to be implemented in the project that is specified in phases 3 and 4.

3.1 Phase 1: Formulating Strategic Goals

The first phase is linked to the project inception. Specifications are gathered that the products to be developed in the new project are to meet; these stem from the stakeholders. In this process, old specifications that are complemented by new ideas can be re-used.

This phase represents what might be called brainstorming on the project level. The results of this phase are analyzed in phase 2, checked for their viability and thus consolidated, and then a commitment is made to the standards.

Goal

The goal of this phase is to provide a general organizational and technical outline for the project. The functionality that is supposed to be built into the products must be planned just like organizational and financial framing conditions. In particular, the product variants that are supposed to be built as an automobile product line are sketched out.

Non-goal

It is not a goal to establish detailed technical standards or to formulate precise and "good" requirements (in accordance with requirement modeling concepts).

Artifacts and Activities

Artifacts and activities for phase 1 are listed in Table 1.

Table 1 Overview of artifacts and activities in Phase 1 "Formulating Strategic Goals"

Feature	Contents
Input	– Goals of the enterprise – Legal standards – ...
Processing	Description of the basic qualities of the products (product variants, functionality, technical architecture) and the corresponding framing conditions (e.g., costs, facilities, personnel)
	↪ ...

Feature	Contents
Output	– Strategic goals and framing conditions for the vehicle product line – Product line description (feature descriptions) – Service map as reuse standard and / or stakeholders' "wish list" – Initial technical architecture as reuse standard and / or stakeholders' "wishlist"
Implementation	Strategy, marketing
Reuse	– Functionality known and already implemented in predecessor products, – Technical architectures known and already used in predecessor products, – Experience values from predecessor projects (regarding feasibility, costs, etc.)
Variance	The project is defined so far that the (product) variations being developed can be outlined: for example through considerations of variance in the equipment (product line model).

Examples

Using currently available sensor technology, it is possible to offer service that is based upon actual wearout due to use, e.g. notification of the driver when the next oil change or brake replacement is due.
This should make it possible to optimize the service life of the vehicle between two service appointments based upon the driving profile.

Toward achieving a "Best in Class" rating, the integration of standard consumer electronics (iPod, MP3 player, lap-top, mobile phone) should supported.

3.2 Phase 2: Definition of Product Line and Products

The main purpose of the second phase is a detailed definition of the products to be developed in the project and/or the vehicle product line (Definition of the scope of the product line, definition of all product variants): The initial requirements from Phase 1 are to be consolidated and the service map is to be refined to the degree that the functionality to be implemented is described precisely as a black box in terms of its interface and behavior. With the technical architecture, the basic bus structure can already be considered and the flash and communication requirements adopted.

Goal

The goal of this phase is the completion of the initial service map from Phase 1 (Have all services been noted?), the elimination of contradictions, and the resolution of ambiguities. The services (black box functions) are to be described concretely enough so that the design of the function net can begin (in phase 3). In addition, the interface of the services and their essential behavior (ideal behavior) are to be specified. The interface of a service comprises the inputs (sensors, setpoint devices) and outputs (actuating devices, displays, additional outputs) at the system limits. Complete consideration of error cases, for example, is still not necessary in this phase.

In the area of the infrastructure description, the goal is to outline a topology of control units with which the functionality can be realized. As far as it is reasonable and possible, an initial distribution of the services on the technical architecture can be done in order to make first estimates of whether (on the basis of experience values) the functionality foreseen will, in principle, be possible with the infrastructure envisaged.

The product variants to be developed are to be determined and noted in the product line description. That is, the product features are to be defined (e.g. equipment features, or customer-value features) as is their assignment to the products of the product line. Here, for example, the scaling of the vehicle power supply for the product variants is determined, which leads to an early control unit product line model that is represented in the technical architecture.

Non-Goal

A non-goal is the modeling of the internal behavior of the services (internal functionality decomposition). The complete internal behavior is not a priority, only new or important behavior. Also, the information flow is noted, but details are not gone into.

End

This phase is finished when:

- all products of the product line have been determined (designated and their features described);
- all services have been designated;
- the black-box behavior (ideal behavior) of all services has been described;
- the framing conditions for the individual services have been determined;

– the framing conditions for the technical architecture have been defined.

Artifacts & Activities

The artifacts and activities for this phase are listed in Table 2.

Table 2 Survey of artifacts and activities in phase 2 "Definition of Product Line and Products"

Feature	Contents
Input	All results from Phase 1.
Processing	– Analysis of the service map (from Phase 1) and the framing conditions with respect to feasibility, efficiency, conflicts, etc. – Revision, consolidation, and concretization of the initial requirements (service map and framing conditions). – Modeling of the services in detail so that their black-box behavior is evident. The behavior does not have to be absolutely complete (in regard to error handling, etc.). – Further determination of the framing conditions in regard to infrastructure.
Output	– Vehicle product line model (Equipment features for the product line and its product variants) – complete service map with service descriptions including interfaces and black box behavior – refined, consolidated technical architecture (with respect to the service map) including first variance (control unit product line model) – refined and consolidated framing conditions
Implementation	"Delegates" of the competence centers
Reuse	Feature models, service map, service descriptions, technical architectures from predecessor projects and/or from enterprise standards
Variance	Variances to be considered and supported in the project ("external variances") are to be analyzed in detail with respect to feature models, services, and technical architecture analyses.

Examples

The CBS service assists in measuring actual wearout of wearing parts. Wearout or use of the following is to be monitored: oil, brake linings, brake fluid, spark plugs (in gasoline engines), particle filters (in diesel engines), ...If sensors are available, they are to be used. Otherwise wearout is to be estimated based on the driving profile. On the basis of use analysis, optimized service appointments are to be indicated by the multi-purpose instruments. If the vehicle has comfort equipment, a heads-up display is also to be used.

Infrastructure scaling: The infrastructure for standard variants is to

```
be designed without MOST. A MOST bus is to be installed with the first
upgraded audio equipment.
```

3.3 Phase 3: Development of the System Architectures

In this phase, the products and the product line are to be reviewed at the overall system level divided into suitable subsystems or functions, infrastructure, and software. The goal is to indicate the framework for the competence centers, on the basis of which the "virtual" integration of the functions, control units, and software components is possible.

Goal

The goal of Phase 3 is to describe the functions of the products or the product line so that their behavior and interfaces are determined to the degree that a function that is implemented is run capable and is compatible with the other functions.

Infrastructure considerations seek to enable a control unit topology as a part of the technical architecture on which the functions have to or are able to run. A corresponding distribution of the functions onto the control units is undertaken.

The units to be implemented (subsystems) are determined and distributed among the competence centers that carry out the respective further developments in Phase 4.

Non-Goal

A non-goal is detailed modeling of the functions and to make them available as a white box. This is to be done in Phase 4 under the responsibility of the competence centers (cf. discussion system vs. component levels, Section 2).

End

The phase is ended, when:

– all functions are modeled so far that their compatibility is secured, the exchange of data is described and the behavior is described, i.e., their interface is determined; behavior requirements are determined from the system viewpoint, and the flow of information is defined on the function net level level; (Have all the

necessary signals been provided? Are there signals that have been provided that will not be used?)
- the technical architecture is determined with the interfaces and buses of the control units;
- the technical signals ("information catalogue") has been determined;
- the framing conditions for the individual functions has been determined (e.g., distribution, configuration).

Artifacts & Activities

Artifacts and activities for this phase are listed in Table 3.

Table 3 Overview of artifacts and activities in Phase 3 "Development of the System Architectures"

Feature	Contents
Input	All results from Phase 2.
Processing	– During this phase, the architecture of the control unit topology and the function net is designed. – This involves "realization" of the services of the service map through functions: services are divided into functions, and dependent services are combined with functions and integrated into a function net. – The hardware topology is simultaneously refined, and the resulting function net is distributed onto the technical architecture. – This phase is finished when the functions have been described adequately so that they can be distributed onto the topology and given to the appropriate competence center for the detailed internal modeling. – Requirements of the technical architecture are formulated with respect to possible effects on the software architecture, e.g., whether AUTOSAR is to be employed or which basic software is to be used, etc.
Output	– Function net with function models (determination of the interface) – Technical architecture: Hardware topology with control unit standards – Distribution of the function net onto the technical architecture – Information catalogs
Implementation	"Delegates" of the competence centers, integration units
Reuse	– Specifications of functional framework architectures and function components from predecessor projects or from cross-project domain models – Framework architectures for the control unit topology and hardware components from predecessor projects or from cross-project domain models
Variance	Clarification of variances at the system architecture level, like the identification of variance (from Phase 2) is to be dealt with: "Implementation of variance", binding time, product line approaches for subsystems (e.g., software product lines for individual system functions that can be used across projects, etc.

Examples

```
The conception of CBS is the responsibility of competence center X.
Software for the CBS which, for example, is to be used on the
multi-purpose instrument, is to be coordinated matched with the
competence center responsible for the particular multi-purpose
instrument.
```

```
The door control unit should be able to control both two-door and
four-door models.  Implementation variations that have to be clarified
include:  two alternative functions or a single function that is
intended for all situations; the construction of two alternative
control units or one that fits every situation.  Variants that suit all
situations would require functions or control units that can be
configured or alternative variants for interchangeable functions and
control units.
```

3.4 Phase 4: Development of the Subsystems

In the fourth phase, all of the subsystems identified in Phase 3 that are related to function modeling and technical architecture are developed further to reach the software level, and corresponding software architecture are designed. The design of the software architecture includes, among other things, decisions about variances in the software. The software product lines are developed.

It is possible that the subsystem to be developed might only contain the function area (development of the functions and software design), and it might also only comprise the infrastructure area (development of the technical architecture at the subsystem level, and development of the control unit); it might, however, encompass all of them. Further distributions are possible. Depending on what is included, a greater or lesser amount of agreement and coordination is necessary. An administrative unit that is responsible for the system overview is to be organized at the same time. It can be a separate competence center ("an integration unit") or the work can be done through decentralized coordination among all competence centers.

If it emerges in this or a succeeding phase that the overall system description (from Phase 3) requires change, this takes place in coordination with the "integration unit" or other competence centers (in accordance with the change management structures).

Goal

The goal of this phase is to describe and to model the functions and control units of the product so that they can be implemented by a supplier or that the available functions or software components and control units can be integrated. This requires, among other things, the preparation and provision of the documents to be used by the supplier. Furthermore, all decisions about variance in the software architecture and therefore about the software product line need to be made.

End

This phase is ended when

- the functions have been described adequately for them to be implemented, i.e., the software architecture and software components have been specified and possible reference implementations for this have been designed,
- the control units have been described adequately for them to be implemented,
- the documents for the suppliers have been prepared,
- the implementation has been agreed upon with the subcontractor.

Artifacts & Activities

An overview of the artifacts and activities for this phase is given in Table 4.

Table 4 Overview of the Artifacts and Activities in Phase 4 "Development of the Subsystems"

Feature	Contents
Input	All the results of the previous phase.
Processing	– The subsystem to be implemented is adequately specified so that it can ultimately be delivered to a supplier for implementation, which might require a complete modeling process. It is also possible to subdivide the subsystem again and to implement these parts separately. – Detailed function modeling (or where appropriate, hardware-specific refinements of the function models), specification of required hardware, function distribution built on: software architecture development and specification of software components. – Consideration of existing functions and software components – Decisions about variance in the software architecture (software product lines) – In this phase it is possible to integrate the competence of the supplier.
Output	– Hardware and software description (architecture and components) as a functional requirements (models for structure and behavior) – Software as reference implementation

↪ . . .

Feature	Contents
Implementation	Competence centers and integration units; possibly in coordination with suppliers
Reuse	all possible artifacts
Variance	Variance is further refined and evaluated, and implementation concepts are developed (for example a completion schedule is set and a decision regarding 150 % approach or a software component product line approach are made, etc.)

Examples

```
Detailed specification of CBS regarding the product, function, and
infrastructure area.
Functional aspects:
 - Formulation of the algorithms for the determining the wearout of the
   individual wearing parts.
 - Conception for the redundant data storage.
Software aspects:
 - Deciding between a “master-slave” approach and a “client-server”
   approach.
 - Determining the range of functions to run on the various control
   units and the nature of the software components for this purpose.
 - Determination of the communication mechanisms.
 - Construction of a software product line for CBS.
Hardware aspects:
 - The effects of variance in the technical architecture on the range of
   the CBS are...
 - ...
```

3.5 Phase 5: Implementations

The phase represents the activities of the suppliers in the realization of the specifications drawn up so far. These are usually delivered in the form of requirement specifications. The results can be installable software components (in the form of binary code and software source code), buildable control units, or whole subsystems with hardware and software components.

Within the framework of the preparation of the requirements specifications and implementation, the suppliers can have also retrospective effects on the systems that have been modeled. These are these are then changed accordingly by mutual agreement.

This phase is not taken into further consideration within the framework of VEIA. Consequently, only an abridged table of the artifacts and activities is presented.

Table 5 Overview of Artifacts and Activities in Phase 5 "Implementations"

Feature	Contents
Input	All the results of the previous phase.
Processing	Implementation of the specified models.
Output	Implemented control units with software components.
Implementation	Mainly by suppliers.

3.6 Phase 6: Integration and Testing

The sixth phase represents the right fork of the V model. The model-based and system-oriented approach in the VEIA reference process is used to support this phase in such a way that the respective (virtual) safeguards for each the preceding phases will be carried out and the corresponding testing (test cases, etc.) developed. In the context of the product development process, this leads to an early, more economical safeguard since less total hardware is required (fewer A prototypes). The product lines to be used also offers better test coverage.

Previous integration tasks and tests will not be completely finished. However it should be possible to detect mistakes and erroneous conceptions as early as possibly and to eradicate them.

Table 6 Overview of Artifacts and Activities in Phase 6 "Integration and Testing"

Feature	Contents
Input	All artifacts from the preceding phases, in particular test cases and the supplier's intermediate products (control units, software components) that are to be assembled.
Processing	Testing and integration of the components.
Output	Test outputs, change standards on development artifact.
Implementation	Integrators, competence centers
Reuse	Experience values, test cases from earlier projects, etc., model tests
Variance	Must be adequately considered during integration and testing.

In this phase, the benefits of the modeling and the product line models of the entire reference process are demonstrated very clearly. On the one hand the artifacts and phases demonstrate that the models developed meet all of the relevant previous standards. As a result, following the implementation, the algorithms no longer need to be functionally tested since the behavior has already been verified at the model level.

In addition, testing and safeguard structures of the modeling can be employed again when the implemented systems have been tested. That means for testing purposes, the finished systems can be given the same stimulations as the models. This leads both to fewer errors being built into the finished systems and less effort being needed to find errors.

4 Case Study "Condition Based Service" (CBS)

Using the case study of *CBS* that has been provided by project member BMW for the VEIA research effort, in the following section, illustrations are given of the artifacts in the VEIA reference process that were previously discussed. This is used to illustrate the concepts and does not necessarily reflect the actual development of CBS at BMW, concerning which no decisions are made in the VEIA project. More detailed modeling of CBS in regard to the project goals will take place in later stages of the VEIA research project by the individual project members, cf. [GR06].

The following analysis of the case study, which serves to illustrate the corresponding development artifacts discussed in Section 2 and 3, is based on information the following documents provided by BMW:

- BMW Specifications for CBS 5 [BMW05a],
- Service Manual CBS for E87, E90, and E91 [BMW05b],
- The Rhapsody Model for the CBS Client.

Of note is that artifacts presented are primarily the result of reengineering. Had forward modeling been used, specific artifacts would appear different or be motivated differently. During the research project, reengineering is replaced by engineering.

Task and Purpose of the Specification Document

The specification document contains information that has to be determined in the process during Phases 1 to 4 (cf. sections 3.1 to 3.4). In the specification document itself, this is described as follows [BMW05a, p. 15]:

> This specification document sets out an explanation of the technical requirements for a system functionality that is to be developed.
>
> Depending upon the comments in the modification documentation, this document serves as a technical record of an inquiry (inquiry regarding specifications) for an offer for a development service or as a technical definition of the development objective (specification of a development objective) as a supplement to the development contract. The specification documents the current status of the stages

of development of the project and forms the basis for inquiries and negotiations.

The requirements in the specification document are fundamentally related to the functional characteristics. Determination of the system function and oversight of implementation is the task of the CBS integration unit. Responsibility for the implementation of the component functionalities lies with the respective development integration units.

The responsibility for the CBS product concept, that is, which functions are to be integrated into the vehicle and how they are to be realized, rests with the Service Technology, Market Development, Maintenance and Repair Department.

Task and Intended Purpose of the Service Manual

The service manual is a reference for BMW workshops and service and contains information compiled in accordance with product development as documentation or a user's manual. Nevertheless, information of a retrospective nature can be derived that is relevant to the formulation of strategic goals, framing conditions, product line descriptions, services, and an initial technical architecture in accordance with Phase 1 (Section 3.1) and Phase 2 (Section 3.2).

Unlike the specification documents, the present document describes the previous version, version 4.

Task and Intended Purpose of the Rhapsody Model

The Rhapsody model for the CBS client represents stages of development from Phase 4 (Section 3.4). It specifies the software architecture for CBS Clients: the interface definition and behavior descriptions. The goal is to obtain executeable, simulatable, and thus more readily understandable specifications that will become part of the specification document. In addition, reference code generated by the Rhapsody model that is ready for start-of-production will be made available to the relevant suppliers.

In the following sections, individual development views and levels of abstraction in terms of the "functionality" of CBS (context, requirements and proposed solutions) are identified in accordance with the artifacts outlined in the previous chapters and presented for discussion. They represent problem descriptions and requirements for the methodology to be developed in the research project from a case study viewpoint.

4.1 Strategic Goals / Framing conditions

Some time ago, a new maintenance concept named `Condition Based Service` (`CBS`) was introduced at BMW; it was initially called `Requirement Oriented Service` (`BOS`). Its goal was to make a break with the previous routine maintenance service in favor of a service based on wearout [BMW05a, p. 17].

The routine-maintenance-service concept is based on fixed intervals. `CBS`, in contrast, introduced a consumption-oriented service that makes maintenance and service intervals which are adapted to actual wearout possible. Both the driver and the seller are, thus,—directly or indirectly—affected or involved in maintenance as well as service.

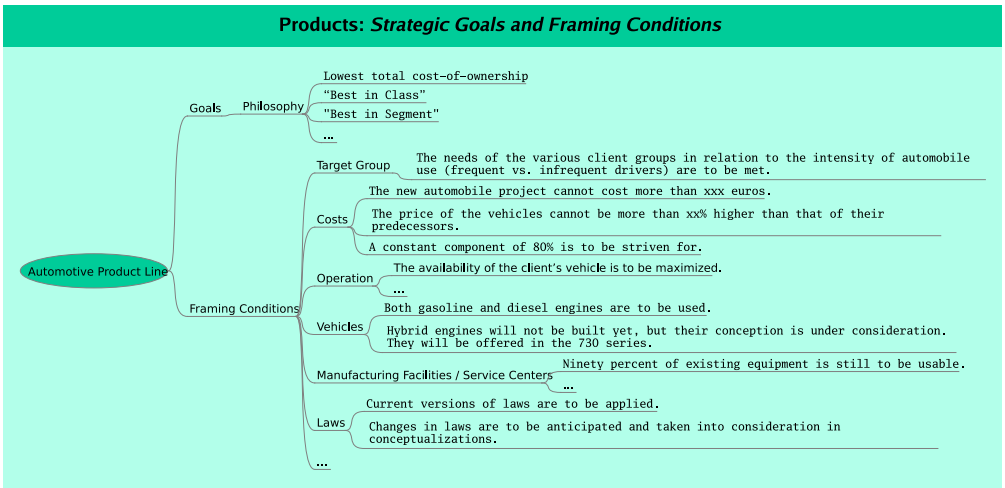
The actual wearout should be ascertained or calculated:

1 on the basis of existing sensors and / or
2 on the basis of wearout models,

or, where appropriate, with the aid of driving profiles.

The strategic goals and framing conditions for a vehicle project are given in Figure 11.

Figure 11 Products: Strategic goals and framing conditions (Mindmap, extract).



Goals and framing conditions of this type for `CBS` are found in [BMW05a, p. 17 F.] ("CBS philosophy"). They provide basic ideas as well as standards in regard to the

realization of this functionality.

The strategy for the implementation is: "as much of the available existing on-board information should be used to determine the best point for service; that is, when possible, virtual sensors are preferred over the selection of new, hardware-based sensors".

The CBS concept brings the customer perceptible improvements:

- Through the measurement of the wearout, previously necessary safety reserves can be reduced. This extends the service life of a component.
- The client's ability to combine service appointments improves mobility and increases the vehicle's availability.
- Through the measurement and evaluation of the actual wearout, clients who drive more prudently are rewarded in the form of low cost-of-ownership.

CBS pursues following goals:

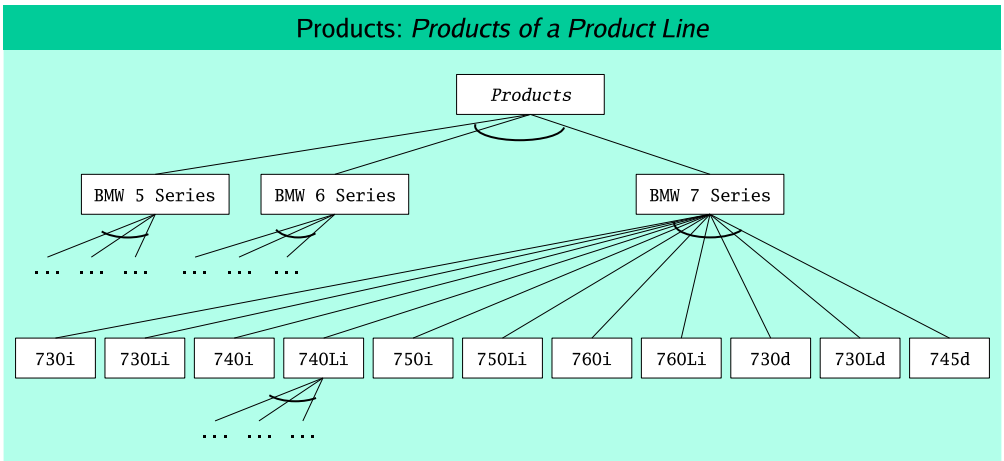
- An individualized maintenance concept related to actual vehicle wearout.
- The ability to plan, optimize, and set maintenance appointments through intelligent interval forecasting by the control units that measure wearout.
- The display of maintenance parts sorted according to a scheduling sequence.
- Improved, high-quality client consultation through visualization of vehicle data in the service reception module.

4.2 Product Line Description

In the course of defining the vehicle product line to be developed (Section 3.2), the products to be made in the product line have to be designated, and the features of all products have to be analyzed to show which commonalities and distinctions (should) exist. The product line description is the starting point of the configuration of the other development artifacts.

In Figure 12, the products of a product line (here for vehicle types in accordance with [BMW06]) are represented as a feature tree (cf. for example [KCH+90, CE00]). This Service Map and Description of Services can, as illustrated, occur at several

Figure 12 Products: Products of a Product Line (extract).



levels in the hierarchy. For example: BMWs of the 700 series are 730i, 730Li, . . . or 745d. Besides the BMW 700 series, the BMW 500 series and the BMW 600 series can also be part of the product line.

In Figure 13, a selection of the equipment features for 700 series product line, cf. [BMW06], are presented as a feature tree. Each feature is characterized as to whether every product of the product line has it (mandatory feature), whether only some products have it (optional feature), or whether there are alternative manifestations of the feature in the various products (XOR features).

In Figure 14, a selection of the equipment features of the CBS functionality is represented: which wearing parts are to be observed and the connection between them and other product features. For example, the wearing part "spark plugs" can only occur when the vehicle has a gasoline engine.

4.3 Service Map and Description of Services

The Service Map contributes to structuring functionality-related requirements to be implemented that are in the form of *services*. Strategic goals and framing conditions are included, concretized, and consolidated in the service map. The Service Map in particular is part the rough draft of the VEIA reference process research. Thus these illustrations represent preliminary rather than final results.

Figure 13 Product Features (extract).

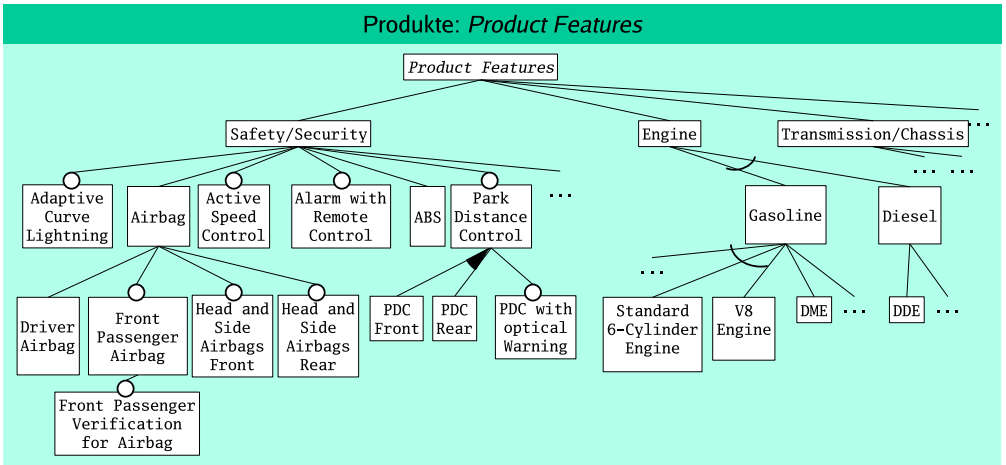
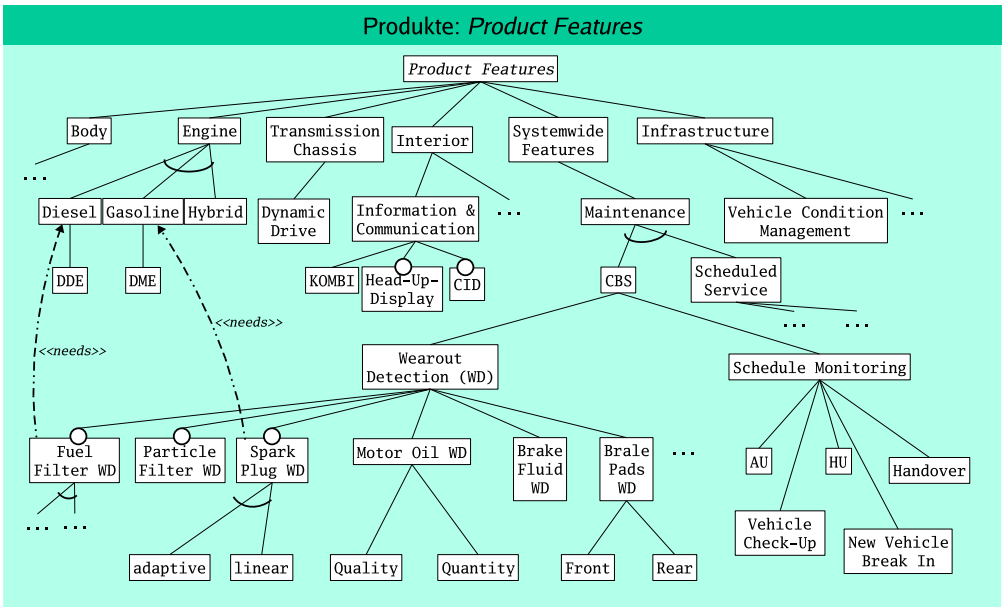


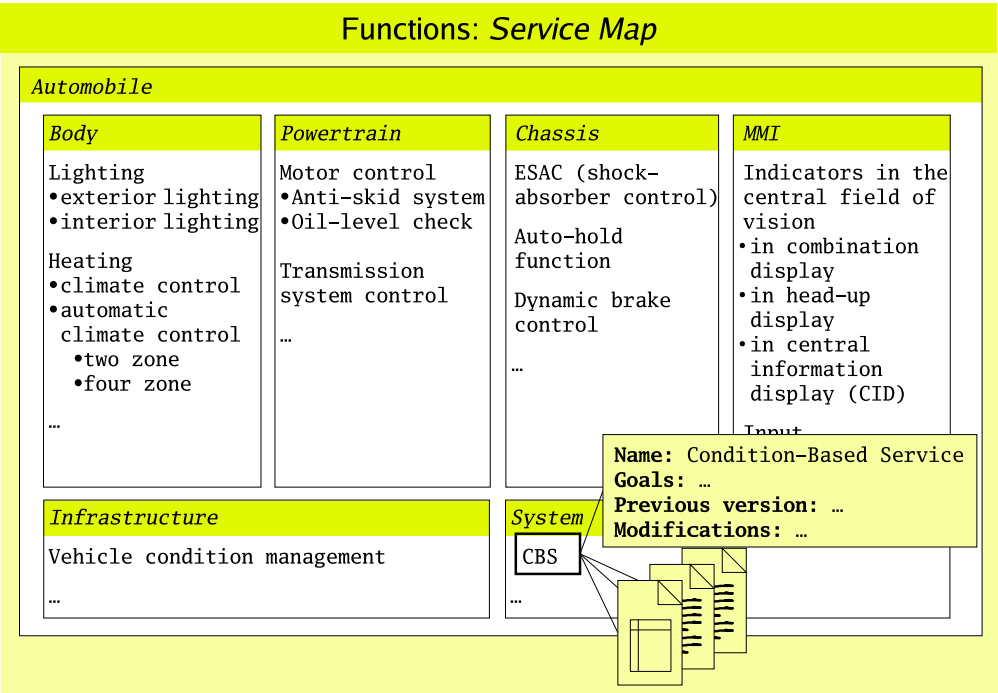
Figure 14 Product Features, Detailed for CBS (extract).



The goal is to *define* the product line to be realized and the corresponding products in regard to functionality and constraints. `Condition Based Service (CBS)` is thus just one of many services is in this system concept.

Figures 15 and 16 give two examples of a service map: Figure 15 is a schematic representation; Figure 16 is a representation as a Mindmap.

Figure 15 Service Map (Diagram).



The detailed consideration of every individual service in the service map (i.e. the functionalities that are visible "externally") provides information required by the service as to how they are processed and who is involved.

Figure 17 shows which partial functionalities are part of the service `CBS` and which system interface is necessary for it. The subservices represented in Figure 17 are described later in greater detail.

Subservice: "`Setting necessary service intervals due to wearout`"

Task:

– Ascertaining wearout—There are different methods of measuring wearout: use

Figure 16 Service Map (Illustration as Mindmap).

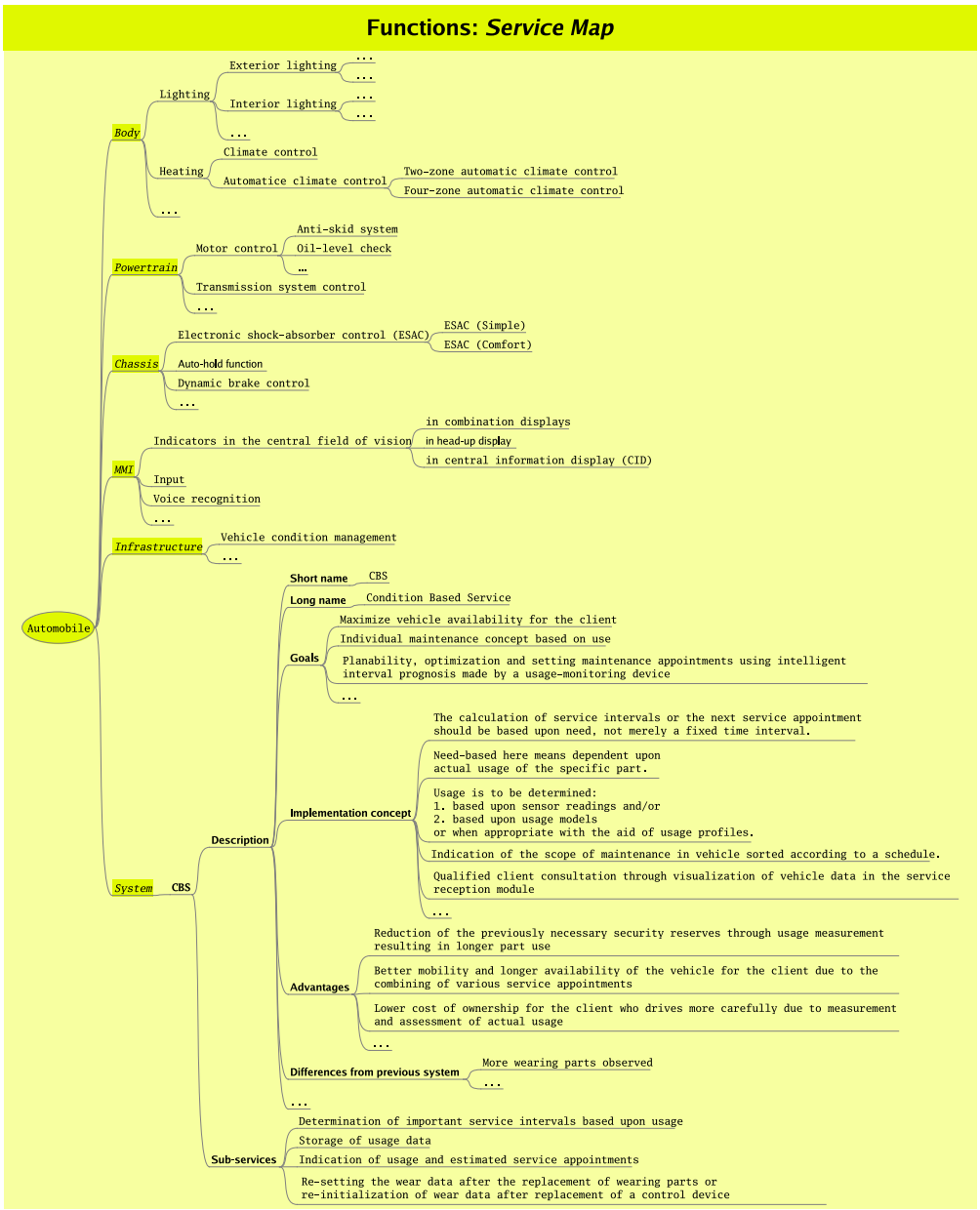
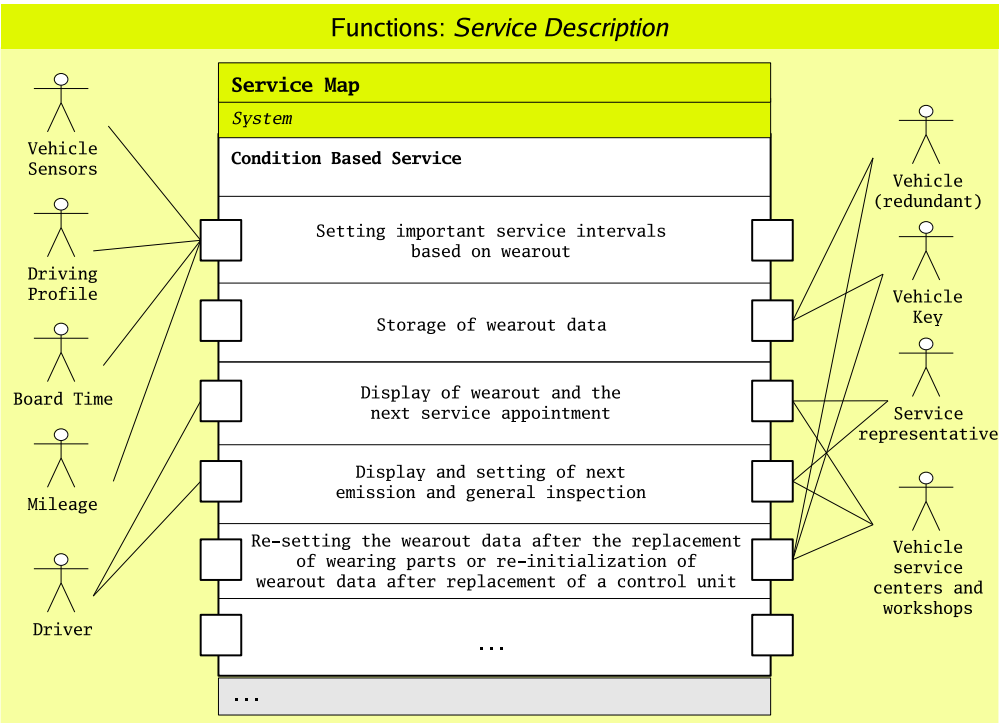


Figure 17 Service Description CBS.



oriented (based on sensor data), time and/or mileage based.

– Assessing the wearout

– Setting service intervals: Sorting, summarizing, optimization of appointments[3]

Interfaces:

– **`Vehicle sensors:`** The subservice requires information about the current status of wearing parts, for which (real or virtual) sensors are used. If possible, no "new" sensors should be installed specifically for CBS.

– **`Driving profile:`** Where appropriate, a driving profile should be considered in the assessment of wearout.

[3] In the previous realization of CBS, the intervals were set by the `CBS master`. A CBS client proposes a service appointment for the respective wearing part and sends this to the `master`.

- `Vehicle operation time:` The vehicle operation time is needed for a time-dependent wearout.

- `Mileage:` The odometer reading is needed in regard to wearout that is related to the distance driven.

Subservice: "`Storing vehicle equipment wearout data`"

Task:

- to "continuously" make the generated data persistent

Interfaces:

- `Vehicle (redundant):` The collected wearout data and set service intervals are to be stored in a persistent way, so they can be read or accessed for or after repairs.

- `Vehicle key:` For the persistent storage of collected wearout data and set service intervals the vehicle key is used. The vehicle key is read out when the vehicle is brought in for service.

Subservice: "`Display of the wearout and the next service appointment`"

Task:

- (standard or special equipment) display of the wearout data and the next service appointment

Interfaces:

- `Driver:` The relevant wearout data and service appointments are to be appropriately displayed to the driver. The type and sophistication of the display would depend on whether the vehicle has standard equipment or is fitted with deluxe options.

- `Service representative/Service Center and Workshops:` Service representative, service center, and workshops must be able to read out the wearout data and specified service appointments (via vehicle keys and `on-board diagnosis (OBD)`).

Subservice: "Displays and Setting Appointments for Emission and General Inspection"

Task:

- Vehicle inspections both for emissions and of a general nature are prescribed by law in many countries. The appointments are managed by CBS, and thus it must be possible to enter information about them into the system and access it when needed.

Interfaces:

- `Driver:` The driver can request to see emission and general inspection dates and, where appropriate, enter them into the system.
- `Service representative:` The service representative can access emission and general inspection dates and also enter them into the system.
- `Service centers and workshops:` Service centers and the workshops can enter emission and general inspection dates into the system and access them as well.

Subservice: "Re-setting the wearout data after the replacement of wearing parts or re-initialization of wearout data after replacement of a control unit"

Task:

- Reset the availability of the wearing parts (initialization, reset): if a wearing part is replaced, the corresponding control unit must be informed and re-initialized. If a control unit is replaced, the replacement must be initialized using the previous availability data.

Interfaces:

- `Service centers and workshops:` Following the replacement of a wearing part or the related control unit, it must be possible for the service centers and the workshops to reset or reconstruct the corresponding wearout data and service appointments.
- `Vehicle key:` It must be possible to access wearout data stored on the vehicle key.
- `Vehicle (redundant):` It must be possible to access wearout data stored in other vehicle devices.

4.4 Function Net / Description of the CBS Function

The functional decomposition of the overall system into functional subsystems and subfunctions is indicated with the aid of the function net. Unlike the service map, internal functions are also considered. Functions in the function net specify the implementation of the desired functionality defined in the service map by specifying how dependences between services are dealt with, for example by using controller functions or client-server relationships between functions.

The function net is used for the functional integration of individual functions and subsystems that are designed in the individual competence centers. Software architecture to be designed is prepared with function nets and functional models.

In Figures 18 and 19, the functionality outlined concerning CBS at the service level is concretized by functions. Since some subfunctions are dependent on wearing parts, a client-server architecture has been designed at the function level in which a distinction is made between a central master and distributed clients. This functional decomposition (client vs. master components) is a *design decision* at the transition from the service map to the function net and must be correspondingly documented and justified.

Both illustrations also make clear that there are different perspectives of the functionality of CBS: On the one hand there is the perspective of a vehicle project in which it is determined precisely how specific wearing parts are to be observed, cf. Figure 18 with the CBS range for the product line L6, according to [BMW05a, p. 46f.]. On the other hand there is the perspective of a software product line for CBS, whose goal is the development of a universal solution that is usable for all or at least several vehicle product lines. At the function network level, the sketch in Figure 19 gives a first approximation of this by giving an example of the basic functional architecture for CBS—independently, e.g., of the specific number of the CBS features in a vehicle.

In the illustrations, functional variation is already represented. In accordance with the wearout determination methods determined in the product feature model for the specific wearing parts, there are examples of two functional alternatives for the CBS range spark plugs: linear and adaptive.

As shown in Figure 19, the two wearout determination methods incorporate different ranges of information: whereas the linear variant refers to the mileage, the current vehicle operation time, and possibly to the driving profile (ascertained through other methods), in the adaptive variant, the sensors for the various wearing parts are also relied on.

Figure 18 CBS function: Functional distribution in master and client components (L6-range, extract).

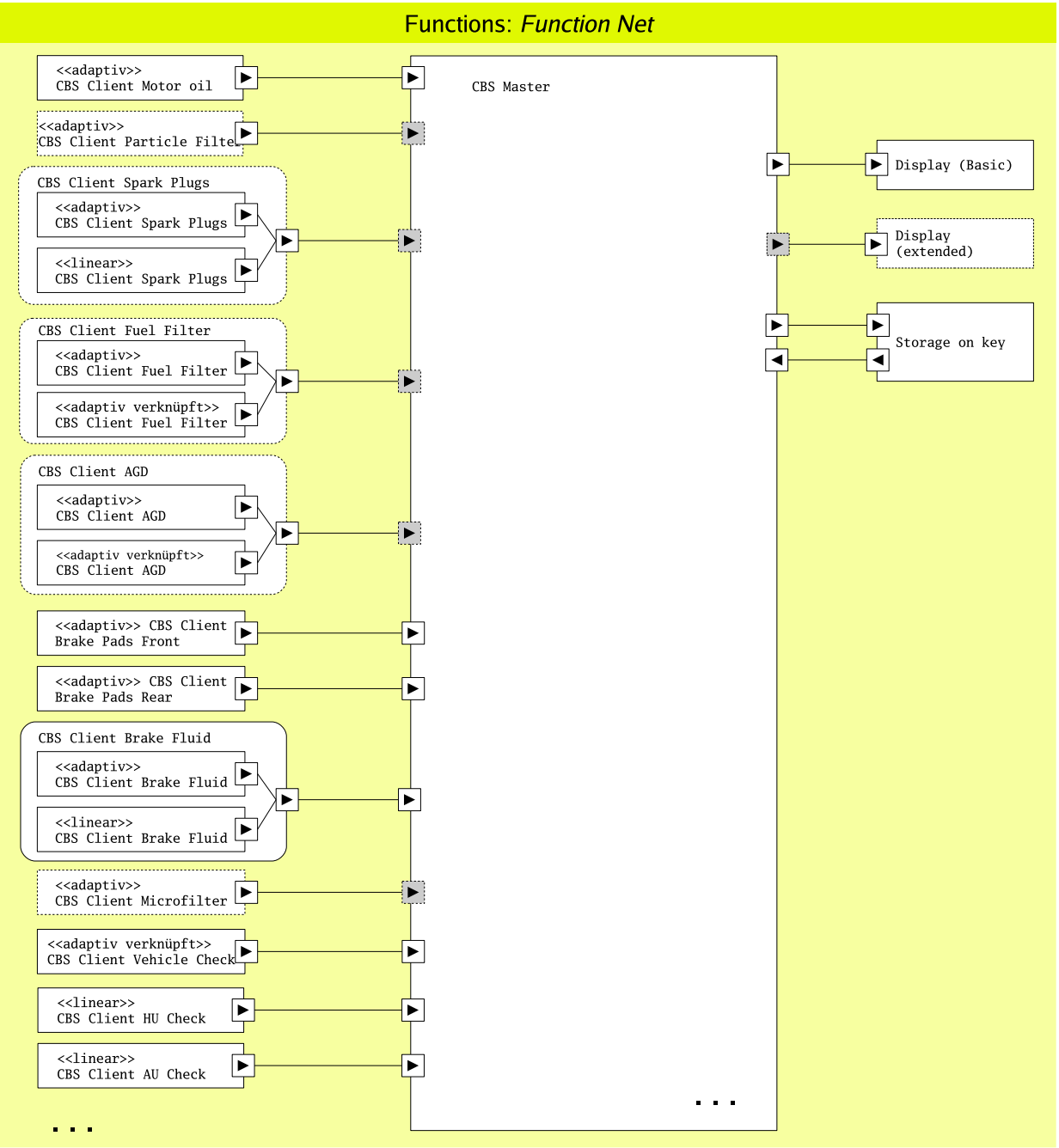


Figure 19 CBS Function: basic function distribution in master and client components (model).

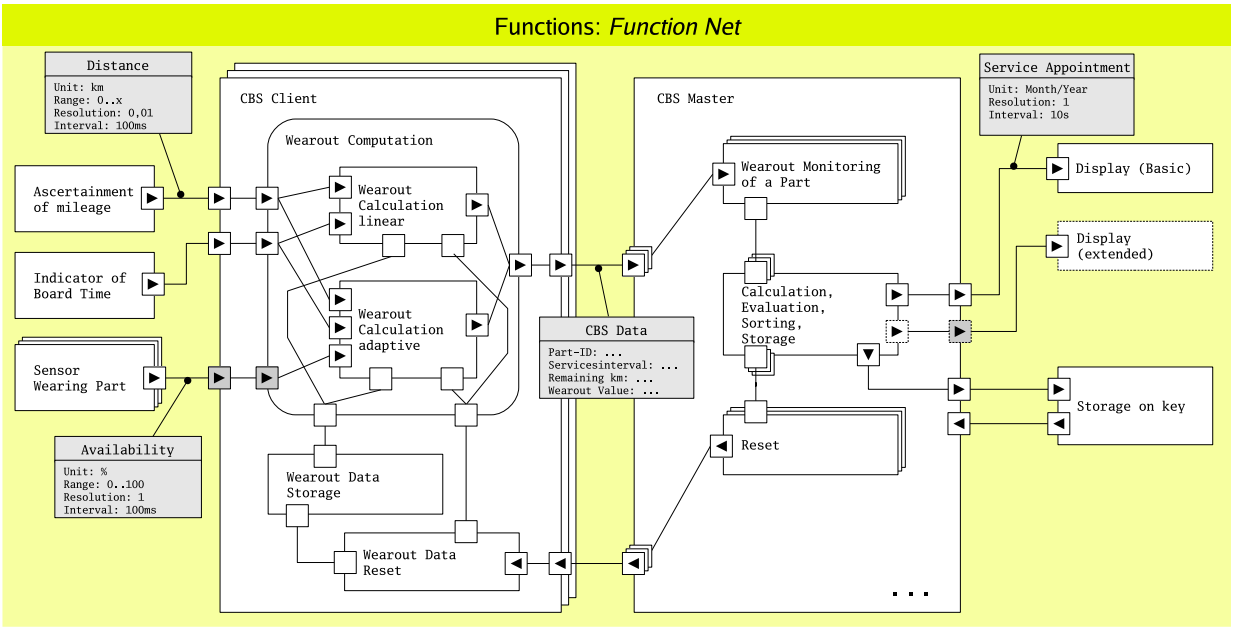


Figure 19 also depicts examples of how signals can be entered in a function net. In order to preserve clarity and because the specifications are still incomplete, only four signals are shown. When all signals are listed, it will be possible, for example, to discern which input signal influences a specific output signal.

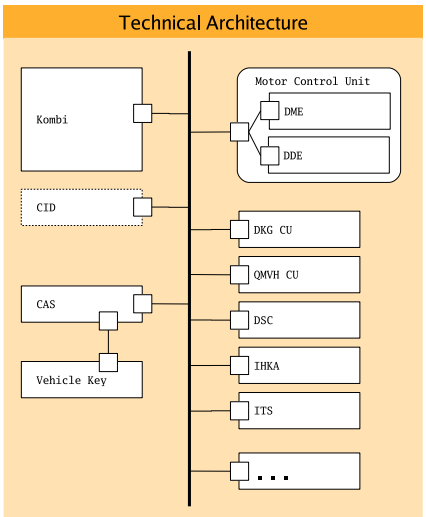
4.5 Technical Architecture

The technical architecture, relevant for CBS, which consists of the bus topology and control units, is outlined in Figures 20(a) and 20(b). The perspectives "specific vehicle product line" and "basic technical architecture" (for a software product line) can be distinguished here: Figure 20(a) shows an extract of the technical architecture for product line `L6`. Figure 20(b) generalizes this and shows the basic technical architecture that is relevant for `CBS`.

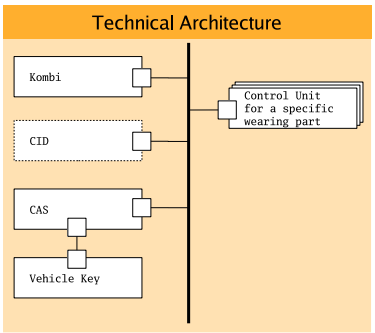
The technical architecture considers the variance specified in the product line description in relation to the optional deployment of control units (e.g., `Central Information Display (CID)` for luxury optional equipment) or in regard to control unit alternatives (e.g., the variance point `engine control units`; the alterna-

Figure 20 Technical Architecture, relevant components for CBS.

(a) *L6 Range.*



(b) *Basic Architecture (example).*



tives `Digital Motor Electronics` (`DME`) and `Digital Diesel Electronics` (`DDE`) exist because there are both gasoline and diesel engine vehicles in the product line).

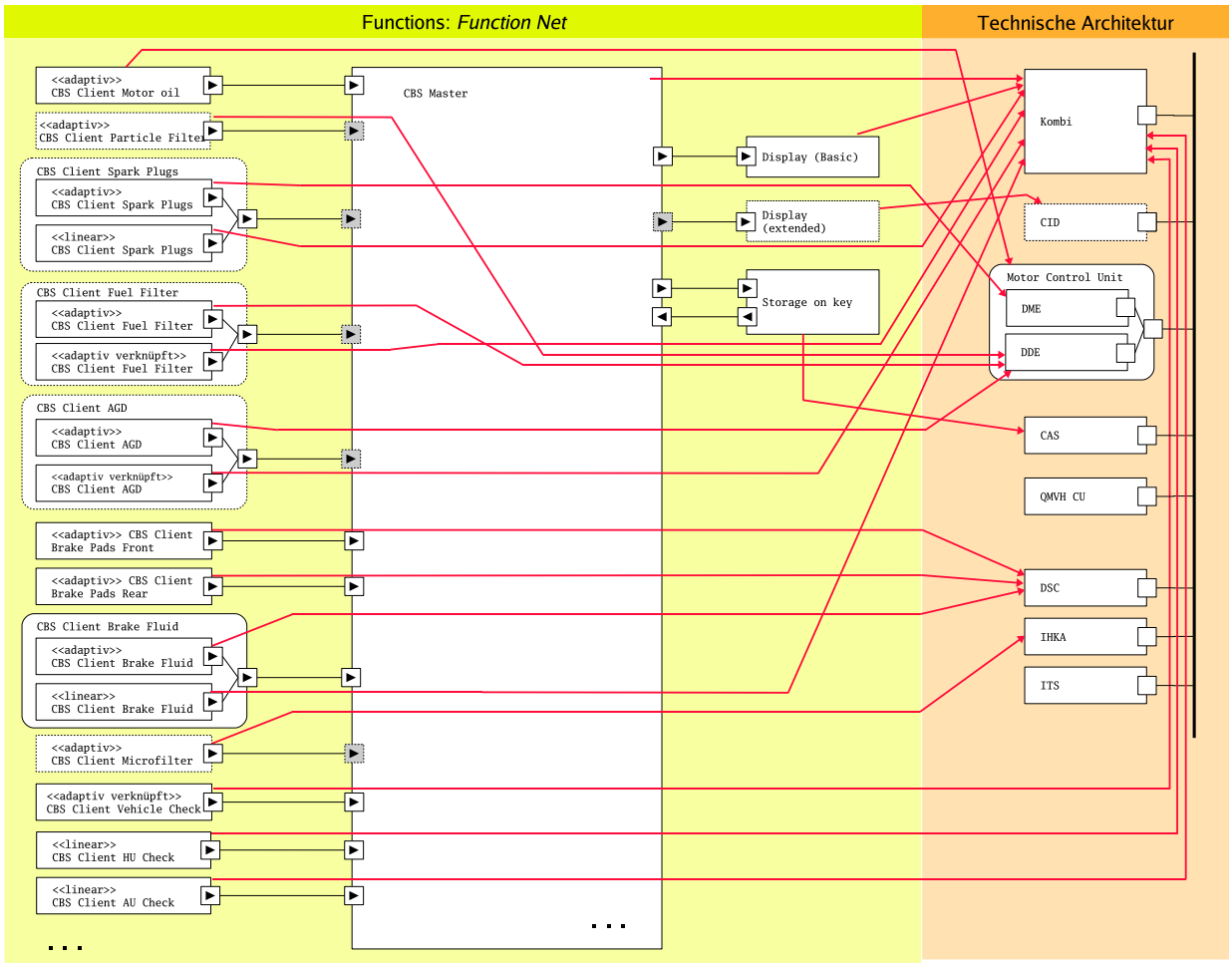
4.6 Distribution of the Function Net onto the Technical Architecture

The relationship between function net and technical architecture is represented with the aid of distribution models: The illustration of functions onto the technical architecture specifies the control unit upon which the corresponding functionality is to be implemented, i.e., on which control unit the corresponding software has to be installed.

The distribution of functions onto the technical architecture is used to define the requirements on the technical architecture based on functionality. It also makes it possible to carry out early evaluations of the compatibility of the ranges of functions and the technical infrastructure.

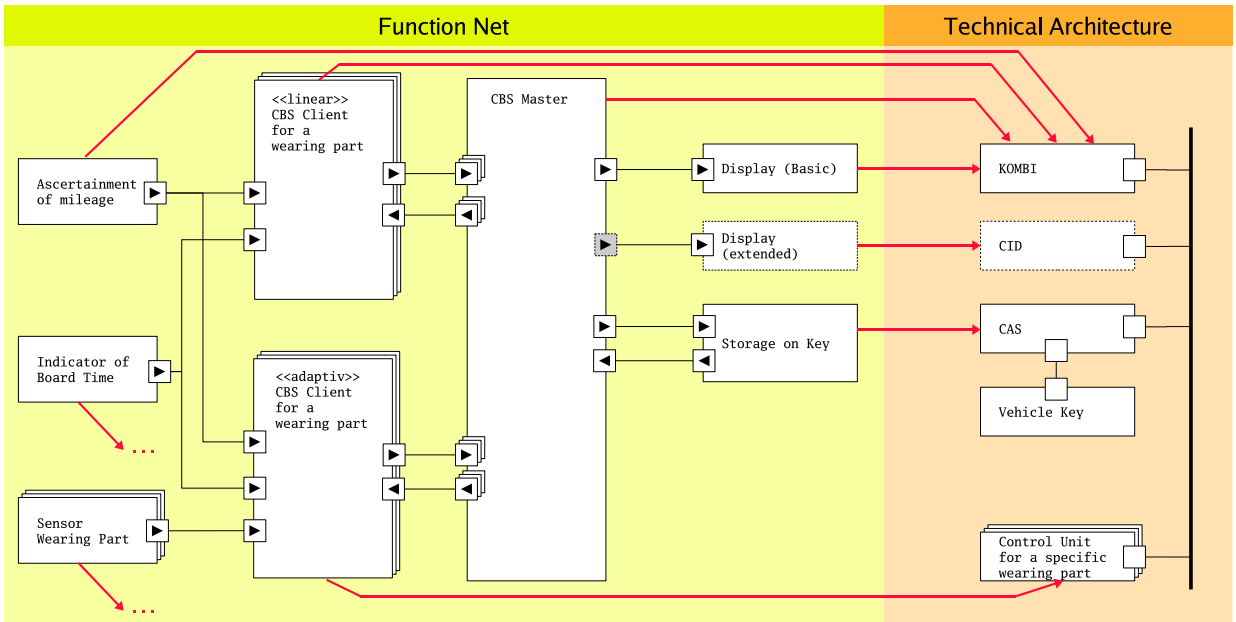
The function distribution planned for product line `L6` concerning the CBS range is in shown in Figure 21. The basic planned distribution of `CBS` subfunctions is outlined in Figure 22.

Figure 21 Function Distribution: Distribution of the CBS Functionality onto the Topology (L6-range, extract).



It can be seen in Figure 21 that the variance in the function net and the technical architecture can have effects on the distribution or can be used for it. For example, the function `CBS client motor oil` is assigned to the variance item `engine control unit`. Depending on whether the vehicle has a gasoline or a diesel engine, the function is assigned to the `DME` or `DDE` control unit accordingly. Spark plugs are only used in gasoline engines (cf. Figure 14); accordingly the function `CBS client spark plugs` is optional. It is a variant at the same time (that is, it represents a functional variance point), since two different wearout determination methods are to be found in the product line. For the function distribution, it is specified that the linear spark plug wearout determination is to be assigned to the `multi-purpose instrument (KOMBI)`, while the adaptive,

Figure 22 Function Distribution: Basic Distribution of the CBS Functionality onto the Topology (Example).



i.e. wearout determination based on the sensor, is taken over by the DME. Since this distribution is a more general model, it is singled out in the basic, exemplary function distribution in Figure 22.

Note

There are similar distribution models both for the service level and the software level, although there is a refinement relationship between the distribution models of services and functions as well as between the distribution models of functions and software. Examples of distribution models of services and software are not indicated in this document.

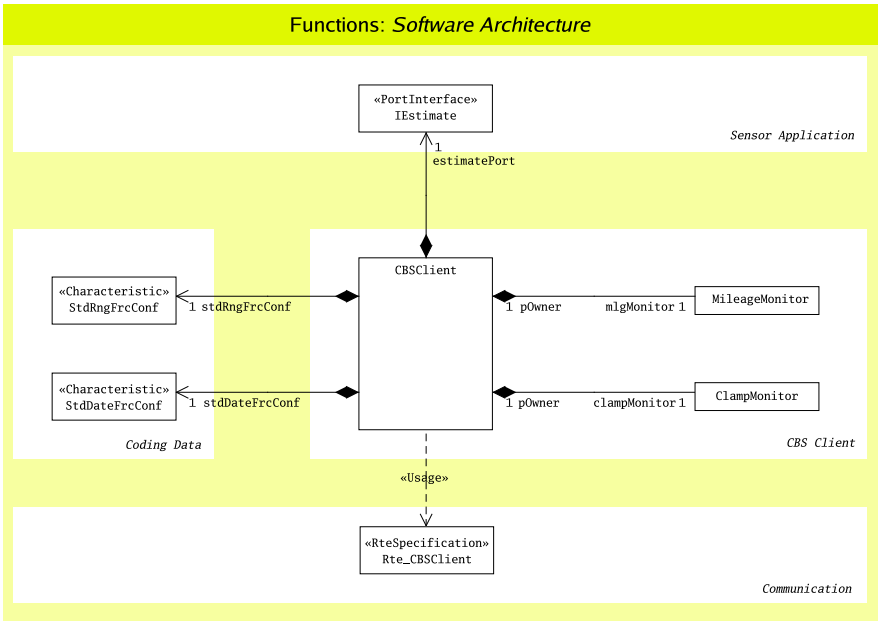
4.7 Software

To depict software factors, in the following, illustrations are given based on previous BMW design that were created with the CASE tool Rhapsody.

The basic architecture of the software implemented by the `CBS client` is shown

in Figure 23 (by means of a class diagram). The goal of the implementation of `CBS clients` is to make out software components that are independent of the control unit architecture. This is prepared, for example, by the abstraction `Rte_CBSClient` at the `communication` layer.

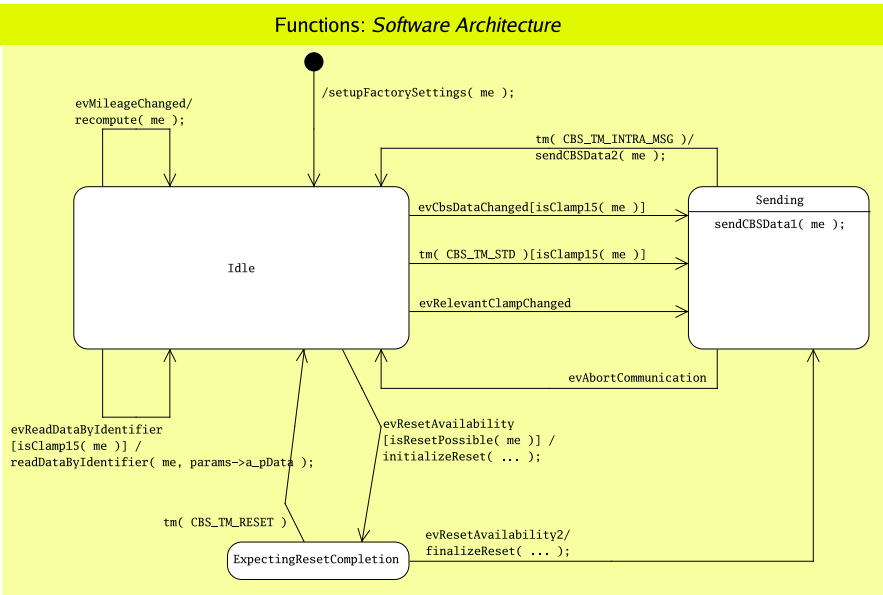
Figure 23 Software: Basic Architecture of the Software for the CBS Client (with Indication of Layers).



Next to class diagrams, behavior specifications are also part of the individual classes in the description of the software architecture. The behavior specifications make a simulation of the models possible and can also be used for the generation of (SOP) software code.

Figure 24 shows an extract from the behavior specification of the class `CBSClient` in the form of a state chart. Figure 25 depicts the basic process using an activity diagram for the operation `recompute()` of the class `CBSClient`, which is implemented in C.

Figure 24 Software: State Chart of the Class `CBSClient` (extract).



4.8 Context of the Artifacts: Variance Considerations, Distribution Models, and Foundation for the Metrics

In the following section, the previously illustrated development artifacts are discussed in context.

Within the context of the research project VEIA, a method is elaborated for how the variation in the individual evolution artifacts is registered and described due to the product line approach Variance, Distribution, and Metrics registered and described due to the product line approach and what their interrelationship is. Figure 27 shows this using the example of feature model, function net, and software components. The various CBS ranges and the various methods of determination of wearout in the products of a product line are cause for the variance shown in the function net (represented in the illustration by the varying `CBS master` function with the optional and varying ports for the spark plug wearout data).

In the situation shown, there are two basic possibilities for implementation of the `CBS master` function through software:

1 By means of *150 % software components*— as the `CBS master` is realized nowadays (cf. [BMW05a, p. 46f.] for the `L6` range): all relevant variance of

Figure 25 Software: Description of Operation `recompute()` of the Class `CBSClient` (indicated).

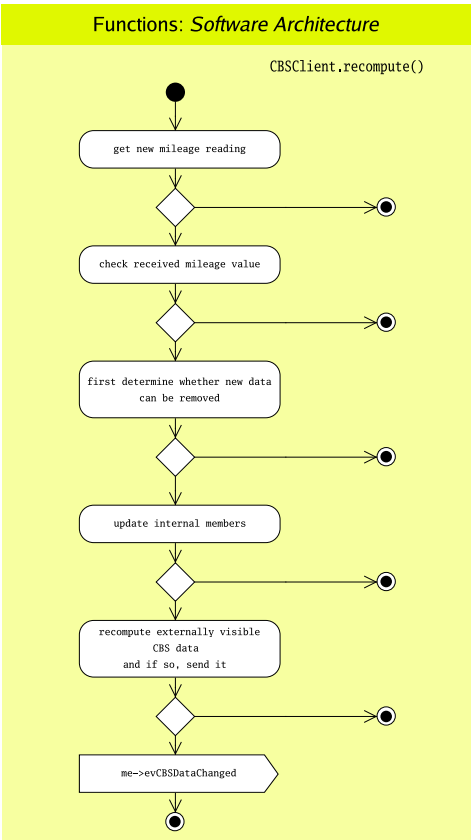
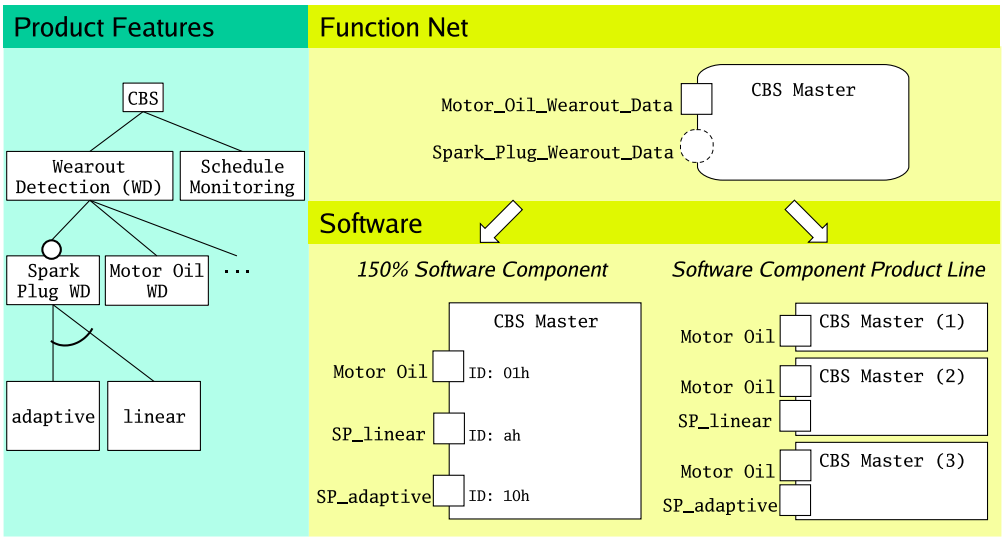


Figure 26 From the Function Net to the Software.



the feature model, the function net, and from the technical architecture are "expanded" and maintained in the software component implementing the `CBS master`;

2 By means of *software component product lines*— with which a specific software component would be made for every vehicle or vehicle type that covers exactly what has been built.

The advantage of the *150 % approach* is that a vehicle can later be re-equipped, since the newly required functionality has already been built in. Furthermore, there is less effort required in the administration of the software components because there is only one. A disadvantage is certainly that the software component contains more code than is actually ever needed ("dead code"), which means the software component occupies more storage and cause longer flashing times. In some circumstances, more effort might also be involved in parameterization. The testing work is also more complicated since it has to be guaranteed that no deactivated code is executed by mistake.

An advantage of the *software component approach* is that software components are smaller, contain less "dead code", and thus that the flashing times can be shortened. The debugging of every individual software component is simplified. A disadvantage is that more software components are involved, which have to be tested. In addition, if a vehicle is re-equipped, in some circumstances, the software component of the `CBS master` must be replaced in order to match the

new vehicle configuration.

The two implementation alternatives thus represent extremes, whereas the optimal solution is perhaps somewhere in the middle. Besides searching for an ideal way to realize the variance at the function level with appropriate software, in the research project, evaluating these potential solutions will play an important role in determining the circumstances under which one is to be preferred.

Figure 27 Development Artifacts for CBS and Their Compatibility.

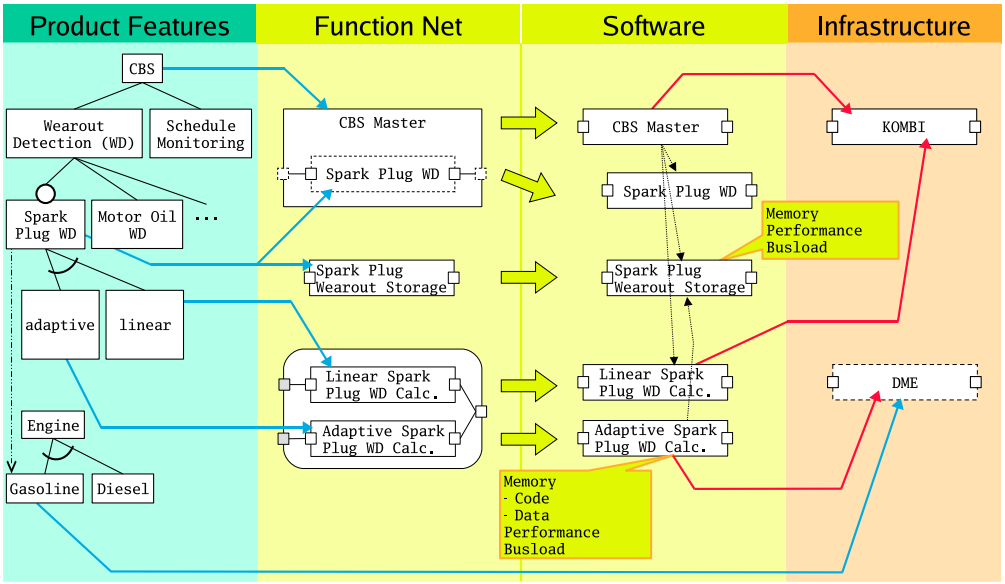


Figure 27 continues the example in a greater context, depicting, in addition, technical architecture and distributions. Important measurement variables at the software level that have to be regarded for the evaluation of the architecture are, for example, the storage requirement of a software component due to the code volume and the runtime of the data that occurs and which is to be stored. Other factors could be the necessary computing power that the control unit has to provide or the bus load, both actual and potential. Once this type of data has been collected in a project, it is possible to "reflect" it at higher levels of abstraction (e.g., on the function net) in order to be able to evaluate the architecture in a succeeding project as early a possible.

An example of the result of this type of evaluation would be the ability to make the decision for or against a software product line depending storage requirements, computing power, or bus load.

5 Conclusion and Forecast

In the "Distributed Development and Integration of Automotive Product Lines" project, the rough draft of the reference process serves as a basis for future work. In the reference process, product line description, service map, function net, software architecture, technical architecture, configuration models, and distribution models have been defined as artifacts to be constructed.

The system (or family of systems) to be developed is created with features and variance in the product line description. The service map, the function net, and the software architecture describe the function of the system. The description of the technical realization in hardware is the task of the technical architecture. The relationship models—configuration and distribution models—serve as a link between the individual artifacts.

In this way, various artifacts that describe the system to be developed are defined from different points of view. The consistence of the descriptions can be checked and guaranteed due to the cross-references. Evaluations of the system are possible with all artifacts, making them available very early in the process. Integration and testing are supported through continuous models and formal relationships between the models and can already be prepared in the left fork of the V model.

In addition to the pure description of the artifacts, a chronological process indicating the sequence in which specific partial artifacts are to be constructed is defined. Six stages are distinguished in this process, the first four of which have been explained in detail. The first four phases occur in the OEM, the fifth at the supplier. The sixth phase—integration and testing in the right fork of the V model—is, again, the task of the OEM.

Four subprojects will be appended in the future stages of VEIA: modeling of requirements, modeling of architecture, model management, and tool support. In these subprojects, methods, notations, and tools will be developed to construct the defined artifacts, i.e. to make it possible to implement the reference process. Case studies will accompany the process so that the results achieved can be evaluated on the basis of actual practice.

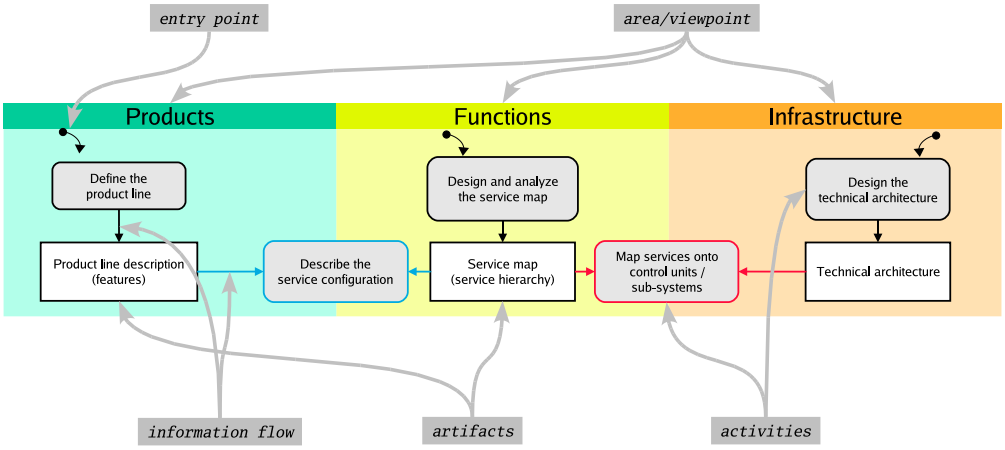
Notation

The following is an explanation of new notations used in the document. Explanations for notations already introduced in MOSES or generally known elements, e.g. from Unified Modeling Language (UML) are not included.

Artifact descriptions

Figure 28 gives an overview of the elements of artifact descriptions.

Figure 28 Artifact descriptions (Notation).



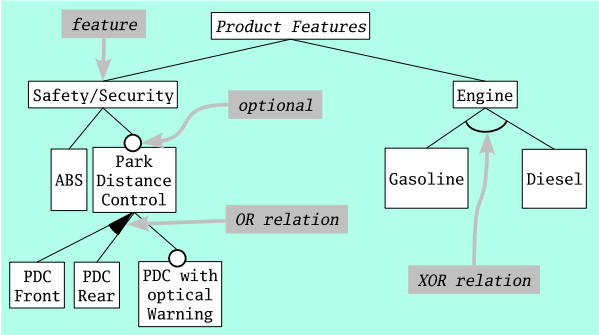
The area and/or viewpoints relevant for an E/E-system are arranged vertically. Artifacts are symbolized by white, angular boxes, activities as gray boxes with round corners. The flow of information of activity to artifact and vice versa is represented by arrows. Analogous to state charts of the UML, filled circles rings with arrows serve as representations of the entry points into the modeling.

Product Line Models

An overview of the elements of product line model descriptions is given in Figure 29.

The product features are drawn as white boxes. A stroke between two product features means: the feature above is composed of a combination of the two fea-

Figure 29 Product Line Models (Notation).



tures below together (or of several features). A circle is added to the combination to indicate optional features. Several of the features below can be in an OR or XOR relationship. An OR relationship that no features or any number of features can be used; in an XOR relationship, only one feature must be taken.

Acronyms

AGD Inlet Silencer: (German: *Ansauggeräuschdämpfer*) reduces the sucking noise of an engine.

AU AU Emission Inspection: (German: *Abgasuntersuchung*) an inspection prescribed by law that guarantees that the emission values of registered authorized motor vehicles remains within the legal emission limits during the registration period. It is a periodic inspection at designated intervals.

AUTOSAR Automotive Open System Architecture: open, standardized software architecture for the automobile industry.

BOS Requirement Oriented Service: (German: *Bedarfsorientierter Service*) old name for the system function CBS

CAS Car Access System: control unit that controls entry of the vehicle.

CBS Condition Based Service: a system function for the evaluation of the wearout of the use or wearout of vehicle resources as engine oil, spark plugs, etc. and the optimal calculation for service appointments. CBS was previously called BOS at BMW.

CID Central Information Display: a display in the middle of the console.

DDE Digital Diesel Electronics: (German: *Digitale Dieselelektronik*) motor control electronics for diesel engines.

DKG Double-clutch transmission: (German: *Doppelkupplungsgetriebe*) a special transmission that makes faster switching of gears possible. The DKG transmission oil is monitored as part of CBS.

DME Digital Motor Electronics: (German: *Digitale Motorelektronik*) motor control electronics for gasoline engines.

DSC Dynamic Stability Control: (German: *Dynamische Stabilitäts-Control*) optimizes both the driving stability at start-up and acceleration as well as tire traction.

EDC electronic shock absorber control: (German: *Elektronische Dämpfercontrol*) improves comfort and vehicle security by stepless adjustment of the shock absorber pressure strength smooth and adapts to variations in road, load, and going-driving conditions.

HU General inspection: (German: *Hauptuntersuchung*) guarantees the maintenance of vehicle standards— a periodic inspection at fixed intervals.

IHKA Automatic heating / air conditioning: (German: *Integrierte Heiz-Klima-Automatik*) automatic regulation of temperature in the vehicle interior.

IHR Standard temperature-control equipment: (German: *Integrierte Heizreglung*).

ITS Intelligent Tire System: Control for the monitoring of tire wearout. (Source: http://www.conti-online.com/)

KIFA body, interior, chassis, powertrain: (German: *Karosserie, Interieur, Fahrwerk, Antriebsstrang*) a possible classification of vehicle domains

KOMBI multi-purpose instrument: (German: *Kombiinstrument*) display between speedometer and rotation speed indicator

MOSES Model-based Systems Engineering: (German: *Modellbasierte Systementwicklung*) was an R&D project commissioned by BMW Group and carried out by Fraunhofer ISST, cf. [Kle06].

MOST Media Oriented Systems Transport: A serial bus system for the transfer of audio, video, speech, and data signals via optical wave conductors or electrical conductors.

NO_x Nitrogen oxide: (German: *Stickoxid*) Emissions caused by the burning of fossil fuels. CBS monitors NO_x additives.

OBD on-board diagnosis: A function of the DME that recognizes malfunctions before they cause damage. Error messages are stored and later displayed on the screen of the BMW information diagnosis system's in the workshop.

QMVH Rear lateral moment control unit: (German: *Quermomentenverteilung im Heck*) semi-active rear-axle kinematic that influences the distribution of the lateral forces via the longitudinal axis. CBS includes monitoring of the rear axle transmission fluid.

SAM Service Receipt Module: (German: *Service Annahme Modul*) A control unit that reads and displays CBS data and other information.

UML Unified Modeling Language: A standardized language developed by the Object Management Group (OMG) for the modeling of software and other systems.

VEIA Distributed Development and Integration of Automotive Product Lines: a joint project supported by the German Federal Ministry for Education and Research in the framework of the research offensive "Software 2006". (German: *Verteilte Entwicklung und Integration von Automotive-Produktlinien*)

The following acronyms appear in the illustrations, but not in the text: AU, AGD, CAS, DKG, DSC, HU, IHKA, IHR, ITS, NO_x, QMVH, SAM

References

[BMW05a] BMW Group. *Lastenheft CBS 5*, October 2005. SAP-Doknr.: 10000943-000-03.

[BMW05b] BMW Service: Technik. *Condition Based Service: E87, E90, E91*, May 2005.

[BMW06] BMW Group. *BMW 7er-Produktkatalog*, September 2006.

[Bro05] Manfred Broy. Service-oriented systems engineering: Specification and design of services and layered architectures. the janus approach. In *Engineering Theories of Software Intensive Systems*, pages 47–81. Springer, 2005.

[BS01] Manfred Broy and Ketil Stølen. *Specification and development of interactive systems: Focus on streams, interfaces, and refinement*. Springer-Verlag, New York, 2001.

[CE00] Krzysztof Czarnecki and Ulrich W. Eisenecker. *Generative Programming – Methods, Tools and Applications*. Pearson Education. Addison-Wesley, 2000.

[Fra03a] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme, Mollstraße 1, 10178 Berlin, Germany. *MOSES 2: Modellierung von hierarchischen, vernetzten Funktionen. Eine Methodik für die BMW Group*, July 2003. Projektabschlussbericht für BMW Group.

[Fra03b] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme, Mollstraße 1, 10178 Berlin, Germany. *MOSES 3.1: Architekturmodellierung, Teil 1, Metamodell für technische Architekturmodelle und Partitionierung von logischen Funktionen*, September 2003. Projektabschlussbericht für BMW Group.

[Fra04a] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme, Mollstraße 1, 10178 Berlin, Germany. *MOSES 3.2: Architekturmodellierung, Teil 2, Erweiterte Modellierung und Bewertung von Partitionierungen*, January 2004. Projektabschlussbericht für BMW Group.

[Fra04b] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme, Mollstraße 1, 10178 Berlin, Germany.

MOSES 4.1: Validierung und Umsetzung der modellbasierten Entwicklungsmethodik MOSES, Teil 1: Konfiguration, Produktlinien- und Domänenmodelle für Funktionsnetzwerke, September 2004. Projektabschlussbericht für BMW Group.

[Fra05a] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme, Mollstraße 1, 10178 Berlin, Germany. *Das MOSES-Gesamtmetamodell*, April 2005. Projektabschlussbericht für BMW Group.

[Fra05b] Fraunhofer-Institut für Software- und Systemtechnik, Abteilung Verlässliche Technische Systeme, Mollstraße 1, 10178 Berlin, Germany. *MOSES 4.2: Validierung und Umsetzung der modellbasierten Entwicklungsmethodik MOSES, Teil 2: Produktlinien- und Domänenmodelle technischer Architekturen*, April 2005. Projektabschlussbericht für BMW Group.

[GHHJ06] Alexander Gruler, Alexander Harhurin, Judith Hartmann, and Elmar Jürgens. Veia – ebenenhierarchie. Draft, Juli 2006, July 2006.

[GR06] Martin Große-Rhode. Verteilte Entwicklung und Integration von Automotive"=Produktlinien (VEIA). Projektantrag, May 2006. Projektantrag im Rahmen der BMBF"=Forschungsoffensive "Software Engineering 2006".

[KCH+90] Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, and A. Spencer Peterson. Feature-oriented domain analysis (foda) – feasibility study. Technical report cmu/sei-90-tr-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, November 1990.

[Kle06] Ekkart Kleinod. Modellbasierte Systementwicklung in der Automobilindustrie – Das MOSES-Projekt. Isst-bericht 77/06, Fraunhofer-Institut für Software- und Systemtechnik, Berlin, Abteilung Verlässliche Technische Systeme, April 2006.

[KMM05] Ingolf H. Krüger, Reena Mathew, and Michael Meisinger. From scenarios to aspects: Exploring product lines. In *Proc. 4th Int. Workshop on Scenarios and State Machines: Models, Algorithms and Tools*, 2005.

[KSTW04] Leonid Kof, Bernhard Schätz, Ingomar Thaler, and Alexander Wisspeintner. Service-based development of embedded systems. In *Net.Object Days Conference, OOSE Workshop, Erfurt, Germany*, 2004.

[PBL05] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering – Foundations, Principles and Techniques*. Springer-Verlag, 2005.

[SS03] Bernhard Schätz and Christian Salzmann. Service-based systems engineering: Consistent combination of services. In *Proc. ICFEM 2003, 5th Int. Conf. on Formal Engineering Methods*, LNCS 2885. Springer, 2003.