Proceedings of the 19th CIRP Design Conference – Competitive Design, Cranfield University, 30-31 March 2009, pp99

A Criteria-based Measure of Similarity between Product Functionalities

D. P. Politze[1], S. Dierssen[2]
[1]Daimler AG, Research and Development, Böblingen, Germany, daniel.politze@daimler.com
[2]Swiss Federal Institute of Technology, Zürich, Switzerland, dierssen@mavt.ethz.ch

Abstract
Today‘s customers request product functions and not components. A specific, modular description of the product functions and how they are realized becomes widely accepted to track how a product function is realized and to support future development. This will result in an additional, modular product structure from a functional viewpoint that is orthogonal to the physical product structure.
Because the extent of each functional module has to be defined according to some kind of similarity between product functionalities this article introduces a corresponding concept an presents an approach how it can be assessed based on defined criteria.

Keywords:
Product Structuring, Design Units, Function Oriented Product Descriptions, Functional Model, Similarity Criteria, Function Module Driver, Modularization

1 INTRODUCTION

The growth of system complexity in automotive industry is mainly driven by the fact that product functions are more and more realized by the combination of mechanical, electric/electronic, and software components.

Many people have argued that understanding and integrating the users requirements in the development process may be a feasible solution. Subsequently the work of Houdek [1] and Heumesser et al. [2] describe the need and challenges for mature product specifications and raise the question for an adequate description approach. Based on that Allmann [3] suggests an additional abstraction layer for specifying customer-related product functions, which can be compared to the level of abstraction of a user manual.

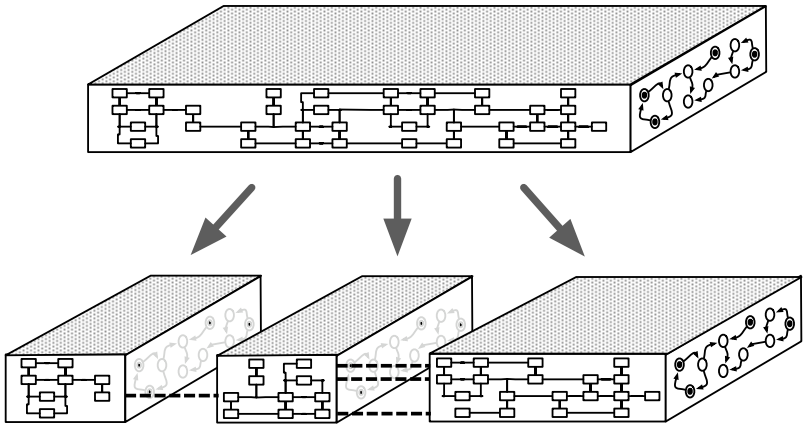
In this paper the understanding of a product function is as a label for „to do something“. The only purpose of a product function is to serve humans or computer systems in such way that they can communicate by referring to the same concept. Every product function is realized by a specific solution. which in turn comprises a specific functionalitythat is seen as the behavior or „what is actually happening“.

It is further assumed that customers demand product functions and that they think of them, when buying a product or complaining about it. Therefore, one can infer that the quality of a product is also perceived by its functionality. That means, in order to provide high quality product functions, it is not sufficient to bring mechanical, electric/electronic and software components to perfection separately. Hence, the component‘s interactions and their contributions to the product functions become important and should be considered very early in the design process. For this reason product functions and the way how they are realized shall be captured and described, resulting in an explicit function oriented product description (FOPD), which allows reuse and improvement of existing descriptions and thus evolves to a mature function oriented product specification that can be used for future design projects. In addition to that many industrial enterprises act as an integrator for development tasks that are made around the world, which in turn requires a detailed description of the intended product functionality [4]. In particular this is the case when the development is made by an external supplier.

Unfortunately in the domain of complex products with high variety - which is the case in the automotive industry - the FOPD becomes very extensive and difficult to use. This is mainly driven by the high number of existing product functions, the increasing share of software and the fact that the description of a product function may differ between product variants. Thus subsequent development steps are confronted with a unmanageable and high complex specification. In this context, modularization is often used as a decomposition technique to define a product structure, consisting of smaller design units or subsystems to master the complexity.

Most of the time this is seen on a par with defining the physical part structure of a product, but this is only one application of modularization. According to [5] the structure depends on the viewpoint and there exist many of them for the same product. Particularly for the development of highly complex mechatronic products the function-oriented structure of a product is seen as equally essential as the part structure [6]. Thus this article agrees to a very abstract definition of modularization that is given as „partition of a system into a set of parts (modules) connected in some way with each other“ [7].

As depicted in figure 1, such structure may be defined by an encapsulation of functionality descriptions that are grouped together according to some kind of similarity , which addresses certain aspects. Unfortunately similarity between the description of product functionalities is not clearly defined. Thus, this article presents a criteria-based definition of similarity that can be used for modularization and thus deriving a function-oriented view on a product. Therefore in the next section a model is given that is appropriate for describing product functionalities in a formal way and enables tool support for creating,



Description of
Product Functionalities (FOPD)
- from observation or specification
- describes technical solutions

Functional Modularization
- based on similarities
- pays respect to certain criteria

(Function-Oriented) Product Structure
- functional design units & their interfaces

Figure 1: The functional modularization task based on the description of product functionalities.

managing and using a FOPD. In section 3 the understanding of similarity is explained and a corresponding measure is defined. Based on that section 4 presents criteria that are appropriate for our kind of modularization task. Section 5 gives a short and simple example on how to use the FOPD model and the similarity measure. Finally section 6 refers to related literature before section 7 concludes.

2 MODELING OF PRODUCT FUNCTIONALITIES

In this section a formal model is given that is suitable to build up a FOPD. This means it is appropriate for describing or specifying functionalities that constitute solutions for product functions of highly variant and highly complex mechatronic products, such as automobiles.The model presented here allows device-centric functional modeling, as described by Pahl and Beitz [8] and further provides a solution for the modeling of activities, sequences, preconditions and variety, as identified in [9].

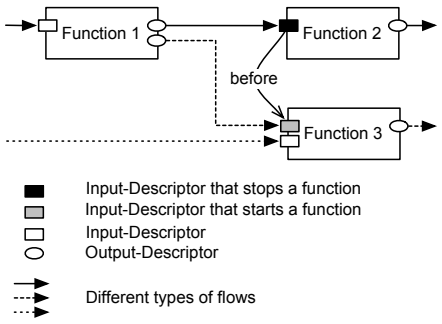


Figure 2: Modeling of functionalities

As the product functionality may be realized by a composition of subfunctions, thus decomposition is also part of the model. More specific, the paradigm of Pahl and Beitz where a hierarchy of functions is working on flows is applied. Based on that a function is assumed to have inputs and outputs, which are defined separately in this model. By describing corresponding input descriptors and output descriptors explicitly, the traditional distinction between function and flow is reinforced and allows the integration of variety aspects. Thus, a product function may be described as to work with different sets of flows, depending on specific variability aspects.

It should also be noted that the provided model is meant to be used recursively, which means that every function-object in the model can represent a solution for another product function.

Figure 2 shows the basic idea of a functional model for product functions as it is described in [9]. It shows different types of flows and functions depicted as lines and flarge white boxes respectively. For every outgoing flow, there exists an output descriptor, represented by a small white ellipse. Similar to that there is an input descriptor for every incoming flow. Depending on the type of associated activity, an input descriptor is either represented as a small coloured box. In this example grey and black are used, which means a flow starts or stops a function respectively. Additionally there is the possibility to define a chronological order between two activities. Thus, from the above figure it can inferred that „function 1“ first stops „function 2“ and then starts „function 3“. At this point it should be clear how „Activities“ and „Sequences“ may be represented with this model.

Furthermore the model allows to define preconditions and to handle variety information, which is done with special attributes for functions and descriptors.

3 DEFINING AND MEASURING SIMILARITY

As stated in the introduction, there is a need for a measure of similarity that can be used for modularization and deriving appropriate design units. In this section a mathematical measure is presented and therefore a notation shall be introduced as shown in figure 3.

Since two objects may be similar regarding different aspects (e.g shape, size or color), this article states that a measure for similarity between two product functionalities has to be dependend on a number of different criteria. Because these criteria may have a different importance a adequate weighting is also needed.

f_i, f_j	product functionalities
$s(f_i, f_j)$	similarity between two product functionalities
n	number of different criteria
w_k	weighting for criterion k
$I_k(f_i, f_j)$	indication of similarity between two product functionalities regarding criterion k
r	a positive integer

Figure 3: Definition of variables

Similarity can then be assessed with respect to a certain aspect and quantified by an indicator function that indicates differences by returning a value between 0 (no difference) and 1 (totally different).

All of the above can be consolidated in a single expression which is a modified version of the weighted

Minkowski metric and delivers a value for similarity. A higher value also means a higher degree of similarity. An additive approach is preferred to a multiplicative approach, because the latter one would consider two functionalities as not similar whenever there is a total difference regarding just a single criterion.

$$s(f_i, f_j) = [\sum_{k=1}^{n} w_k * (1 - I_k(f_i, f_j))^r]^{\frac{1}{r}} \quad (1)$$

Based on the formula given in (1) similarity between two product functionalities is defined as a subjective weighting of correspondences regarding given criteria. In this context it is very important to note that similarity is bounded to a specific context e.g. an application or a point of view, which is expressed by the selection of criteria. Since the application of this article focusses on modularization in order to derive a product structure from a function-oriented point of view, corresponding criteria are presented in the next section.

4 DRIVER FOR SIMILARITIES

In order to determine the right criteria for measuring similarity within this context, a series of expert interviews has been conducted at Daimler. Each of the 15 interviews with experts from the domain of the development department took about two hours and was audiotaped for later analysis. From that, individual aspects have been analyzed and extracted which were then be aggregated to main criteria or main reasons for the function-oriented modularization task. Similar to the findings of Erixon 8 main criteria which are further referred as Function Module Drivers have been identified. Those Drivers comprise the individual aspects for the intended purpose and serve as a basis for the assesment and measuring of similarity. Whereas the Module Drivers of Erixon [10] focus on establishing and justifying a physical part structure, the Function Module Drivers aim at a function-oriented product structure, as distinguished in the introduction of this paper. In order to better recognize the overlaps an analogical naming of the Drivers has been chosen. The Drivers and the individual aspects that have been found in the interview series are given in table 1. In this article a corresponding question for each of the individual aspects is provided in order to improve comprehensiveness and practicability.

In the following the Function Module Drivers and some of the individual aspects are explained in detail. Therefore it is important to note that some of the aspects are rather subjective than objective.

Comparable to Erixon a **carry-over** is referring to functionalities or part of their realization that are re-used in different products, product generations or in several places in one product. There are also similarities in terms of a carry-over when a functionality uses the same components as another.

The **technical evolution** designates functionalities that are bound to a certain technology that might be replaced in future. In this context it is also likely that the old and the new technology will coexist in different products.

Based upon the **technical solution** of a functionality there may exist functionalities that result in the same effect, but are realized in alternative ways or functionalities that are complementary to others. The variety in the realization of a product functionality also provides a reason why such functionalities could be grouped to a function module. Further evidence of being

Driver	Individual Aspects
Carry-over	- Is a functionality also used by other products? - Is a functionality used in different places within one product? - Is a functionality realized by the same actors, sensors, control units?
Technical evolution	- Is a functionality bound to a certain technology? - Is a underlying technology known to be replaced soon? - Is there evidence that two or more underlying technologies coexist?
Technical solution	- Is there an alternative solution / realization for a functionality? - Is a functionality fulfilling the same abstract transformation function? - Is there a complementary functionality (e.g. one that reverses another functionality)?
Common unit	- Is a functionality used in all markets, products and product variants? - Is a functionality used by many other functionalities (interfaces)? - Is a functionality not used at all?
Process	- Is a functionality very important, critical or costly/time-consuming? - Does a functionality need special treatment (i.e. testing)?
User perception	- Is a functionality involved in creating a desired user experience? - Is a functionality executed at the same time (or within a defined interval) with others? - Is a functionality executed permanently or just once a while? - Is a functionality a reaction of / an indicator for another functionality? - Does a functionality lead to a state that enables other functionalities? - Does a functionality belong to a sequence that is desired by a user?
User Intention	- Is a functionality evoked similar to others (e.g. by the same button)? - Is a functionality part of a scenario or use case of a customer? - Is there an alternative functionality that fulfills the users intention?
Company strategy	- Is a functionality provided by an external supplier (e.g. by software)? - Does a functionality require high communication / coordination effort? - Does a functionality need special attention (i.e. importance, piloting)? - Is there a need to describe a functionality in more detail than others? - Is a functionality too complex and its description unmanageable?

Table 1: Function Module Drivers

alternative technical solutions may be found by comparing the function according to Pahl and Beitz [8]. Whenever two functionalities fulfill the same abstract flow-based transformation function they might also be grouped together.

Another driver that is similar to Erixon findings is called **common unit** and refers to functionalities that are similar in that way that they are used commonly or as standard in all product variants. Furthermore, functionalities that may be seen as infrastructure (e.g. providing power) are also adressed by this driver. Very often such functionalities have many dependencies and thus interfaces with others. Besides, functionalities that have no dependencies at all, may also be grouped together, since they have isolation in common.

The Driver labelled as **process** focusses on functionalities that have to be treated special in certain business processes. For example one could group all functionalities together that are ready at a certain point in the assembly process or group all functionalities that may be tested automatically. This Driver also adresses functionalities that are very important, critical or very time consuming in the processes.

Beside the drivers that have some overlap with the findings of Erixon, there exist additional drivers that have not been identified yet and are special for the function-oriented viewpoint.

The **user perception** aims at grouping those functionalities together that are perceived in a certain time intervall or as a causal reaction to something (e.g. when pressing the button for unlocking, the car unlocks, the blinker flashes two times and a sound can be heared). The similarity is in that way that certain functionalities contribute to a desired user experience. Also functionalities that are bounded to the evokation of another functionality shall be grouped together according this Driver, since these functionalities may influence the perception and the behavior of a user.

In addition to that **user intention** refers to functionalities that fulfill a customer‘s goal or consitute a scenario that is demanded by a user. An example for that could be a function module consisting of all functionalities that play a role when a customer wants to put his shopping bags in the trunk. Therefore it should be assessed how functionalities are evoked by a user, because thiy implies similarities regarding the user‘s intention and the desired effect. Hence a grouping can be done according to evokation, scenario or use case descriptions and by finding alternative functionalities that lead to a desired effect

Finally the **company strategy** is an important Driver for deriving a function-oriented product structure. Unfortunately this Driver is very subjective. It allows a grouping of functionalities that consititue special cases, such as novelties, functionalities that need a separate responsibility, functionalities that exist in the context of a piloting or functionalities that are important in any other sense. Also functionalities that are provided by a dedicated external supplier may be determined and grouped according to that Driver. Furthermore this driver decides whether a functionality or an existing grouping shall be split in order to maintain manageability and readability. Finally this Driver also allows grouping functionalities with many dependencies, when the cost for communication and coordinaton becomes too high.

5 EXAMPLE

In this section provides a short example on how to use the formal descriptions, the formula and the drivers to determine similarity. To keep the example simple it is restricted to only three very simple functionalities *F1*, *F2* and *F3* and only take the drivers common unit and user perception into account. Regarding the formula given in (1) the value for *r* has been chosen as 1. Furthermore a weighting of 1 and the existence of only two products *P1* and *P2* are assumed.

Since *F2* is only used in half of the product as is *F1*, it is easy to see that the indicator function regarding common unit *I1* delivers a value of *0.5*. On the other hand *F2* is stopped by *F1* which means that a user will perceive it in a certain time intervall. Thus there is no big difference and a value of *0.0* for *I2* may be assumed. In the same way the values for *F1* and *F3* may be assessed, resulting in no difference regarding the commonality but a total difference regarding perception.

Finally the similarity is calculated according to the formula given in section 4 and the result shows that *F1* is more similar to *F2* than to *F3*.

6 RELATED WORK

In the past decades, functions and the modeling of functions have become a part of some well-known design methodologies and in order to support a common understanding of functions between all stakeholders in the design process formal function representations and vocabularies (sometimes called ontologies) have been defined.

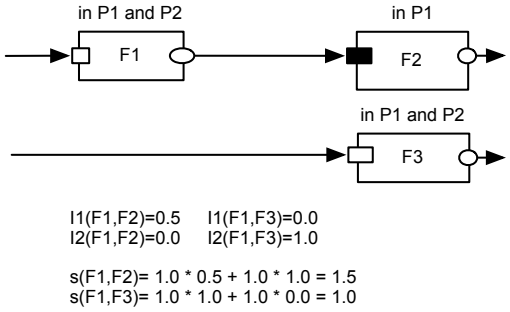


Figure 4: Example

The work of [11] gives a very good and detailed overview over functional modeling and they distinguish different types of such ontologies. Whereas a device ontology describes a system to be composed of black box modules, a functional concept ontology aims at modeling the functionality of a system from the viewpoint of human such as the work of Chadrasekaran and Josephson [12], Umeda and Tomiyama [13] or Gero [14]. While integrating the user in the modeling of functions already points in the right direction there is additional need on describing functionalities and thus technical solutions for the development of high complex mechatronic systems in order to enhance re-use and improve quality [2],[3]. The approach that is described in section 2 is also pointing in that direction.

In the same way as functional modeling the understanding of modularization has become very ambigious in the past years. A collection of definitions and a very good review on that topic may be found in [15] which also agrees on different types of the modularization task e.g. for design, for production or for use.

In this context, the idea of having criteria for modularization has been conducted very succesful by Erixon and his Module Drivers that were found in case studies [16], [17].

A mathematical approach for measuring similarities or commonalities has been presented by Kota [18]. His Product Line Commonality Index is an objective measure for sharing parts across product variants, but not applicable for product functionalities.

7 CONCLUSION

This article refers to a model for describing product functionalities in a formal way. Based upon that a definition of similarity is developed and expressed as a mathematical and criteria-based measure of similarity between two product functionalities. This measure shall be used for the development of highly complex mechatronic products with great variety.

Then objective as well as subjective aspects are presented that have been collected at Daimler and were subsumed in question form by Function Module Drivers. Those aspects and drivers help define an application context of the similarity measure aiming at the derivation a function-oriented product structure.

Finally a short example has been given. This example shows how the formal model, the formula and the criteria may be used to determine a value for similarity.

Since five of the provided eight criteria are comparable to Erixons Module Drivers, this research also approves the validity of his findings in parts.

Future work will focus on transferring the Function Module Drivers into industry application and on ways how the similarity measure may be used to derive a function-oriented product structure. Therefore we currently analyze how the data in the formal model gives answer to the questions that correspond to the individual aspects and thus to the drivers.

8 REFERENCES

[1] Houdek F, 2003, Requirements Engineering Erfahrungen in Projekten der Automobilindustrie, Softwaretechnik-Trends, 23(1).

[2] Heumesser N and Houdek F, 2003, Towards systematic recycling of systems requirements, Proceedings of 25th International Conference on Software Engineering. 512–519.

[3] Allmann C, 2007, Anforderungen auf Kundenfunktionsebene in der Automobilindustrie, SE 2007 – die Konferenz rund um Softwaretechnik, Hamburg.

[4] Eversheim W, 1998, Organisation in der Produktionstechnik: Konstruktion, Band 2, 3rd edition, Springer Verlag.

[5] Andreasen MM, Hansen CT and Mortensen NH, 1995, On Structure and Structuring, Workshop Fertigungsgerechtes Konstruieren, Erlangen, Germany.

[6] Eversheim W, Schernikau J and Goeman D, 1996, Module und Systeme: Die Kunst liegt in der Strukturierung. VDI-Z 138 (1996), Nr. 11/12 – November/Dezember.

[7] Stevens WP, Myers GJ, and Constantine LL, 1974, Structured design. IBM Syst. J., Vol. 13:115 – 139.

[8] Pahl G and Beitz W, 2007, Konstruktionslehre: Methoden und Anwendung. Springer, 7th edition.

[9] Politze DP and Dierssen S, 2008, A functional model for the function oriented description of customer-related functions of high variant products. Proceedings of NordDesign'08, Tallinn, Estonia, August, (to appear).

[10] Erixon G, 1998, Modular Function Deployment – A Method for Product Modularisation. Doctoral Thesis, Royal Institute of Technology, KTH, Stockholm.

[11] Erden MS, Komoto H, Van Beek TJ, D'Amelio V, Echavarria E and Tomiyama T, 2008, A review of function modeling: Approaches and applications. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 22:147–169.

[12] Chandrasekaran B and Josephson JR, 2000, Function in device representation. Engineering with Computers,16:162–177.

[13] Umeda Y and Tomiyama T., 1995, FBS modeling: modeling scheme of function for conceptual design. In Proc. Working Papers of the 9th Int. Workshop on Qualitative Reasoning About Physical Systems, 271-278, Amsterdam.

[14] Gero JS, 1990, Design prototypes: a knowledge representation schema for design. AI Magazine 11(4):26-36.

[15] Salvador F, 2007, Toward a product system modularity construct: Literature review and reconceptualization. IEEE Transactions on Engineering Management, 54(2):219–240.

[16] Erixon G, 1996, Design for Modularity. Design for X - Concurrent Engineering Imperatives. Chapman & Hall. Edited by G. Huang.

[17] Östgren B, 1994, Modularization of the Product gives Effects in the Entire Production. Lic. Thesis, The Royal Institute of Technology. Stockholm, Sweden.

[18] Kota S, Sethuraman K and Miller R, 2000, A metric for evaluating design commonalities in product families, Journal of Mechanical Design, 122:403-410