Proceedings of the 19th CIRP Design Conference – Competitive Design, Cranfield University, 30-31 March 2009, pp420

Function Orientation beyond Development – Use Cases in the Late Phases of the Product Life Cycle

A. Warkentin[1], J. Gausemeier[1], J. Herbst[2]
[1]Heinz Nixdorf Institute, Germany, Fürstenallee 11, D-33102 Paderborn
Andreas.Warkentin@hni.uni-paderborn.de, Juergen.Gausemeier@hni.uni-paderborn.de
[2]Daimler AG, Germany, Wilhelm-Runge-Str. 11, D-89081 Ulm
Joachim.J.Herbst@daimler.com

Abstract
The importance of a function driven way of working in the field of Electric/Electronic-systems (E/E) is increasing. However, the existing methods are focusing on the development phase. In contrast to this, we performed a comprehensive use case analysis concentrating on the late phases of the product life cycle. In this paper we describe the results of this analysis by illustrating the main use cases identified. For each use case we present a solution of how to exploit potential of function orientation. Based on this we will be able to define a concept of a function-oriented representation.

Keywords:
Functions, Function Orientation, Product Life Cycle, Manufacturing Process Planning, Manufacturing

1 INTRODUCTION

Modern automobiles have a huge amount of innovations inside and are characterized by high complexity, especially concerning Electric/Electronic systems (E/E). There are many functions which are distributed over several components. At the moment, the way of working is oriented towards the components of an automobile. For example, this becomes apparent in the product documentation which is focused on components. Moreover, the arrangement of the organizational structure in development is influenced by components. In addition, development processes concentrate on components. However, this component driven way of working is not sufficient to deal with the complexity of today's automobiles.

In order to meet this challenge, there is an ongoing paradigm shift towards function orientation. Function orientation implies that the functions of an automobile are being considered explicitly, i.e. by documenting functions or including functions into development processes. This way it is easier to perceive the interdependencies within a product. Moreover, functions are the most important issue of a product from the customer's point of view. By having an explicit view on functions it is possible to ensure that these functions are fulfilled at the end of the development process.

The use of functions in the early stage of the product development process has often been addressed in recent research. For example, in [1] and [2] the focus is directed to the usage of functions in order to find new product concepts and solutions. We are convinced that function orientation can also generate an additional benefit beyond the development phase, i.e. during manufacturing process planning, manufacturing and usage of a product. Therefore we performed a comprehensive use case analysis at an automotive OEM. The goal of this use case analysis was to identify areas in which a function-oriented representation improves certain tasks and to analyze how a function-oriented representation has to look like in order to support these tasks. So, a use case in our context describes a situation in which a function-oriented representation is helpful. We concentrated on the phases beyond development. In this paper we describe the results of this analysis by illustrating the main identified beneficial use cases and presenting possible solutions.

The initial modeling of information contained in a function-oriented representation of a product is associated with time and effort. In the same way, maintenance of this information throughout the product life cycle is expensive in labour. Consequently, it is important to know which elements of a function-oriented representation lead to a benefit in the different phases of the product life cycle. This way it is possible to find an optimum between the effort associated with modeling and updating of information on the one hand and the benefit associated with the usage of this information on the other hand. Current approaches which deal with a function-oriented representation are not concerned with this question. So, there are different approaches to represent a product in a function-oriented way but they do not answer the question which elements out of the representation have to be modeled and updated throughout the product life cycle. Our use case analysis, by contrast, shows the different benefits resulting from a function-oriented representation in the form of use cases in the late phases of the product life cycle and presents possible solutions. These solutions define the corresponding elements needed from a function-oriented representation. Having this information will enable us to find an optimum concerning the elements to be modeled. Besides, the use cases presented in this paper are assigned to different points in time. These circumstances have an impact on the need to update the elements of a function-oriented representation. Consequently, a detailed analysis of the use cases concerning the positioning in the product life cycle will enable us to define the points in time where the relevant elements have to be updated.

The remainder of this paper is structured as follows. Section 2 discusses related work. Section 3 describes the basic terms in our context of function orientation while section 4 presents several beneficial use cases and corresponding solutions. Section 5 concludes with a summary of the main results and an outlook on an approach which allows configuring an appropriate function-oriented representation on the basis of desired solutions for certain use cases.

2 RELATED WORK

In this section, relevant design methodologies in the context of a function-oriented representation are discussed.

In the past decades, the utilization of functions has become an important part of several general design methodologies, e.g. [1], [3], [4] and [5]. In the following we describe some of these methodologies and mention the basic concepts concerning function-orientation contained in these methodologies.

Axiomatic Design [3] aims at guiding the decision-making process within the development of new products by means of two axioms. Here, SUH defines several levels of abstraction, the so called domains, which are used to handle the complexity of a design task. In the context of this paper, the customer domain, the functional domain and the physical domain are important. The customer domain is composed of customer needs. The functional domain contains functional requirements which are derived from customer needs whereas the physical domain comprises relevant design parameters. The elements contained in one of these domains are mapped to elements in the following domain. Another basic concept used in Axiomatic design follows from the complexity of products: to describe the functional aspects, the definition of a single function is not sufficient. Therefore, functions are decomposed to subfunctions using decomposition relations. This decomposition leads to a function hierarchy or a function tree respectively.

PAHL and BEITZ [1] defined another well-known methodology aiming at the development of new products. Here, functions are used to find appropriate solutions. In this approach, functions express desired relations between inputs and outputs within a system. Thus, apart from the decomposition relations, functions are also connected with flows of energy, material or information. This is expressed with the term function network. Moreover, PAHL and BEITZ define five general function classes.

The development of such a standard vocabulary to describe functions (also known as function taxonomy) has been addressed by several approaches, e.g. [4], [6], [7], [8]. An overview can be found in [8]. The objective of such taxonomies is to establish a universal language to facilitate communication during the design process and to simplify the search for appropriate solutions.

Another example of utilizing functions in the design process can be found in the specification technique for the description of the principle solution of self-optimizing systems as shown in [5]. This specification technique consists of several partial models which represent different aspects of the system to be developed. A function hierarchy is one of these partial models. The functions contained in the hierarchy are developed from defined requirements and are used to derive solutions patterns.

In the area of automotive E/E there are also several methods to describe systems with consideration of functional aspects, e.g. [9], [10], [11] and [12]. These methods adapt the basic concepts mentioned before to the description of E/E-systems. Here, the utilization of several levels of abstraction as shown in Axiomatic Design is also widely accepted.

In [9] a specification technique for the description of automotive E/E-systems in the design phase is defined. It consists of three levels of abstraction. In the first level, among other things, the expected functions from the customer's point of view are described. The second level called logical architecture comprises functions on a logical level. The third level, the technical architecture, contains information concerning technical realization subdivided in software and hardware.

A similar approach for the development of automotive E/E-systems is described in [10]. Here, functions are described as they are perceived by the user. These functions are mapped to software and hardware.

The approaches described in this section provide a basis for the function-oriented representation of automotive E/E-systems. However, these approaches focus on supporting the development phase within the product life cycle and they do not deal with the question whether a function-oriented representation of the product should be continuously documented and maintained and up to which point in the product lifecycle this should be done. In contrast the goal of this work is to answer the questions how to utilize and profit from a function-oriented representation of automotive E/E-systems in the following phases of the product lifecycle, how to adapt such a representation for this purpose and what the documentation process for the function-oriented representation should look like. Therefore, we performed a comprehensive use case analysis presented in this paper.

3 BASIC CONSIDERATIONS

In this section, we introduce terms which are used in the remainder of this paper. These terms are based on certain approaches in the field of automotive E/E described in section 2.

In the remainder of this paper, we use three levels of abstraction and corresponding terms that are based on the approaches described in [9] and [10]. In the first, most abstract level functions are presented as they are perceived by the user or customer respectively. This also includes a high-level description concerning the expected behaviour of an E/E-system. On this level, functions are independent of realization details. This level is called **user level**. To represent the user level, function hierarchies are often used. Figure 1 shows an exemplary function description on the user level. The function "to control tire inflation pressure" is decomposed in two subfunctions which are directly perceivable by a user. These functions are independent of realization details.

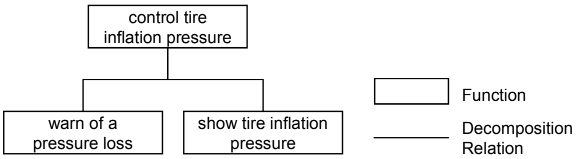


Figure 1: Exemplary User Level.

In contrast to the user level, the second level concentrates on the way the functions are realized on a logical level. Therefore, this level is called **logical architecture** or design level. Here, the description of functions is more detailed. The logical architecture contains a decomposition of functions and information concerning the in- and output on a logical level. Another important issue is the description of behaviour of a function, e.g. via a state transition process.

Figure 2 shows an exemplary logical architecture which concretizes the functions shown in Figure 1. It becomes obvious, that a logical architecture contains also functions which are not perceivable by the user, for example the function "capture tire inflation pressure". Moreover, Figure 2 shows that a logical architecture contains assumptions concerning the realization of functions as the illustrated functions describe only one possible solution. The function "warn of a pressure loss" could for example also be realized by comparing the number of rotations between the left and the right tire. In this case, the logical architecture would be different whereas the function shown in Figure 1 would be the same for both possible solutions.

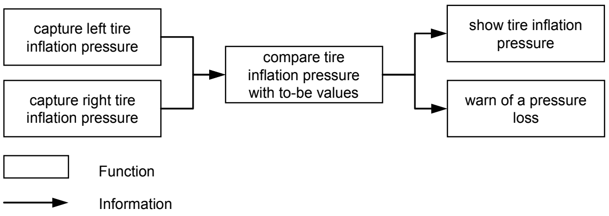

capture left tire inflation pressure
capture right tire inflation pressure
compare tire inflation pressure with to-be values
show tire inflation pressure
warn of a pressure loss
Function
Information

Figure 2: Exemplary Logical Architecture.

The third level describes the technical details of E/E-systems. Therefore, this level is called **technical architecture**. The technical architecture consists of hardware and software architecture. The hardware architecture includes the physical components of an E/E-system. Above all, these are actuators, sensors as well as control units. The software architecture describes the software components of an E/E-system. In our context, the relations between functions of a logical architecture and elements of the technical architecture, i.e. hardware and software components, are important. These relations describe which parts, i.e. hardware and software components, contribute to the fulfillment of the related function. These relations are called **mapping relations**. There is a wide range of possible levels of detail concerning the modeling of mapping relations, i.e. the target of a relation can be on different levels of the logical architecture or technical architecture, respectively. For example, a function can be related to a control unit. A more detailed relation could link a certain information output of a function to a physical connection between hardware components.

4 USE CASES

In this section several beneficial use cases identified in our analysis are described and corresponding possible solutions are presented. The first use case is settled in manufacturing process planning and deals with the prioritization of functions to be tested in manufacturing. Another use case focusing on manufacturing process planning is the specification of test cases for functional testing. The third use case is occupied with the extraction of compatibility information for certain purposes in manufacturing and usage. The fourth use case focuses on capturing of customer feedback during the usage of a product and the last use case describes the update of functions.

4.1 Prioritization of functions to test

Functions on the user level represent the customer's view on a car. Thus, testing functions is the direct way to ensure the functional aspects of a car's quality from the customer's perspective. Via testing of functions in manufacturing it is possible to assure that functions are fulfilled at the end of the manufacturing process. Moreover, the high number of variants of modern automobiles increases the importance of testing of functions during manufacturing. The following examples illustrate the high number of variants: Audi states that there are 10^{20} possible configurations, at Daimler, there are 10^{27} possible configurations and at BMW 10^{32} [13], [14]. Therefore, only a restricted percentage of the possible configurations can be tested in the development phase. Testing of functions in each possible configuration would result in an unreasonable effort. Moreover, this effort is unnecessary as not each possible configuration is being actually ordered. This shortcoming can be resolved by an additional testing of functions during manufacturing as the tests are applied on a customer's car, i.e. on a particular configuration.

On the one hand, testing of functions during manufacturing is important as we have mentioned. On the other hand, this testing of functions causes a high effort as there are more than 2000 functions in a car [15]. It is not feasible to test all of these functions during manufacturing. Therefore, there is a need to prioritize functions to be tested on the basis of defined criteria.

In order to find a solution for prioritizing functions, the failure mode and effects analysis (FMEA) and the field of risk management are helpful. These approaches address a similar issue. In FMEA and risk management, the following factors are relevant: probability of a failure and consequences of a failure [16], [17], [18]. In FMEA, detectability of a failure is additionally taken into account.

Thus, according to these approaches, following influencing factors have to be taken into consideration in order to prioritize functions to be tested:

- Severity of the consequences caused by a failure in a function: This factor describes the seriousness of consequences that result from a defective function from the customer's point of view.
- Probability of a failure in a function: This factor describes the likeliness of a failure to occur in a function.
- Probability of detecting a failure (detectability): This factor describes the likeliness to find a failure before a product arrives at the customer.

The combination of these three factors leads to the prioritization of functions to be tested.

There are several ways to determine values for the three factors. The first alternative is to estimate values in a subjective manner on the basis of the knowledge of experts. Thus, it is possible to prioritize functions without a comprehensive basis of information concerning functions, e.g. information about the mapping relations between components and functions. Only a documentation of functions on the user level of an automobile is needed for the estimation of values in a subjective manner. Moreover, a documentation of the logical architecture might be helpful as it provides a better inside into consequences of a failure of a function. Here, the effect of a failure in a function on other functions becomes transparent.

Another way is to determine or calculate estimated values for severity, probability and detectability on the basis of detailed information as shown in Figure 3. The following examples shall deliver an insight into the possible information that could be taken into consideration.

For an estimation of severity criteria like the safety relevance and the importance of a function from a customer's perspective can be helpful. The safety relevance specifies whether there is a hazard when the considered function is not fulfilled or not. Consequently, this is very important information for prioritizing functions. Furthermore, the importance of a function for the customer should be regarded. Thus, a documentation of functions on the user level and of values for these criteria for each function would be helpful for the estimation of severity.

For a determination of probability of a failure in a function it is helpful to take, among other things, the complexity and error rates of related components into consideration. Complexity can for example be estimated on the basis of the number of hardware and software components that are necessary to fulfill the considered function. Here, information like the lines of code (LOC) of participating software can give an additional hint concerning complexity of a function. Moreover, existing information regarding error rates of the components related to the considered function improves the determination of the probability of a failure. To sum it up, information about mapping relations between components and functions is important for a determination of probability of a failure in a function.

The probability of detecting a failure is influenced by many criteria. In the context of our use case analysis, i.e. at the OEM the detectability is especially influenced by the ability to test the physical connections between components that are related to the considered function. The reason is that in manufacturing the testing of connections between components is dominating. With testing of single components and connections between components there is a kind of an implicit testing of functions. So, if connections between components that are contributing to a function are not testable, the considered function cannot be tested implicitly. Consequently, the probability of detecting a failure in this function is quite small when tests are limited to connections. The ability to test physical connections is determined by the type of involved components and the corresponding connections. Consequently, information about the mapping relations between functions and components and especially the physical connections is helpful for determining detectability.

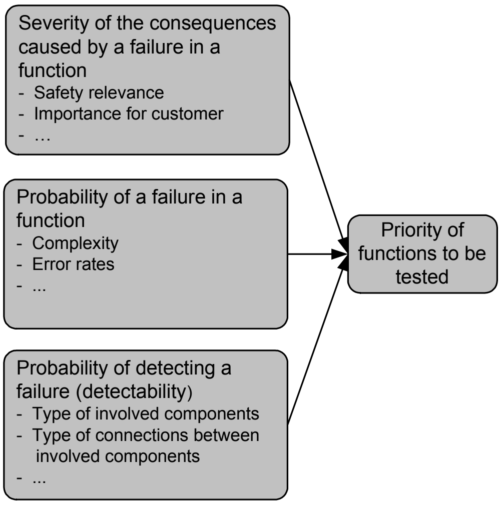

Severity of the consequences caused by a failure in a function
- Safety relevance
- Importance for customer
- …
Probability of a failure in a function
- Complexity
- Error rates
- ...
Probability of detecting a failure (detectability)
- Type of involved components
- Type of connections between involved components
- ...
Priority of functions to be tested

Figure 3: Influencing Factors for Prioritizing Functions.

The combination of values for severity, probability and detectability for the function in question leads to the determination of the priority to be tested.

4.2 Specification of test cases

As we pointed out, there is a need to test functions during manufacturing. In order to execute tests the corresponding specifications of tests have to be derived.

There are several methods to test technical system depending on the objective of the testing. In our context, the objective of testing is to ensure that functions are fulfilled at the end of the manufacturing process. So, out of the existing methods to test, functional testing has to be used and corresponding specifications have to be generated. Functional testing means that a stimulus is created and acts on the tested automobile. Afterwards, the real response is observed and compared to the to-be response. Consequently, information about stimuli, preconditions and to-be responses are needed to specify a test (Figure 4). Preconditions can be further subdivided into conditions that have to be fulfilled at the beginning of a function and conditions that have to be fulfilled throughout the whole execution of a function.

No information concerning the internal design of functions is needed for functional testing. Consequently, functional testing is also known as black-box testing [19].

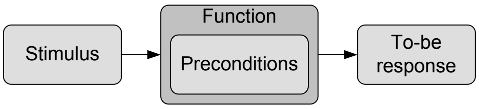


Figure 4: Information for Specification of Function Tests.

In general, testing of functions can be executed manually or automatically or via combination of both. Manual execution means that a person initiates a certain stimulus and checks the response of the automobile. This method is characterized by a high congruence with reality, i.e. functions are tested just like they are used by customers [20]. However, manual testing is time consuming and not always reliable because of the probability of human errors. Automatic execution is achieved without any intervention by a person. Thus, there is sometimes the need to manipulate stimuli, e.g. to simulate that a button was pushed. This leads to a smaller congruence with reality in comparison to manual testing. However, automatic execution of testing is less time consuming and more reliable than manual execution of testing [20]. Because of the specific advantages of executing testing of functions in a manual and automatic way there is a need to support both methods.

There are several ways to generate a specification for a test of a function. First of all, a specification can be created manually with a documentation of functions on the user level. To complete a specification, stimuli, preconditions and the to-be response must be defined (see Figure 4). The specification of tests can be supported by integrating this information into a function-oriented representation. A function would be described by stimuli, preconditions and the to-be response. To derive a specification for a test from such a function-oriented representation, a consistent selection out of this information has to be made. For instance, if a function can be initiated through several stimuli, one stimulus has to be chosen and integrated into the test specification.

Especially the specification for the automatic execution of tests can be simplified by additional technical details concerning the elements shown in Figure 4. This is illustrated by the following examples. Stimuli, preconditions and to-be responses of a function could be detailed by specifying corresponding signals in the logical architecture. This way it is possible to generate a test in which a function is initiated by sending a certain stimulus in the form of a signal. The to-be response would be observed by controlling the corresponding signal. An additional help for specifying tests is also offered by considering information about mapping relations between components and functions. That way it is possible to use characteristic properties of components for defining a functional test. For example, by knowing the current consumption of a component related to the response of a function it is possible to specify a test in which the to-be response is detected by observing the current drain.

4.3 Extraction of compatibility information

The function range of modern vehicles is reached by an interaction of many components. Therefore, it must be ensured that the components contained in a vehicle are compatible to each other as a whole. So the knowledge about the compatibility must be available. With this knowledge it can be ensured over the product life cycle that the components used in an assembly are compatible, for example during manufacturing. Here, knowledge about compatibilities is essential for assuring that the components mounted in an automobile are compatible.

A further example is the case of an error during the usage of an automobile. If one or several faulty components must be exchanged by newer versions or new software has to be brought in, a new configuration arises as a result. So, after the elimination of errors it must be assured that the components are compatible to each other again [21], [12]. This can possibly require further measures, for example the exchange of further, non-defective components or the installation of new software. Again, knowledge about compatibilities is needed.

The derivation of knowledge about compatibilities causes an effort which should be minimized. Compatibility is affected by different factors. Apart from non-functional aspects like mounting constraints or physical characteristics of components also functional criteria and thus the view on the functions realized by the components play an important role. Thus a function-oriented representation is helpful for the determination of compatibilities of components.

In [22] it is shown that two scenarios are important in the field of compatibility: compatibility concerning replacement (replaceability) and interaction compatibility. Replacement compatibility focuses on the versions of a single component. More precisely, replacement compatibility means that different versions of components are exchangeable. Interaction compatibility implies that the components within a configuration cooperate faultlessly and don't exclude each other [22].

There are several ways to determine replaceability concerning functional aspects. A first estimation can be given by comparing the functions related to the versions of the considered component with the help of mapping relations. A version of a component can supposedly be replaced by another version if the former contains all the functions of the latter. Obviously, this method offers only a hint concerning replaceability.

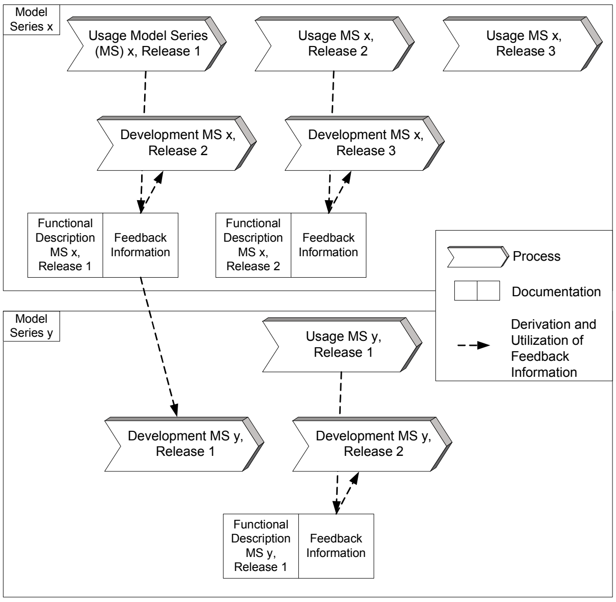
In [21], functional compatibility is subdivided into structural and behavioral compatibility. Considering these two aspects brings more significance to the estimation of replaceability. Structural compatibility looks upon the in- and outputs or signals respectively. So, a documented logical architecture has to be taken into consideration. Here, several levels of abstraction from an abstract signal delivered from a function to the corresponding concrete signal on a bus can be taken into account. In particular, criteria of signals like the type or unit are used. So, an analysis of these criteria of functions related to different versions of the component in question leads to a statement concerning the structural compatibility.

Behavioral compatibility focuses on behavioral aspects which are visible to the environment [21]. So, a statement concerning the behavioral compatibility of different versions of a component can be given by comparing the behavioral aspects of the related functions.

In order to analyze interaction compatibility, similar to the determination of replaceability, structural compatibility has to be considered. To ensure structural compatibility the in- and outputs of functions in a logical architecture within a configuration have to be consistent. In particular, there have to be outgoing inputs for all required inputs. Moreover, the in- and outputs must fit to each other. Again, several levels of abstraction from an abstract signal to the corresponding concrete signal on a bus can be taken into consideration.

4.4 Capture of customer feedback

The number of functions in an automobile increases more and more. This trend is accompanied by a rise in development effort and complexity and by the corresponding disadvantages like the increase in potential error sources. Therefore, it is very important to concentrate on the development of functions that are actually perceived and required by the customer. This way it is possible to avoid an unnecessary increase in the amount of functions [23]. Consequently, feedback concerning functions from the customer's perspective during usage must be gained. This feedback is especial useful as it is based on experience with real automobiles in a common environment. Having this knowledge it is possible to improve functional aspects of new releases [24] or of a new model series as shown in Figure 5. Feedback information related to a certain release of a particular can be gained during the usage and utilized for an improved development of the next release or of another model series.



Model Series x
Usage Model Series (MS) x, Release 1
Usage MS x, Release 2
Usage MS x, Release 3
Development MS x, Release 2
Development MS x, Release 3
Functional Description MS x, Release 1
Feedback Information
Functional Description MS x, Release 2
Feedback Information
Process
Documentation
Derivation and Utilization of Feedback Information
Model Series y
Usage MS y, Release 1
Development MS y, Release 1
Development MS y, Release 2
Functional Description MS y, Release 1
Feedback Information

Figure 5: Integration of Customer Feedback.

In relation to feedback concerning functions the following aspects are relevant: Importance of functions, satisfaction with functions, issues concerning functions and demand for new functions.

To analyze the importance of functions from the customer's perspective two general methods can be used: questioning customers and utilizing of feedback functionalities embedded in a product.

There are several approaches to analyze the importance of product properties by questioning customers. For instance, the ranking method is such an approach. Here, customers are asked for arranging properties of a product in a ranking depending on their importance. Pairwise comparison, constant-sum-scales and rating-scales are further examples of approaches to analyze the importance of product properties by questioning customers [25]. These approaches can also be used to analyze the importance of functions. A documentation of functions on the user level is the basis for such an analysis. Moreover, the values for the importance of functions should be contained within this documentation. However, the approaches mentioned are not sufficient in the case of a great amount of product properties [25]. As we mentioned above, the amount of functions in an automobile is enormous. Consequently, the utilization of the approaches mentioned for analyzing the importance of functions is problematic.

An alternative method to analyze the importance of functions is to use feedback functionalities embedded in a product. Here, the utilization of functions is monitored and analyzed automatically. In particular, the following aspects can be captured: frequency and possibly intensity and duration of function usage [24]. On the basis of this information it is possible to derive a statement concerning the importance of functions. The existence of software and sensors in the product is a precondition for this method [24]. This precondition is fulfilled in modern automobiles. For enabling this method it is important to have technical information concerning function monitoring. For example, information about the stimuli of a function and about the possibility to detect these stimuli is necessary.

Apart from importance of functions the satisfaction with functions is also an important issue related to customer feedback. There are several methods to measure customer satisfaction. So-called objective methods derive a conclusion concerning customer satisfaction on the basis of aggregated indicators like turnover or market share. However, these indicators are influenced by many determining factors apart from customer satisfaction [25]. Moreover, the level the indicators are focusing on is to coarse-grained for a statement concerning customer satisfaction with functions. The so-called subjective methods are more suitable for measuring customer satisfaction. Subjective methods are based on individual customer satisfaction judgements [24]. The customer satisfaction is usually analyzed with the help of customer surveys - either by satisfaction scales or by measuring of the fulfillment of expectations [25]. These methods can also be used to analyze the customer satisfaction with functions. A documentation of functions on the user level is the basis for such an analysis. However, because of the enormous amount of functions the utilization of these methods is problematic.

Another important aspect of customer feedback is the capturing and documentation of issues related to functions. These issues include reports about failures of functions, about handling problems during usage of functions or suggestions for improvements, for example. These issues should be linked to the corresponding functions on the user level. In this way it is possible to identify problems related to functions and to find potentials for improvements.

In [24] and [26] it is stated that customer feedback can also be used to identify new requirements. So, in our context, the demand for new functions should be derived from customer feedback and integrated in the function-oriented representation.

4.5 Update of functions

Production series are developed further also after the beginning of the series production. Thus, new functions are integrated into vehicles during the production period of a production series. The increasing share in electronics and primarily software offers the potential to update a vehicle already produced with the new functions with relatively low effort. In this way it is possible to increase customer satisfaction and customer binding.

To enable the update of new functions it is necessary to know which functions have been added during continued development in comparison to the automobile to be enlarged. Moreover, activities that have to be performed for the update must be identified. Examples of such activities are the application of new software or the exchange of components.

Thus, the procedure to enable an update of functions consists of several steps as shown in Figure 6. In the first step new functions are identified by a comparison of the current functions on the user level with the functions at the time of the production of the relevant automobile. In the second step the activities which must be carried out for the realization of the desired new functions are identified. The identification of the activities can be done either on the basis of expert knowledge or on the basis of detailed information about the mapping relations between functions and software, functions and components and so on. This information has to be documented in the function-oriented representation. For every new function the configuration of the automobile required for the realization is determined. Among other things, this includes the required hardware and software with information about the appropriate variants and versions. Moreover, the required variant coding is to be determined. Thus, information about the corresponding variant coding for a function is helpful.

The required activities for every new function are defined via a comparison of the required configuration and the actual configuration. Regarding software, the following activities can become necessary: an exchange, parameterisation or variant coding. Concerning hardware, a replacement or an addition of hardware might become necessary.

After the execution of the activities for updating the functions desired by the customer it is reasonable to document the new composition of the automobile. This documentation includes the modified function-oriented representation and configuration of the automobile.

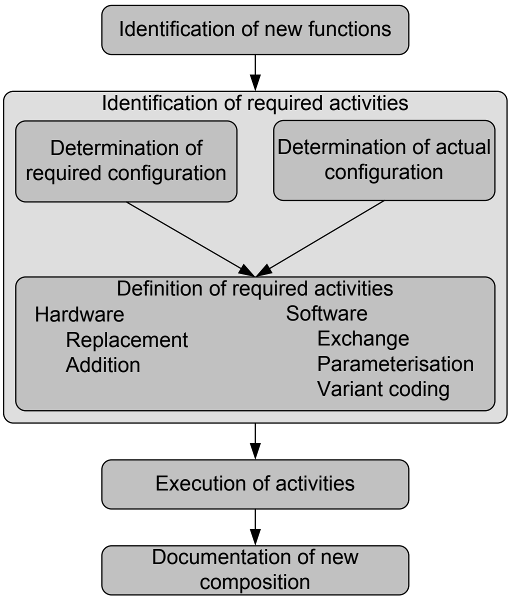

Identification of new functions
Identification of required activities
Determination of required configuration
Determination of actual configuration
Definition of required activities
Hardware
Replacement
Addition
Software
Exchange
Parameterisation
Variant coding
Execution of activities
Documentation of new composition

Figure 6: Procedure for Updating of Functions.

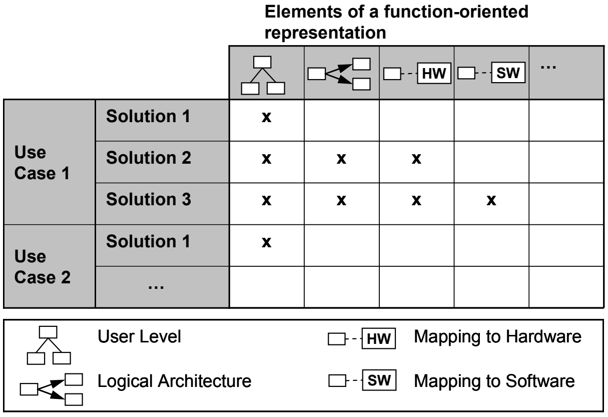
5 SUMMARY AND OUTLOOK

With the increasing complexity of Electric/Electronic systems of modern automobiles the so-called function orientation becomes more and more important. So far, the existing methods in this field are focusing on the development phase. However, function orientation can generate an additional benefit in the late phases of the product life cycle, especially manufacturing process planning, manufacturing and usage of a product. In this paper, several beneficial function oriented use cases during these phases of the product lifecycle were described. Moreover, we presented a possible solution for each use case. An appropriate function-oriented representation is a crucial factor for enabling these solutions of the use cases. However, a concept of a function-oriented representation with consideration of the needs of use cases beyond the development phase does not exist up to now.

Therefore, our goal of further research is to define a function-oriented representation which supports use cases in manufacturing process planning, manufacturing and usage of a product. During our use case analysis we have realized that there are several possible solutions for each use case. This issue has to be considered in the definition of an appropriate function-oriented representation. We will face this challenge by allocating each solution to the corresponding element of the function-oriented representation. Figure 7 shows this approach with a simplified example. The left columns contain the use cases and the corresponding solutions. The top row contains an excerpt of elements of a function-oriented representation as mentioned in section 3. These elements of a function-oriented representation are, among other things, a documentation of functions on the user level, a logical architecture and information about mapping relations. The latter describe which parts, i.e. hardware and software components, contribute to the fulfillment of the related function.

For each solution of a use case, there is a statement about the required elements of the function-oriented representation to support this solution. For instance, solution 1 of use case 1 (e.g. prioritizing functions to be tested on the basis of an estimation of severity, probability and detectability in a subjective manner) is supported by a documented user level. An example for a more demanding solution is the estimation of values for severity, probability and detectability on the basis of detailed information for prioritization of functions to be tested as described in section 4.1. Here, a documentation of functions on the user level and of values for the importance and safety relevance of a function within this documentation would be helpful for the estimation of severity. Considering the function network helps for the estimation of effects of a failure in a function, for example. Information about mapping relations between components and functions is important for an estimation of complexity and therefore among other things, for the determination of probability of a failure in a function. Moreover, documented mapping relations between functions and components and especially the physical connections are helpful for determining detectability as shown in section 4.1.

Thus, it will become visible which elements are required for a certain solution of a use case. With this means it will be possible to configure an optimal function-oriented representation that it suitable for the desired solutions of each use case.



Elements of a function-oriented representation

				□ - - HW	□ - - SW	...
Use Case 1	Solution 1	x				
	Solution 2	x	x	x		
	Solution 3	x	x	x	x	
Use Case 2	Solution 1	x				
	...					

User Level
□ - - HW Mapping to Hardware
Logical Architecture
□ - - SW Mapping to Software

Figure 7: Instrument for Configuring a Function-oriented Representation on the Basis of desired Solutions of Use Cases.

6 REFERENCES

[1] Pahl, G., Beitz, W., Feldhusen, J., Grote, K., 2007, Engineering Design - A Systematic Approach, Springer, Berlin.

[2] Roth, K., 2000, Konstruieren mit Konstruktionskatalogen, Springer, Berlin.

[3] Suh, N.P., 1990, The Principles of Design, Oxford University Press, New York.

[4] Altshuller, G.S., 1984, Creativity as an Exact Science: The Theory of the Solution of Inventive Problems, Gordon and Breach Science Publishers, New York.

[5] Gausemeier, J., Zimmer, D., Donoth, J., Pook, S., Schmidt, A., 2008, Proceeding for the Conceptual Design of Self-Optimizing Mechatronic Systems, Proceedings of the International Design Conference - DESIGN, Dubrovnik, 19-22 May: 1-12.

[6] Stone, R.B., Wood, L., 1999, Development of a functional basis for design, Proceedings of DETC99, 1999 ASME Design Engineering Technical Conferences, Las Vegas, 12- 15 September: 1-15.

[7] Szykman, S., Sriram, R., Racz, J., 1999, The Representation of Function in Computer-Based Design, Proceedings of the 1999 ASME Design Engineering Technical Conferences (11th International Conference on Design Theory and Methodology), Las Vegas, 12-15 September.

[8] Hirtz, J., Stone, R., McAdams, D., Szykman, S., Wood, K., 2002, A functional basis for engineering design - Reconciling and evolving previous efforts, Research in Engineering Design, 2/2002: 65 - 82.

[9] Frauenhofer Institut für Software- und Systemtechnik, 2006, Modellbasierte Systementwicklung in der Automobilindustrie - Das MOSES Projekt, Fraunhofer-Gesellschaft, Berlin.

[10] Ringler, T., Simons, M., Beck, R., 2007, Reifegradssteigerung durch methodischen Architekturentwurf mit dem E/E-Konzeptwerkzeug, 13. Internationaler Kongress Elektronik im Kraftfahrzeug, Baden-Baden, 10-11 October: 199-209.

[11] Lönn, H., Freund, U., Orazio Gurrieri, L., Küster, J., Migge, J., 2004, EAST-EEA - Definition of language for automotive embedded electronic architecture. Report.

[12] Pretschner, A., Broy, M., Kruger, I.H., Stauner, T., 2007, Software Engineering for Automotive Systems: A Roadmap, International Conference on Software Engineering, Minneapolis, 20-26 May: 55-71.

[13] Andres, M., 2006, Die optimale Varianz, brand eins, 1/2006: 65-69.

[14] Katzenbach, A., 2003, Lösungsansätze zur Beherrschung der Komplexität in der Automobilindustrie, 12. AIK-Symposium - Herausforderung Komplexität - Komplexitätsmanagement und IT, Karlsruhe, 17 October.

[15] Broy, M., 2006, Challenges in automotive software engineering, ICSE '06: Proceedings of the 28th international conference on Software engineering, Shanghai, 20-28 May: 33-42.

[16] NN, 1980, Military Standard 1629 A. Procedures for Performing a FMECA.

[17] Pfleeger, S.L., 2000, Risky business: what we have yet to learn about risk management, The Journal of Systems Software, 3/2000: 265-273.

[18] Amland, S., 2000, Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study, The Journal of Systems Software, 3/2000: 287-295.

[19] Wallentowitz, H., Reif, K., 2006, Handbuch Kraftfahrzeugelektronik: Grundlagen, Komponenten, Systeme, Anwendungen, Vieweg, Wiesbaden.

[20] Guddat, U., 2003, Automatisierte Tests von Telematiksystemen im Automobil, Dissertation at the University of Tübingen.

[21] Bechter, M., Blum, M., Dettmering, H., Stützel, B., 2006, Compatibility models, SEAS '06: Proceedings of the 2006 international workshop on Software engineering for automotive systems, Shanghai, 23 May: 5-12.

[22] Stützel, B., Dettmering, H., 2006, Kompatibilitätsorientierte Entwicklung für softwareintensive mechatronische Systeme, 4. Paderborner Workshop Entwurf mechatronischer Systeme, Paderborn, 30-31 March: 281-291.

[23] Burger, F., 2007, Innere Werte - Mit innovativen elektronischen Steuerungseinheiten lassen sich Milliarden verdienen - aber nur, wenn die Industrie hochwertige System-Software entwickelt, McK Wissen, 20/2007: 110-113.

[24] Schulte, S., 2006, Integration von Kundenfeedback in die Produktentwicklung zur Optimierung der Kundenzufriedenheit, Dissertation at the University of Bochum.

[25] Matzler, K., Bailom, F., 2006, Messung von Kundenzufriedenheit, In: Hinterhuber, H. H., Matzler, K. (Ed.): Kundenorientierte Unternehmensführung Kundenorientierung - Kundenzufriedenheit - Kundenbindung, GWV, Wiesbaden: 241 - 270.

[26] Edler, A., 2001, Nutzung von Felddaten in der qualitätsgetriebenen Produktentwicklung und im Service, Dissertation at the University of Berlin.