INTERNATIONAL DESIGN CONFERENCE - DESIGN 2004
Dubrovnik, May 18 - 21, 2004.

DESIGN 2004

TOPOLOGY OF MODULAR KNOWLEDGE STRUCTURES IN PRODUCT DEVELOPMENT

S. Weiss, B. Berger and H. Birkhofer

Keywords: modularization, product development, design knowledge

1. Introduction

Product development knowledge covers a broad spectrum, including design theory, design methods and solutions. This knowledge exists in literature and in practice in different occurrences with regard to preparation and presentation. Different types of users acquire this knowledge with regard to different application scenarios. Each scenario sets up specific requirements concerning the contents and their presentation. Therefore, product development knowledge occurs in many forms and must be adapted to the corresponding situation. In this case, it is not enough that knowledge is prepared and stored correspondingly; it must also be available in a situative way. In other words, the knowledge must be made available to potential users by storing it in a database according to quality, kind, volume and presentation to be retrieved when needed. Existence and location are important: a comprehensive database is useless if it is not possible to find specific contents [Weiss et al. 2003]. In order to master the content-related variety, a cooperation of different authors is advisable. It is a basic idea that authors from different regions contribute knowledge in a teaching, learning and application system. This system must be able to make stored knowledge available which corresponds to a defined application scenario.

2. State of the Art

For this purpose the ***pinn****gate*-project [Weiss et al. 2003] was established. This project is the successor of the *thekey*-approach [Albers et al. 2001] [Birkhofer et al. 2002]. The main aim of the ***pinn****gate*-project is the appropriation of product development knowledge for different user groups. Here, knowledge is stored in a central modular database. This structure is based on a modularization approach [Berger et al. 2002] [Birkhofer et al. 2002]. The 3-Level-Model describes the knowledge structure. This model is based on so-called elements. These provide text and picture fragments with no semantic character of their own. From elements modules and containers are formed. These have their own semantic character and represent the actual knowledge. In this approach, custom-designed containers are formed from elements and modules. Examples of modules are tables, explanations and surveys. Examples of containers are form sheets, chapters and method descriptions. Custom-designed containers were prepared for a specific use corresponding to a specific application scenario. Elements, modules and containers are provided with suitable meta-data, such as keywords, topics and other meta-data. A net-like structure exists between elements, modules and containers representing their relationships. This modular structure is used particularly to make knowledge available to different types of users in different scenarios. This situative approach provides process and aim causality in the ***pinn****gate*-project [Weiss et al. 2003].

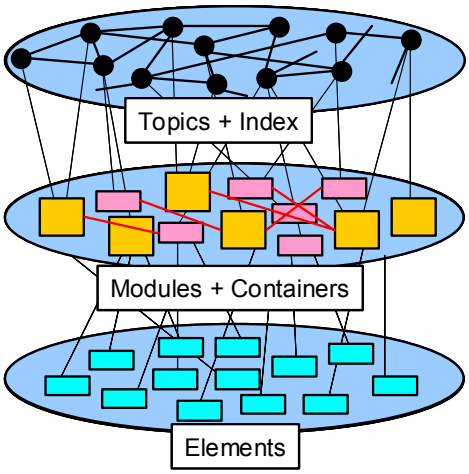

Figure 1. 3-Level-Model

The model (Figure 1) was developed and evaluated from work with product development contents. Therefore, its suitability for other fields can not be deduced from it directly. The subsequent remarks have to a large extent a generally performing character. However, as regards content, this work is a result found by the work with product development contents. But in the authors view, this approach is suitable for different purposes. For example, this approach is actually used in the filed machine elements to describe couplings.

3. Deficits

The 3-Level-Model contains a semantic net on the highest tier. This allows the intuitive access to the available knowledge for users. The aim of the ***pinn****gate*-project is to support different types of users in a situative way. Therefore, intuitive access is not sufficient. Particularly, the semantic net assembled over modules and containers can not be of a static nature since its structure and the meaning of its links depend upon the application scenario. Consequently, a high-powered semantic net must be provided. In this paper the model is extended and formalized. The consequences resulting from modularization are discussed. The effort of building and storing links is discussed, as well.

4. The Formalized Model

In order to allow a computer implementation, a formalization is necessary. Element-, module-, container- and term-objects must also be considered. Different relationships can exist between elements, modules and containers: firstly, those to constitute contents and secondly, those to represent similarity. Content-related relationships define the connections between elements, modules and containers in order to constitute a semantic content based on elements. Similarity relationships represent alternative modules and containers with respect to a considered object. During the formation of modules and containers, the attention of sequence is also necessary. Modules and containers have a semantic character. Consequently, the content-related sequence has a decesive importance. Moreover, the concepts of the semantic net can be equally in relation with each other.

In the following sections, the term EMC-object stands for elements, modules or containers. There are different possibilities to store these structures. This approach uses lists and shows that potential matrices are inefficient for this. The sequence of objects is represented by the sequential nature of lists. Normally, there are a terms in the semantic net and b EMC-objects describing a specific content-related separated field in a suitable way. The terms might be numbered from $1...a$ and the EMC-objects from $1...b$. Usually, there are no duplicates of the elements and every element is used at least once. Dependent relationships are noted down in predecessor lists and successor lists. These contain the ID-number of the objects used in each case. For every EMC-object i a parameter δ_i exists that characterizes whether the object is an element, module or container:

$$\delta_i = \begin{cases} 0 & \text{if Object i is Element} \\ 1 & \text{if Object i is Module} \\ 2 & \text{if Object i is Container} \end{cases}$$

So a first set of lists is required to define content-related EMC-relationships (V, N). A second set of lists is required to define the relationships between EMC-objects (v, n). Both predecessor lists and successor lists are managed. These represent an expedient input dependent on algorithms. To derive the contents of a container, for example, it is suitable to follow the predecessor list down to element level. To determine the frequency of use of an element, the entries in the successor list of this element may be considered. Another way is to search in predecessor lists of all modules and containers. However, the redundant lists are proven to be more advisable. The formalized model is represented in figure 2.

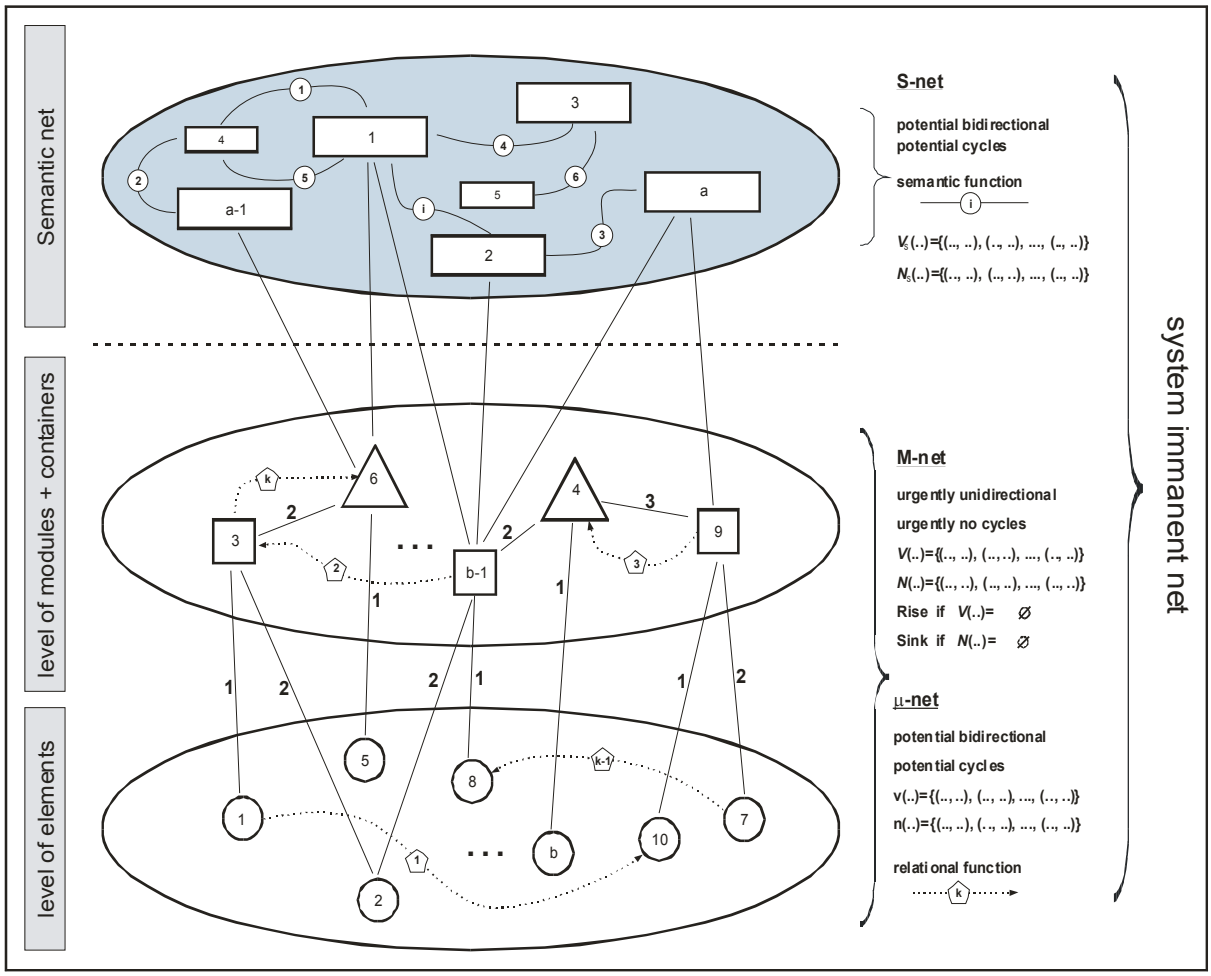

Figure 2. System immanent net

By definition, every element is used at least once. It follows that every successor list must contain at least one entry.

$$N(i) \neq \phi \quad \forall i = 1 \ldots b \text{ with } \delta_i = 0$$

Elements are the lowest units in the modular structure (rise). Thus, the predecessor list of every element must be empty.

$$V(i) = \phi \quad \forall i = 1 \ldots b \text{ with } \delta_i = 0.$$

The successor lists of containers must similarly be empty since these are the highest units of the modular structure (sink).

$$N(i) = \phi \quad \forall i = 1 \ldots b \text{ with } \delta_i = 2$$
$$V(i) \neq \phi \quad \forall i = 1 \ldots b \text{ with } \delta_i = 2$$

On the other hand, it is possible, that well-defined modules are not used in containers. Therefore, the successor list of a module need not contain an entry.

$$V(i) \neq \phi \quad \forall i = 1 \ldots b \text{ with } \delta_i = 1$$

All relationships between EMC-objects constitute contents form the so called M-net. This is the network resulting from modularization. The relationships of the M-net are necessarily unidirectional since modules are based on elements and containers are based on elements and /or modules, but not vice versa. They have no cycles since recursions make no sense during content-related work (no container contains itself).
Between two EMC-objects relationships can exist which describe dependencies. These relationships exist mainly between modules and containers since these have their own semantic character. However, elements are not excluded from this by definition. All dependent relationships form the so called μ-net. This net is potentially bidirectional and potentially cycle-afflicted. Examples of relational functions may be: is similar to, is the long version of.
Figure 3 (text concerning overall function and function structure in Figure 3 according to [Pahl, Beitz 1984]) shows two modules and one container built up from six elements (headline, text parts, pictures).

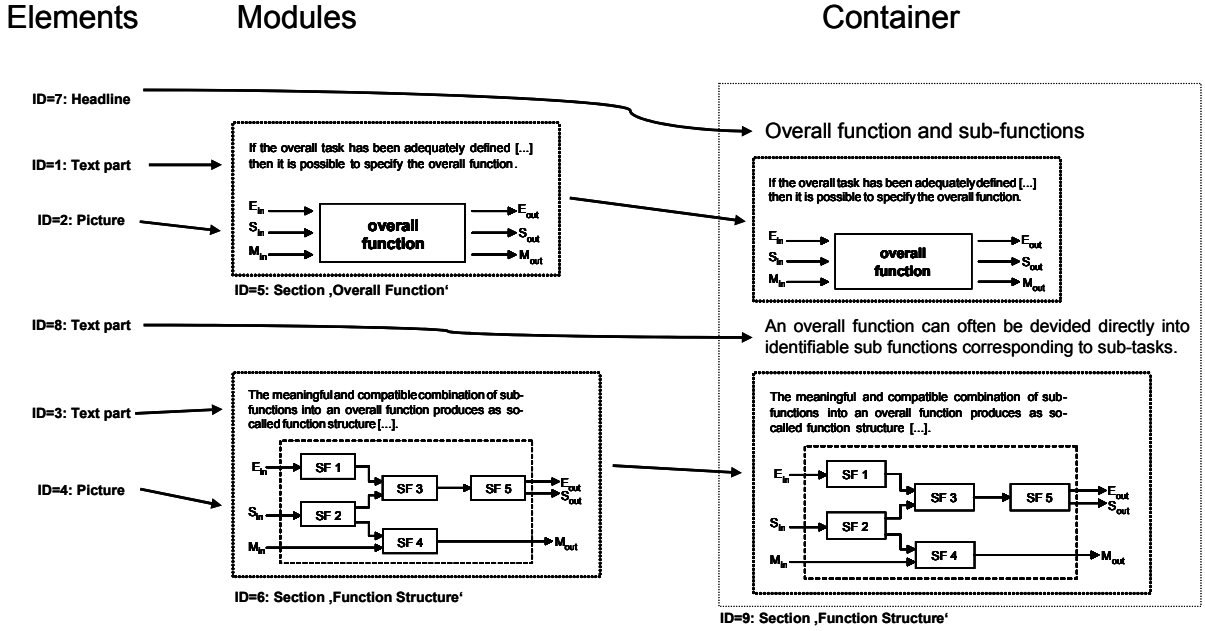

Figure 3. Modules and Container built up on element base

Every EMC-object carries an ID. Two Elements (ID=1+2) form a module (ID=5). So the module consists of a text part and a picture. Module ID=6 is defined analogously. The Container ID=9 is defined by combination of two additional elements (ID=7+8) and the two modules explained before.

So predefined modules were used to derive an new document. Using the model in Figure 2 one can write down (M-net definition):

$$v(1) = v(2) = v(3) = v(4) = v(7) = v(8) = n(9) = \phi$$
$$n(1) = n(2) = \{5\}$$
$$n(3) = n(4) = \{6\}$$
$$v(5) = \{1, 2\}$$
$$v(6) = \{3, 4\}$$
$$n(5) = n(6) = n(7) = n(8) = \{9\}$$
$$v(9) = \{7, 5, 8, 6\}$$
$$\delta^T = [0,0,0,0,1,1,0,0,2]$$

Figure 4 shows a simple S-net. The term 'function structure' and 'overall function' are linked to the modules ID=5 and ID=6.

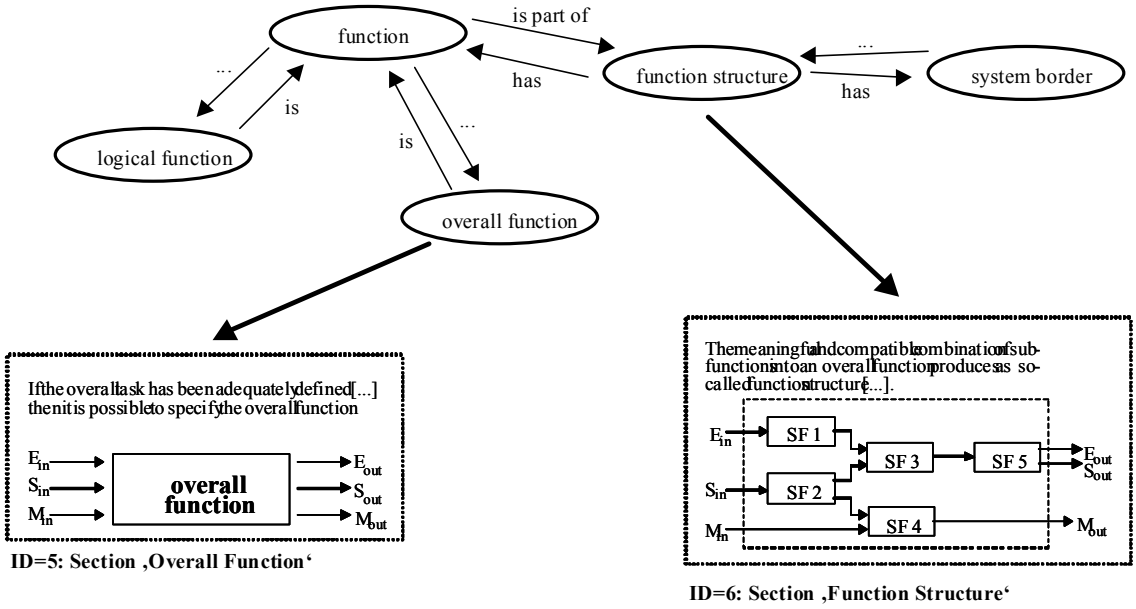

Figure 4. S-net links M-net

The memorization of the semantic net is realized by lists too. But it is necessary to store an additional number representing a semantic function. It is not sufficient to use simple individual links, because most links and their meaning vary depending on the application scenario or situation. In the ***pinn****gate*-project knowledge is provided according to the application scenario. For this, it is necessary that the semantic net corresponds to the requirements of potential types of users in their specific situation. EMC-objects must be represented by the S-net so that the respective user understands the terms and connections used. For example, a potential user may make either a very simple or a very specific search for an entry. Consequently, the semantic function is established. The semantic function constitutes a network that depends on the potential application scenario. A semantic function σ_j is a vector which contains terms in its different dimensions. Each dimension stands for a defined application scenario. One would be able to assemble two dimensions, for example, to define scenarios such as "industrial project" or "lecture". If a cell of the vector σ_j is empty, no link exists in this dimension.

$$\sigma_j = \begin{bmatrix} meaning\ if\ \ scenario\ is\ "industrial\ \ project" \\ meaning\ if\ \ scenario\ is\ "lecture" \end{bmatrix}$$

Therefore, depending on the scenario, links in the semantic net are available or they are not. They vary in their meaning. The database is updated with lists (V_s, N_s). These lists take the number of the semantic function after every object number. So every list entry consists of an object number and the number of the semantic function.

$$V_s(..) = \{(object\ number, j),(..,..),..\}$$
$$N_s(..) = \{(object\ number, j),(..,..),..\}$$

The nodes of the semantic net are coupled with the EMC-objects of the module and container level (figure 4). Each of the a term-objects can assign an arbitrary number to modules and containers. These links also satisfy situational aspects. Modules and containers can not be accessed via S-net without corresponding links or semantic functions defined in the S-net. But this shall not be further discussed here. M-net, S-net and μ-net comprise the "system immanent net".

Net processing algorithms must be interpreted on list structures. Matrix stores did not prove suitable. Assuming one saves the M-net as follows (Figure 5):

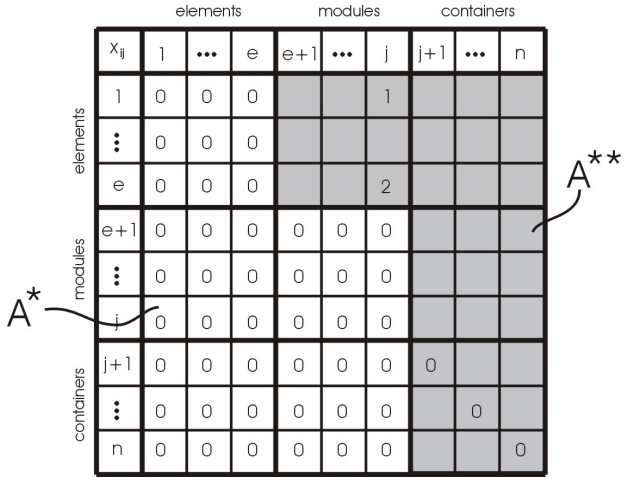

Figure 5. Matrix storage of the M-net

There are *n=b* EMC-objects composed of e elements, m=j-e modules and c=n-j containers. Then, one can write down a number other than zero for every element, module or container on the columns of the matrix if the corresponding EMC-object of the horizontal lines is available in the target object. The number stands for the position. Zero implies no relation. In figure 5, module j contains element 1 at position 1 and element e at position 2. Five combinations are not possible because of unidirectionality: elements (rise) contain no elements, modules or containers and modules contain no modules or containers. Assuming A^* stands for unused fields and A^{**} for used fields. The utilisation η can be thus defined as:

$$\eta = \frac{A^{**}}{A^* + A^{**}} \quad with \quad \begin{aligned} A^* &= e \cdot (e + m + c) + m \cdot (m + c) \\ A^{**} &= e \cdot m + c \cdot (e + m + c) \end{aligned}$$

A result of a post analysis of modularized contents shows that 4-5 elements per module are used on average; analogously 2 elements and 2-3 modules per container. By using ($m = 4.5 \cdot e \quad and \quad c = 2 \cdot e + 2.5 \cdot m$) a utilization of η=69% results. In other words, more than a

quarter of potential storage is wasted. This does not seem efficient. Particularly, algorithms processing matrices or algorithms processing lists - it does not matter for the programmer. So we have chosen the list structure.

5. Conclusions

In theory, modularization requires to work at the module level at least. This is true because elements have no semantic character. Therefore modules must be derived from elements. This means that the number of links created has to be equal to or greater than the number of defined elements at least. The analysis mentioned above results in 4-5 elements per module. By using $m = 4.5 \cdot e \quad and \quad c = 2 \cdot e + 2.5 \cdot m$ 13 elements per container are calculated. This corresponds to 13 links. Modules and containers have to be derived from elements. The effort increases with the number of modules and containers.

However, a much greater effort is necessary to generate the semantic net (S-net). In the simplest case there is only one scenario. So the S-net is reduced to an ordinary semantic net. At an adequate description depth at least $\beta = a$ term-objects are necessary. Since S-net links are potentially bidirectional, $\beta \cdot (\beta - 1)$ links are possible. The number of links ξ to be set between modules and containers has to be added to this value. These links are mostly required because no term-object is suitable without a connection on the level of modules and containers.

Every term-object must be linked with at least one other term-object. Otherwise, the term could not be reached by intuitive navigation on the S-net. So $(\beta - 1)$ links have to be set at the least. Moreover, every term-object has to be linked at least once with a module or container (β). Therefore, $2 \cdot \beta - 1$ connections are required in the minimum.

There is an asymmetry concerning the effort to build up M-net and S-net. This can be explained by a theoretical experiment. For each EMC-object added, meta-data have to be defined. For this, the effort is independent of object type (element, module or container) and approximately constant for every additional object. However, the amount of potential links increases exponentially as the number of concepts, modules or containers increases. Thus, the effort concerning the S-net exceeds the effort concerning the M-net.

For the construction of the M-net relevant elements must be defined. In addition, at least 13 links must be defined per container in the average. The maximum can not be determined because it depends on the desired extent of content. The effort to define the S-net encloses $2 \cdot \beta - 1$ links at minimum and $\beta \cdot (\beta - 1) + \xi$ at maximum. The S-net consists mainly of technical terms. Moreover, several scenarios are taken into account (dimension of σ_j >1). Therefore, this net is much more extensive than the minimum approximation. So the main effort has to be concentrated on the S-net sector.

6. Summary and Outlook

This paper describes a model which was developed to modularize knowledge in product development. This model is used in the ***pinn****gate*-project which supports different user types in various application scenarios with knowledge. In addition, the semantic access of the 3-Level-Model has been extended by semantic functions. The model has also been formalized and the net structures have been analyzed. The main overhead arises from the work on the S-net. The automation of this process would be of considerable benefit for authors and configurators defining elements, modules and containers.

References

Albers, A., Birkhofer, H., Lindemann, U., Meier, M., "Product Development As a Structured and Interactive Network of Knowledge", in: Proceedings of the International Conference on Engineering Design, ICED 2001, Bd. 3, Glasgow, 2001.

Berger,B., Birkhofer, H., „Modularization of design knowledge as a basis for a high quality competence pool in product development", in: Proceedings of DETC 2002, Hrsg., ASME 2002 Design Engineering Technical Conferences, Montreal, 2002.

Birkhofer, H., Berger, B., Walter, S., „Modularisation of Knowledge – A New Approach in the Field of Product Innovation", International Design Conference – Design 2002, Dubrovnik, 2002.
Pahl, G., Beitz, W., "Engineering Design", London, 1984.
Reimer, U., "Einführung in die Wissensrepräsentation: netzartige udn schema-basierte Repräsentationsformate", Stuttgart, Teubner, 1991.
Weiss, S., Berger, B., Jänsch, J., Birkhofer, H., „COSECO (Context-Sensitive-Connector) – A Logical Component For a User- and Usage-Related Dosage of Knowledge", Proceedings of the International Conference on Engineering Design, ICED 2003, Stockholm, Sweden.

Dipl.-Wirtsch.-Ing. Sascha Weiß
Departement of product development and machine elements
Darmstadt University of Technology
Magdalenenstr. 4, 64289 Darmstadt, Germany
Telephone: +49 6151 16 2660, Telefax: +49 6151 16 3355
E-mail: weiss@pmd.tu-darmstadt.de