ELSEVIER
CrossMark
Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 28 (2014) 195 – 203
Procedia Computer Science

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

Development of a modeling language to connect features, functions and components

Axel Grewe[a]*, Marcel Ibe[a], Frank Nehuis[b], Andreas Rausch[a], Thomas Vietor[b]

[a]*Department of Informatics, Software Systems Engineering, Julius-Albert-Straße 4, 38678 Clausthal-Zellerfeld, Germany*
[b]*Institue for Engineering Design, Langer Kamp 8, 38106 Braunschweig, Germany*

Abstract

In order to select the components for a technical system, a detailed description of the systems features is necessary. For this purpose feature-trees are currently preferred. By selecting features from these feature-trees the corresponding components are chosen. Within the development process it is also useful to describe the functions of the technical system as well and to associate these functions with the components. For this functional-networks are used so far, which describe the dependencies between the different functions. But feature-trees and functional-networks are considered separately and not associated with the components within one model. For this reason, the features and functions of a system are usually chosen in advance. So there are two independent ways to choose components for one product. By evolution and reusing of existing components it is possible to create systems with new features or functions. If it would be possible to describe feature, functions and components within one language, proposals for new components could be made. In this paper an approach for describing features, functions and components with one language is introduced.

Keywords: Features; Components; Function Networks; Product Concepts; Development Process

* Corresponding author. Tel.: +49-5323-72-8243 ; fax.: +49 5323-72-99-8243
E-mail address: axel.grewe@tu-clausthal.de

doi:10.1016/j.procs.2014.03.025

1. Introduction

As a first step in the product development process a technical system is described by requirements. However, these requirements are very abstract and have to be refined. So features and functions can be derived. In the next steps the technical system can be composed with a set of components. Every component provides a number of features and functions. So they can be assigned to the corresponding component. So it is possible to select the required components for a system automatically by choosing its desired features and functions.

To describe the features of a system and their dependencies among each other feature trees can be used. Similarly, function networks are used to describe the functions. They enable to describe a system and the component it consists of by describing and selecting features or functions.

Both approaches expect that the components are already known. New components have to be defined before they can be used. It is not possible to create a new component based on desired features or functions. A manual definition of a new component is almost impossible because of the high amount of dependencies between features and the component respectively functions and the component. Since features and functions were described separate it is also hard to support the development of a new component and to embed it into the existing description.

Based on the all existing features and functions a set of them is selected for one specific system. The selected sets of features and functions have to be provided by the components. Thus the components are bound to the specifications of the features and functions (Figure 1). In this paper an approach is introduced that also takes the effects of the components on the features and functions into account.

In practice a large number of dependencies exist between functions and features. If a new component with its specific characteristics and properties should be implemented, new relations between this component and existing features and function will arise. These relations also have to be handled. The actual dependencies between requirements, features, functions and components are schematically shown in Figure 1.

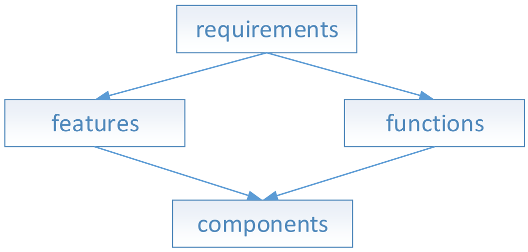

Figure 1: Dependencies between requirements, features, functions and components

It is necessary to model these dependencies. As a result new features, functions and components can be selected in the system development. Furthermore, also changes of the relationships are possible. This makes the system very dynamic. To integrate new features, functions and components tools and languages have to be developed. In this paper an approach is introduced how to model a system. The approach is applied on a car example.

2. Current approach in designing a technical system

As a first step in the development of technical products, the customer wishes as well as the boundary conditions of the enterprises and the restrictions of the product environment will be translated into technical requirements.[7] A transfer into technical requirements is necessary because the customer often does not have the knowledge to formulate his wishes and demands properly. As a result, the product developer may not interpret the requirement correctly in the product development.

As an example in the vehicle development, the customer has the wish to drive an air-conditioned car. This wish, which is formulated qualitatively, must be transferred in a technical requirement the developer can understand as a target.[14] He formulates with reference to several other factors from the product environment that the interior vehicle climate has to be in the interval -5 ° C to +30 ° C. In this way the product developer formulates multiple requirements in the beginning. The resulting set of requirements can be understood as a first concept of the product, which is documented verbally.[15] As a set of requirement, the sum of all requirements to the product is meant. In Figure 2 the current approach in determining requirements and in developing technical systems is shown.

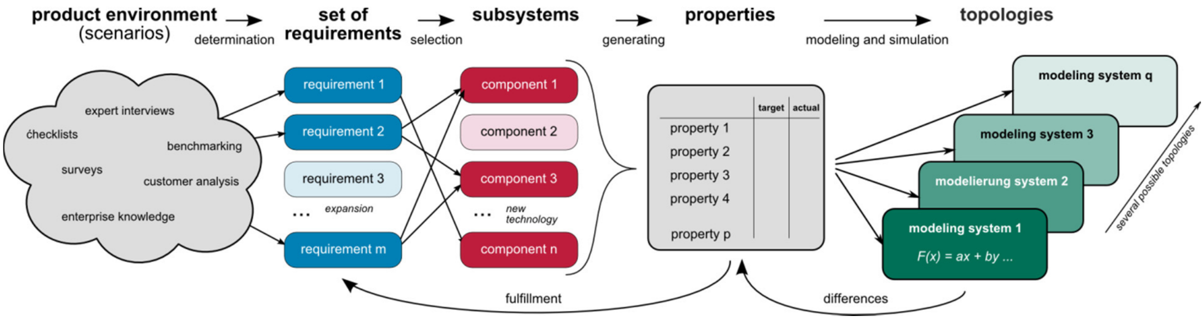

Figure 2: Current approach in designing technical systems

After determining the requirements the subsystems and components are normally selected in current development processes. Subsystems and components can be selected in particular, if a similar product was developed before.[8] A requirement can be fulfilled, for example, through several components. In the example of the interior vehicle climate it is necessary to provide a heater as well as a cooler to fulfil the temperature interval. Also several requirements can be fulfilled by a single component. In general the relationships between requirements and components are much more complex than presented here.

The advantage in using existing knowledge is that the components are already known and they can be investigated in a model (such as PTU-elements to heat the vehicle interior).[9] In addition the effects of the components on the requirements and features are established. However, using already existing subsystems and components restricts the solution space in the development process. In consequence new product generations differ slightly in comparison to older products. Furthermore only a few new product concepts and innovative technologies are used during the product development .[10]

Having selected the components of the technical systems, usually the components and systems characteristics and properties are known. To fulfil the customer wishes and demands especially the product's properties are of interest, because they are often connected to the requirements.[11] defines the product's characteristics and properties as follows:

- **Characteristics** describe the shape, the dimensions, the materials and the surface of a product. Features can be changed directly by the designer.
- **Properties** describe the behaviour of a product (e.g. regarding the function, weight, safety, design, but also with respect to production, testing and costs of the product). Properties usually result from the characteristics and cannot be changed directly by the developer

The parameters of the characteristics and properties usually give some information about the fulfilment of the product requirements, which are set in the beginning. Thus, the fulfilment of the requirements depends on the choice of subsystems and components on the one hand. On the other hand the parameters must be determined carefully.

The characteristics and properties, which are known by selecting the components, are related to each other. To overlook the complexity of the system the relations between the components, the characteristics and the properties must be modelled. This model helps to give first statements about the characteristics and properties of the product. The complexity of such a model is shown as an example in Figure 3.

Again, there may be goal conflicts between the characteristics and properties. Within the modelling process the topology of the product (topology: adjustment of components in three dimensional space) is also determined. If there isn't fixed one specific topology, several topologies must be modelled to find the best one. The goal conflicts within the system become clear by determining the characteristics and properties in simulation and optimization models. On this basis, compromises must be found in order to fulfil the requirement. Alternatively, however, it is also possible to develop different product variants in order to find better compromises between the parameters respectively the requirements. Only the introduction of new technologies enables new product models. With new technologies it can be possible to positively affect or even avoid an existing goal conflict between parameter.

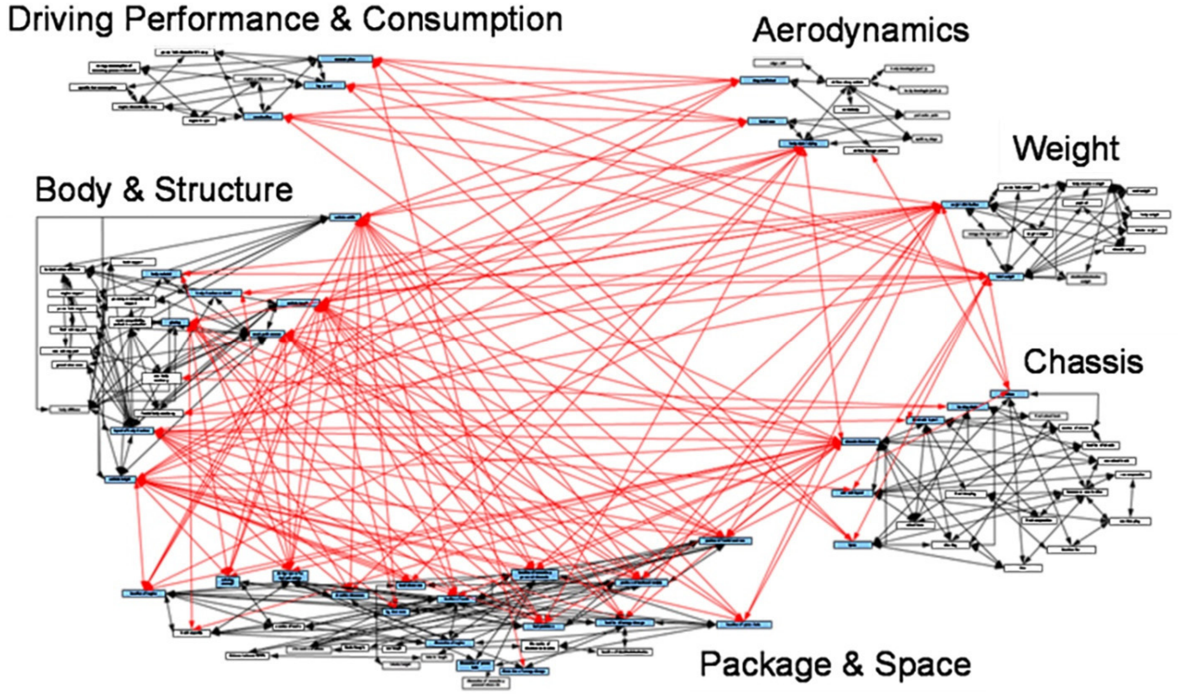

Figure 3: Elements, characteristics and properties of a vehicle and the relationships between them [12]

Feature

Feature trees are used to describe a feature model. The understanding about features and feature trees are taken from .[1] Features describe functional as well as non-functional aspects of a system, which can be observed from the environment of the system e.g. the colour of a car. A detailed description of features, as used in this paper, is described in .[4]

In a feature tree the single features are structured. While a feature is described abstract at the root, the features at the leaves are described more detailed. A distinction is made between two feature types. Firstly, the mandatory feature, and on the other hand the optional feature. Mandatory feature are always included in a system, optional feature may be included. The subfeatures of a feature can be adjusted by logical relationships exactly to the respective system. In Figure 4 an example of a feature tree is shown. As the highest level of abstraction is the car. Every car needs a drive and a drive has a transmission and an engine. For each product line must be chosen between Automatic and Manual. Special equipment is no need for a car. Each component which may be covered by special equipment can be selected individually or as a whole group.

Function

General processes and control systems are denote as function. These functions are usually very complex. Also specific tasks are referred to as function. A function can be separated into many small functions. These single functions can be combined in a relationship model. Relations arise e.g. by transferring parameters or return values. The networking of functions is represented by functional networks. An overview of functional networks can be found in .[5]

A functional network is schematically shown in Figure 5. A function returns a value which is needed by another function. This results in a huge network, which is even confusing for small function networks.

A new approach for the description of functional networks is described in.[6] There an approach is presented where the functions are modelled hierarchically from top level to down level. In the tree the functions are structured with general names. The root is a very abstract description of a function. This method is different to previous approaches, where the basic functions are determined and then the functions on a higher level are set. An unseparated function is referred to herein as main function. An example of this is the measurement of a physical quantity.

For each main function there is a separate child function network. So we get a top level hierarchically element. By such a separation, function networks can be summarized for a better understanding to some top level hierarchically elements. Such a methodology to simplify function networks is presented in.[5] Especially in [6] methods are described which is borrowed from the feature trees. This type of description of functional networks can lighter overlook by people. Through this type of description it offers the possibility feature trees and function networks easier to set in relationship.

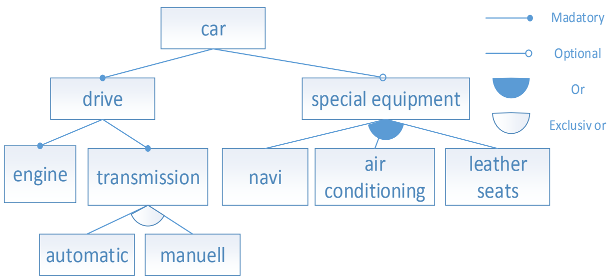

Figure 4: Example of a feature tree

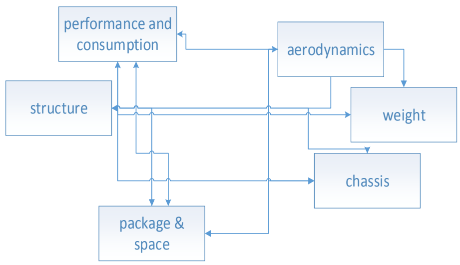

Figure 5: Example of a function network

Component

Components and their properties are described in [1]. The understanding of components and properties is also taken into account in this paper. The different components can be represented in application models. In the development of software product lines the term software architecture is used for application models as well. In this approach a application model represents a classical component architecture as described in [1].

By selecting features components for the application model are chosen. But a different selection of features do not necessarily mean different components. A different connection of components can realize different features. The possibilities of feature selection and component connection is described by the product complexity. Analogue describes the function complexity the possibilities of function selection and component connection. In this way two application models will be created. In view of the independent feature and function selection the chosen components can differ. To create a product it is necessary to merge them.

3. Approach to systematically develop a technical system

Due to the change in paradigm to e.g. electric-powered vehicles in the automotive industry, the demand to develop new vehicle concepts increases. Because of the fundamental change of the power train the designers cannot use already existing knowledge about subsystems and components. For this reason, it is necessary to have a look at the functions of the vehicle. This allows to acquire a basic understanding of the system with its new components. For this the requirements and the basic functions should be connected in a relationship model. Thus, e.g. the requirement to realise a certain climate interval in the interior can be described by the abstract function "ensure interior air conditioning". In this way the needed function in the vehicle will be documented without giving a concrete solution[13]. Similar to the relations between requirements and components in current approaches, one requirement can only be fulfilled by several functions. However, the solution space is significantly larger compared to the current approaches due to the abstract formulated functions. By modelling a relationship model between requirements and functions the solution space can be built. In a specific scenario only a few requirements, which already exist in the relationship model, will be chosen. Afterwards the functions, which have to be realized in the product, can be identified. If a requirement is determined in a scenario, which doesn't exist in the relationship model, it will be easier to connect the requirement with function than to connect the requirement with a component. In this way, the system can be expanded continuously.

New or existing components can be connected to one or several functions. In this way, a component catalogue, which provides different components to realise the selected functions, will be developed. Because the requirements are now connected to the components via the functions it will be easier to identify solutions and to integrate new technologies. The steps to systematically find the functions and suitable topologies and to select the components of the product are shown in Figure 6.

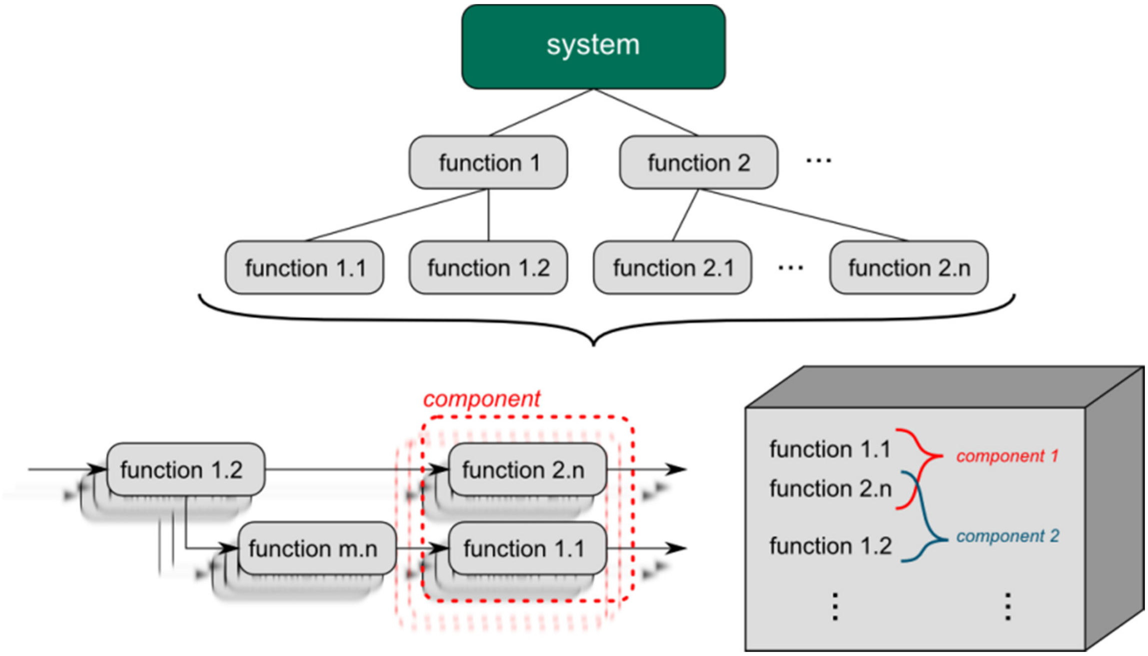

Figure 6: Identification and selection of components and possible topologies based on the functions of a technical system

Using the systematically approach and analysing the relationship model, the product developer can better find the components to fulfil the requirements of the product. If the model provides more solutions, the developer can even select one or more solutions for the further product development process. The integration of new components in the new approach will be better as well, because it is easier to connect a new component to the functions in comparison to connect them directly to the requirements.

4. Implementation of the approach

In this section an approach is introduced how to integrate new components into an existing system. In order to do this additional information flows have to be added. They are shown in Figure 7 (see also Figure 1). The consequences on the features, functions and components are described hereinafter.

For the realization of the systematic approach to designing a technical system, the current models need to be extended. In this section the proposed solutions and approaches are described, which are handled in detail in the future.

Feature

First, the feature *fe* is considered more precisely. The sum of all features is presented in a feature model *MFe*. Every feature can have a set of subfeatures or empty set. They create a subset *Fe* from the entire feature set. The function *FMFe(fe)* returns the subset *Fe* for a feature *fe*. With this function it is possible to determine whether a set contains a specific feature or not. If not, then it has to be added. Therefor it is necessary to know for witch existing feature the new feature is the subfeature.

Function

In the system there are functions *fu*, which describe the functionality of the system. Due to the similar description of the feature trees and function networks, the latter can be treated similarly as the feature trees. The model for the functions is the function network *MFu*. Functions which describe the whole set of subfunctions of a subsystem form a set of functions *Fu* called the main function. A main function always describes a complete functional implementation. To describe a set *Fu*, concrete knowledge about the limits of the subsystem must be known. The function *FMFu(fu)* determines the set *Fu* of subfunctions of a main function *fu*. An abstract representation of the function network is given in Figure 8.

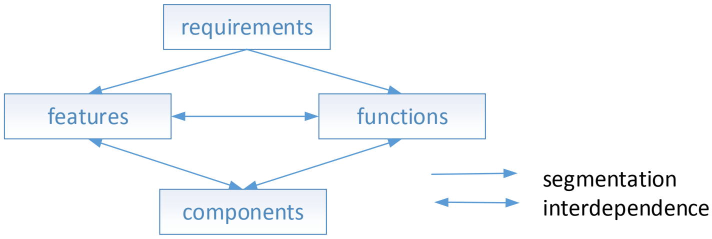

Figure 7: Extended dependencies and information flows

Component

To date features and functions are considered as requirements for the components. Feedback on a feature model or a function network is not considered so far. We want to introduce an approach in which components can be added, which may have an impact on functions and features. Furthermore it shall be possible to create new component by combining existing features and functions. Such a method is necessary to have an opportunity to completely redesign components and evaluate them.

[2] presents a method for a mapping of features on components. Here features and components are represented as sets and put into relation to each other. This approach now needs to be extended to the function networks to integrate the functions. To ensure that all features are used, which are active in the system, a consistency check is performed. Thus, there is already the feedback of components to features but without impact on the feature model. With this approach, the function networks can be connected to the components. Through the use of similar structures for functions, features and components it is easier to handle these three models. The Approach in [2] can now be adopted as follows. Each component has a feature if it is empty. A component has at least one function. So a component is always part of a function or a main function. The component model *MCo* is a set of all components *co* and their relationships.

If a component has a feature that is not contained in *MFe* it has to be included. If this feature requires a function that is not yet contained in the *MFu*, this function has to be added and vice versa. This function must be provided by a component. By combining features and functions the existing components will not always match. In this case, a new component has to be inserted into the system. This component should consist of parts of other components. This assumes that the components are high-level modularity or disassemble by specific rules and can perform together again.

By taking the relations between features and functions into account, an optimal set of components can be chosen for the application model. So the effort for merging two possibly different application models can be avoided.

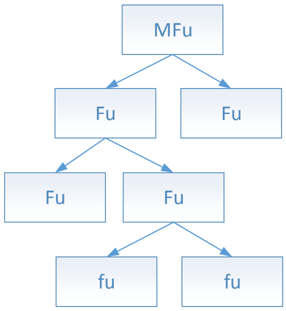

Figure 8: Hierarchically view of function network

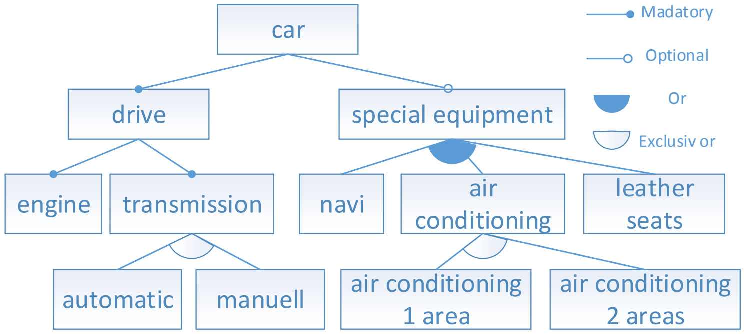

Figure 9: Extended feature tree

Exemplary application

Automatically generated features have to be inserted into the feature tree correctly. To get the necessary information a categorization can be used that is derived from the position of the new component. An example is the different air-conditioning of the driver's compartment and the passenger compartment of a vehicle. Cabs and limousines use a divider between the driver and the passenger. Both areas of the vehicle can be air-conditioned independently. The divider and a second air conditioner have to be added to the component model. By changing the architecture, it is necessary to check the consistency in the feature tree and the function network. There is one change within the function network because another instance of the function air-condition is needed. However, the feature tree has to be adjusted. Since the compartment separation and the independent air condition are new features. The impact on the feature tree can be seen in Figure 9. The derivation for the new air conditioning system is fairly simple. It can be added as a subfeature of the former air conditioning system. The additional divider is already more difficult. One solution could be to relate the divider with the second air conditioner. But it has to be ensured that there are no redundancies or inconsistencies. For new functions there is the difficulty to identify the main function in which the new function has to be integrated or to decide if the new function itself is the main function.

5. Conclusion

Due to increasing customer demands and growing restrictions and boundary conditions, the product requirements are getting more and more complex. This complexity results into more variant diversity in the development of technical products. The paradigm change to energy- and environment-friendly technologies reinforces this process.

If the features and components of a product are known, the designer can use already existing knowledge during the development process. Thereby the relations between the requirements, functions, features and components are known as well. The relations between these elements can be easily modeled in a relationship model. However, to model the system is getting more difficult, if new elements should be integrated into the model.

Concerning this difficulty an approach was presented how to integrate new elements. For this new tools and descriptions have to be developed. By the realization of the approach new elements will be integrated easily into existing product models. The approach will help the designer to better integrate new components and functions in a product.

6. References

1. Trimidad P, Ruiz-Cortés A, Pena J and Benavides D. Mapping Feature Models onto Component Models to Build Dynamic Software Product Lines. Sevilla: Universidad de Sevilla; 2012
2. Buchmann T, Westfechtel B. Mapping feature models onto domain models: ensuring consistency of configured domain models Springer. Verlag Berlin. Heidelberg, 2012

3. Heidenreich F. Towards Systematic Ensuring Well-Formedness of Software Product Lines. TU Dresden
4. Apel S, Kästner C. An Overview of Feature-Oriented Software Development. ETH Zurich
5. Mengi C, Armac I. Functional Variant Modeling for Adaptable Functional Networks. Department of Computer Science 3, Software Engineering RWTH Aachen. University Germany
6. Mengi C, Perez A N, Fuß C. Modllierung variantenreicher Funktionsnetze im Aotomotive Software Engineering. Department of Computer Science 3 Software Engineering RWTH Aachen. University Germany
7. Nehuis F, Ibe M, Stechert C, Vietor T, Rausch A. Clustering Regional-Specific Requirements as a Methodology to Define the Modules of a Car Concept. In: Chakrabarti A, editor. Proceedings of the 22nd CIRP Design Conference (Bangalore, Indien) Bd. Sustainable Product Development; 2013. p. 239 – 249. – ISBN 9781447145066
8. Prinz A, Nehuis, F, Vietor T, Stechert, C. The Effects of Regional Specific Requirements on the Development of Vehicle Concepts. In: Lienkamp, editor. Proceedings of the Conference on Future Automotive Technology (München) Bd. Focus Electro Mobility, 2012. – ISBN 9783658011406
9. Braess HH, Seiffert U. Vieweg Handbuch Kraftfahrzeugtechnik. 5th ed. Wiesbaden: Vieweg Verlag; 2007 – ISBN 3834802220
10. Wallentowitz H, Freialdenhoven A, Olschewski I. Strategien in der Automobilindustrie - Technologietrends und Marktentwicklungen. 3rd ed. Wiesbaden: Vieweg+Teubner Verlag; 2008. – ISBN 9783834807250
11. Weber C. How to Derive Application-Specific Design Methodologies .In: Marjanovic D, Storga M, Pavkovic N, Bojcetic N, editors. Proceedings of the 10th International Design Conference (DESIGN'08 Dubrovnik, Croatia) Bd. Product Development Models and Strategies, 2008. p. 69–80. – ISBN 9789536313891
12. Prinz A. Struktur und Ablaufmodell für das parametrische Entwerfen von Fahrzeugkonzepten. Berlin: Logos Verlag; 2011. – ISBN 9783832528690
13. VDI (Verein Deutscher Ingenieure). Richtlinie 2221 - Systematic approach to the development and design of technical systems and products. Düsseldorf: Beuth Verlag; 1993.
14. Pahl G, Beitz, W, Feldhusen, J, Grote K-H. Konstruktionslehre - Grundlagen erfolgreicher Produktentwicklung - Methoden und Anwendung. 7th ed. Berlin, Heidelberg: Springer-Verlag; 2007. – ISBN 9783540340614
15. Grande, M. 100 Minuten für Anforderungsmanagement: Kompaktes Wissen nicht nur für Projektleiter und Entwickler. Wiesbaden: Springer Fachmedien; 2011. – ISBN 978383488135X