

Static Product Structures: An Industrial Standard on the Wane

Stefan Kehl¹, Carsten Hesselmann¹, Patrick D. Stiefel², and Jörg P. Müller¹

Department of Informatics, Clausthal University of Technology,
Clausthal-Zellerfeld, Germany,
{stefan.kehl, carsten.hesselmann, joerg.mueller}@tu-clausthal.de¹
Volkswagen Group, Wolfsburg, Germany, patrick.stiefel@volkswagen.de²

Abstract. This paper aims at closing the gap between early phases (e.g. design) and later phases (e.g. procurement or production) of the Product Development Process (PDP) by proposing a Virtual Product Model (VPM) as a *collection* of individual components (VPMCs) without the need for a static structure. Based on an analysis of the requirements on product development in the automotive industry, the main problems we observe are limited transparency, limited continuity, and limited reusability throughout different phases of the PDP. Virtual Product Model Components (VPMCs) can be used in *different products* and allow the reflection of *changes throughout the PDP* as well as the derivation of *domain-specific views* on the overall product at runtime. We illustrate these concepts by use case scenarios derived from an analysis of automotive product development practices.

1 Introduction

Over the past years, the complexity of industrial products has been increasing due to rising expectations from customers and the resulting higher variety. E.g. in the automotive industry the number of possible variants is skyrocketing and has reached a level at which “the number of theoretical possible Vehicle Variants is higher than the number of sold Cars” [9]. This trend is often referred to as *mass customization*. According to Tseng and Jiao [18], it is about “producing goods and services to meet individual customer’s needs with near mass production efficiency”. With this “explosion of potential offering variety” [5] many issues arise such as (i) the traceability of parts (and their geometric representation) used in multiple products, (ii) a comprehensive way to manage changes on components, and (iii) an effective structuring of product models which supports reusability, continuity and transparency [13] and need to be addressed in order to preserve success on the market and meet both new technological and customer requirements. Some practitioners propose concepts, like the modularization of complex products [7,12], to share common parts in multiple products and product lines regarding their individual components [2].

In practice static product structures, such as Bill of Materials (BOMs) (see Sect. 2), are still heavily used as the data backbone for Product Lifecycle Management (PLM)-systems [1]. E.g. in [17], Tekin et al. described an approach to use multiple static product structures

for different stages of the PDP. Structures which are used later during the PDP (so called *As Built Bill of Materials (ABOMs)*) rely on previous ones like Manufacturing Bill of Materials (MBOMs) and are reconciled with Engineering Bill of Materials (EBOMs). Chatras et al. argue in [5] that managing complex product models with BOMs is no longer an adequate solution because of the current scale of diversity.

Besides considering the aspect of *how to structure a virtual product model* much work focuses on the engineering perspective of the product development process [7,10,12]. Within the MOKA product model [15] constraints which represent design restrictions are described. Among other things, those constraints can be used to model product choices and to define the order in which design decisions are made. Others like [16] take mechatronic aspects of the engineering processes into account. Furthermore, much attention has been paid to the field of engineering change management or the impact of geometrical changes of components on each other and the overall product [11]. In [1] a method for an automated structuring of geometric product data in a two step procedure is presented which can be used on 3D Computer Aided Design (CAD) models.

We propose a new way of modeling a product without employing a static product structure as the backbone of the Product Data Management (PDM) system. We achieve this by introducing a Virtual Product Model (VPM) that allows the derivation of specialized views at runtime (e.g. a BOM) and that is based on VPMCs. A VPMC is a collection of data elements from different departments (e.g. design or procurement) that describe a single component. By assigning a VPMC to a product, context-specific instances of this VPMC and of the data elements are instantiated in the following denoted as VPMC-U.

The structure of this paper is as follows: In Sect. 2, we describe the current practice in managing BOM data and design data in the automotive industry. In Sect. 3, we describe the concept of the VPM, how VPMCs can be used to provide design data (based on industrial use cases), and how views on the VPM can be derived. In Sect. 4, we illustrate, that the VPM contains (at least) the same information as the EBOM presented in Sect. 2 by providing an algorithm to transform a set of VPMCs and their context specific instances (VPMC-U) into an EBOM. Finally, we conclude with a summary and outline avenues of future work.

2 State-of-Practice in Providing Design Data

In practice, static product structures, such as BOMs, are used to provide a data backbone for PLM systems [1]. E.g. for automobile Original Equipment Manufacturers (OEMs) multiple types of product structures are important. Two of the most relevant ones are: (i) An EBOM to organize a product's design data and (ii) a MBOM that focuses on manufacturing part sequences. In the following we focus on EBOMs.

The main goal of an EBOM is to integrate design data into a BOM. For *releasing* design data (appending the corresponding data elements to an EBOM), a BOM-based approach is

no longer an adequate solution [5], because of the static and inflexible nature of an EBOM and the current scale of diversity. Such an EBOM has to meet requirements of all domain experts (from different departments like virtual prototyping, tool manufacturing, procurement) participating in the PDP. Therefore, the size of the structure easily gets unmanageable. Thus, PLM roll-out projects may fail, when designers are expected to use such an *overloaded* EBOM. Furthermore, transformations of static structures are computational complex and therefore very time-consuming.

The most important contribution of designers within the PDP is the so called *part-geometry alignment* that connects a part $\textcircled{\text{P}}$ (logistical, BOM-oriented element that carries organizational information such as suppliers and procurement channels) and a revision of a geometry $\textcircled{\text{G}}$ (design-oriented collection of 3DCAD-Files and several 2D-Drawings that carry information about the geometric characteristics of a part). In this paper we assume, that the relations between geometry revisions and parts are provided by the designer. Nevertheless, in previous work [3] we proposed an agent-based approach to (partly) automate such alignments. This part-geometry alignment can be used in one or multiple *contexts* \square (generic term for a collaborative developed product or derivative of a product).

A typical product structure that represents a context (see Fig. 1(a)) consists of *nodes* \square (to structure the virtual product) and BOM items (parts), that specify a component of the virtual product in the context of the EBOM. In order to model different derivatives of the same product (e.g. sedan, roadster, station wagon), the so called *technical validity* of those parts (part variance $\textcircled{\text{V}}$) is attached to the structure. E.g. the variance rule $(BODY = SW) \wedge (HD = LEFT)$ describes the validity for a *station wagon* with *left-hand drive*.

Furthermore, the validity of a part according to the context-specific PDP can be defined by assigning *milestones* $\textcircled{\text{M}}$ to the part-node, where a milestone marks the attainment of a specific stage of the PDP. For each milestone the components have to meet predefined characteristics and quality requirements. For Digital MockUp (DMU) a geometry- and context-specific *position* $\textcircled{\text{P}}$ needs to be defined (e.g. the four *rims* of a car are positioned differently and $POS = FrontLeft$ describes the left front wheel). In order to be able to differentiate between these alternate positions (e.g. to use only one positioned geometry for collision detection), *position variance* needs to be applied to the structure.

Taking the requirements above into account, such an EBOM easily gets unmanageable, due to its size and number of levels. They are the result of integrating multiple domains into a single structure. In contrast, the VPM provides the possibility to derive domain specific views based on the consolidated product data.

3 Providing Design Data using VPMCs

In this section we discuss the concept of the VPM based on VPMCs and their context-specific usages. In Sect. 3.2 we introduce the basic concept, and the differences between VPMCs and their context-specific *usages*. Sect. 3.3 shows how use cases derived from the

automotive industry, can be performed based on VPMCs. These use cases cover the evolution of a component over time and its usage in different contexts (derivatives or even products) with temporally disjoint PDPs (see Fig. 1(b)). Finally, Sect. 3.4 explains how to derive domain specific views on our VPM.

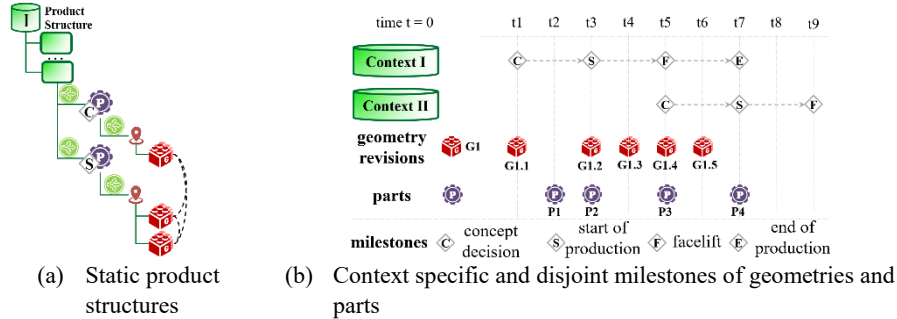


Fig.1. Current practice in providing design data using a static product structure

3.1 Use Case: Aligning Design and Logistic Data




Changes on a component usually result in changes on the geometric representation. In practice not every change on the geometry (new revision) is reflected on the part-geometry alignment, because (i) parts do not depend on the geometric level exclusively, but on changes regarding the procurement (or other organizational matters) as well; and (ii) not every geometry revision fulfills the quality requirements for post-design phases of the PDP. Taking into account the parallel and temporal independent development of multiple contexts (derivatives or products) different revisions of the component (and therefore of the geometry) might be valid for different context-specific milestones. Example:

The component *rim* evolves over time t (see Fig. 1(b)). At milestone *concept decision* of the context *I (sedan)* in $t = 1$ only the first geometry revision of this *rim* is available. At this point there is no need for aligning a part, because organizational information (such as suppliers) are needed not until the production planning starts at $t = 2$.

In $t = 3$ (milestone *start of the production* of context *I*) the latest part geometry alignment consists of the second geometry revision *G1.2* and part *P2* (e.g. *silver rim*). Therefore, this alignment can be assigned to the *sedan* (context *I*) for milestone *S*. In $t = 4$ a new geometry revision has been development, but no part is assigned, because milestone *S (start of production)* of context *I* has past, and neither milestone *F (facelift)* of context *I* nor a crucial milestone of another context has been reached (see Fig. 1(b)).

The development of the *roadster* (second context *II*) starts at $t = 5$. At this point there are four different revisions of the geometry. The newest part is *P3 (black rim)*. Therefore, the fourth geometry and the part *P3* are assigned to the context *roadster* for milestone *C*.

3.2 Virtual Product Model Component and VPMC-U

As mentioned in Sect. 2, parts , geometries , the relation(s) between them, and context-specific information appended to these relations are the basis for components of the VPM. The set of these information regarding a single component is defined as a VPMC . Due to geometry being the most characterizing element (in contrast parts carry much organizational information and may differ according to market conditions), each VPMC is geometrically unique.

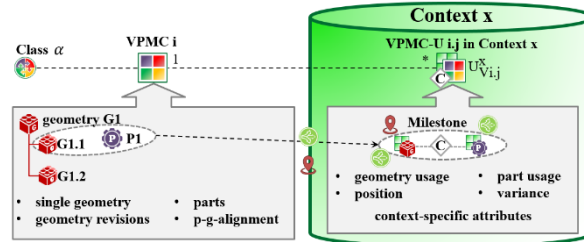










Fig.2. Concept of VPMCs, VPMC-Us, and dynamic views

A VPMC evolves over time, because product development is not a linear process, but a continuous sequence of changes [4,14,8,6]. Taking into account the efforts of most practitioners to increase the commonality of different products [2], a component is most likely *used* in multiple contexts . In the following we refer to this context-specific objects as VPMC-Us . Accordingly, a VPMC is related to multiple VPMC-Us. Because VPMCs contain parts and geometries, these elements are related to multiple *part usages*  / *geometry usages*  as well. These usages are related again to other context-specific elements such as milestones , the positions  of the geometry in that context, and variance expressions  regarding the geometrical (on the geometry usage) or technical validity (on the part usage). A VPMC itself has no relation to any context, but to several VPMC-Us; each VPMC-U serves as a collector for the context specific elements. The number of VPMC-Us depends on (i) the number of contexts the VPMC is used in, (ii) the number of occurrences of that VPMC in each context (e.g. a car typically has four tires), and (iii) the number of updates *inside* a VPMC-U (traceability - see Sect. 3.3). Yet other elements of the VPM are predefined, structured, and tree-like *views*. The leaves of such a tree are classified . This classification is related to the class of the VPMCs. E.g. one leaf contains all components of the class *rim*. Therefore, all VPMCs with a relation to this class can be assigned to the *rim* node. Applying this concept to VPMC-Us provides a context-specific and dynamic view based on VPMCs and classified views (see Sect. 3.4). Fig. 2 illustrates the differences between VPMCs and VPMC-Us.

In summary a VPMC is a reusable component with a unique geometry. It contains the following data elements: (i) A single geometry with several revisions (to trace changes on

the geometry); (ii) multiple parts that are realized by the geometry revisions; (iii) relations between geometry revisions and their parts (part-geometry alignment); and (iv) a reference to a class (classification of VPMCs).

Furthermore, a VPMC-U is a context-specific instance of a VPMC. It contains the following data elements: (i) A geometry usage with a relation to a geometry revision; (ii) the position of this geometry revision in the context; (iii) variance rules, that describe the validity of the geometry; (iv) several alternate geometry usages because of different positions; (v) a part usage that represents the part in that context; (vi) variance rules (on the part usage) that describe the technical validity of the part; and (vii) a milestone that defines the stage of the PDP the VPMC-U is released for.

3.3 Reflecting Changes of Components using VPMCs

In this section we discuss the data model of VPMCs and their context-specific usages (VPMC-Us) capable of managing the evolution of a component (especially their containing part-geometry alignments) in relation to context-specific and disjoint milestones (Fig. 1(b)).

At first the designer creates a new geometry $G1$ representing the *rim* of the previous example. Thus a new VPMC is instantiated. Furthermore, the designer provides a class α , that specifies the “type” of the VPMC (e.g. *rim*).

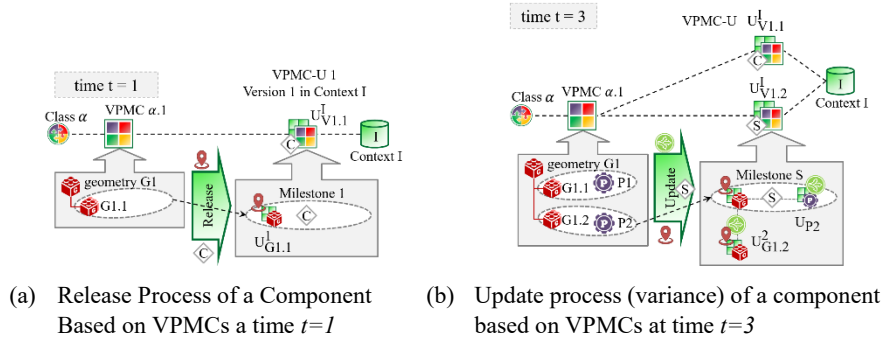


Fig.3. Providing design data using VPMCs

At $t=1$ (see Fig. 3(a)) the designer assigns $G1.1$ of the geometry $G1$ at a context-specific position for milestone C in context I (sedan). Therefore, the following data elements are instantiated: (i) A new VPMC-U $U_{V1.1}^1$ that serves as a collector for the context-specific occurrences of the VPMC; (ii) a geometry usage $U_{G1.1}^1$ that represents the current version of the *rim* in the sedan; and (iii) a node representing the position with a relation to the geometry usage $U_{G1.1}^1$.

At $t=2$ a new part (*silver rim*) is associated to the geometry revision $G1.1$. The VPMC-U $U_{V1.1}^1$ can be updated automatically, because the part-geometry alignment is unique. The data elements created are: Firstly, a new part usage U_{P1} that represents the *silver rim* in the

sedan (context I). Because of the existing relation between the geometry revision and the part, the corresponding usages are related to each other. Furthermore, both usages share the same validity for milestone C . Secondly, a variance expression that describes the validity of the part in relation to different configurations of context I . E.g. this specific *silver rim* might only be valid for cars with a costly equipment package. This variance expression is related to the part usage directly.

At $t = 3$ (see Fig. 3(b)) the geometry of the *silver rim* is updated ($G1.2$), a new part $P2$ (*black rim*) is provided, and both are aligned to each other. These usages (and there the collector $U_{V1.2}^l$) are valid for the sedan's milestone S . The first position of the geometry revision, the validities on the geometry usage $U_{G1.2}^1$ and/or the part usage U_{P2} , can be taken from the previous VPMC-U or, on behalf of the designer, replaced by new information. Furthermore, there might be an alternate position of the geometry revision (e.g. to model the *rim* turned right or the front left *rim*). The data elements created are: (i) a geometry revision $G1.2$, a part $P2$, and an alignment between these elements, (ii) a VPMC-U valid for milestone S with a relation to the VPMC, (iii) a new part usage U_{P2} for the part $P2$. The variance expression of this part usage is taken from the part usage of the previous VPMC-U $U_{V1.1}^l$, and (iv) two new geometry usages $U_{G1.2}^1$ and $U_{G1.2}^2$ for the geometry revision $G1.2$. The position of the first geometry usage is taken from the geometry usage $U_{G1.1}^1$ of the previous VPMC-U $U_{V1.1}^l$ the alternative position and the variance rule are provided.

In summary this approach leads (for one VPMC) to a set of context-specific VPMC-Us with different validities. Each VPMC-U describes the current state of the related VPMC in a specific context at a particular point of the context's PDP. The separation of components into context-free (VPMC) and context specific (VPMC-U) elements, enables the development of a component to be independent of its actual usages. This solves the problem of using the same component (VPMC) in different contexts with disjoint PDPs mentioned at the beginning of Sect. 3.

3.4 Domain-Specific Views on a VPM

One of the main advantages of our VPM is the possibility to create *views* at runtime (see Sect. 3.2) by assigning VPMC-Us to leaf nodes of arbitrary structures based on their classification. Therefore, the computational complex transformation (see Sect. 2) mentioned in Sect. 2 are obsolete.

Furthermore, these *views* can either be context-specific (see Fig. 4(a)) or a collection of VPMC-Us of different contexts by adding context-specific nodes subordinated to the classified nodes of the arbitrary structures (see Fig. 4(b)).

The relation between the context-specific nodes and the VPMC-Us can be derived, because every VPMC-U knows the context it is assigned to. Therefore, this approach can be used to calculate structures at runtime, that describe the share of common parts in different contexts. Moreover, by selecting only the data elements of the VPMC-Us relevant

for a specific domain (e.g. parts for purchasers or geometries for designers) these views become *domain-specific* (see. the right side of Fig. 1(a)).

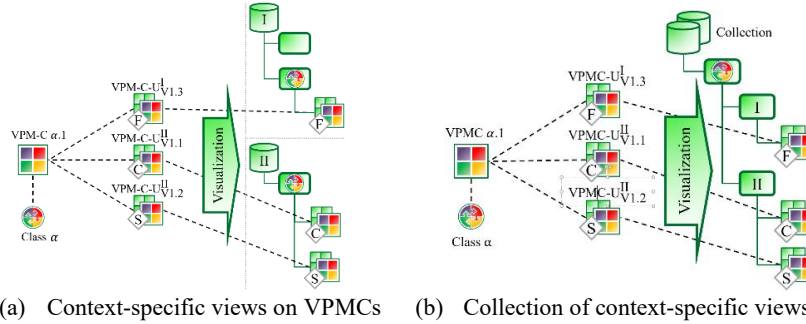


Fig.4. Deriving views from a VPM

4 Evaluation

To evaluate the information content of a product model based on VPMCs, in the following we describe an algorithm to transform a VPM into an EBOM illustrated in Sect. 2. It is unlikely, that this algorithm is used in practice, because of the ability of the VPM to create dynamic views at runtime, but it shows that the VPM contains (at least) the same information as an EBOM. See Fig. 5 for an illustration of the transformation and Alg. 1 for a description in pseudo code.

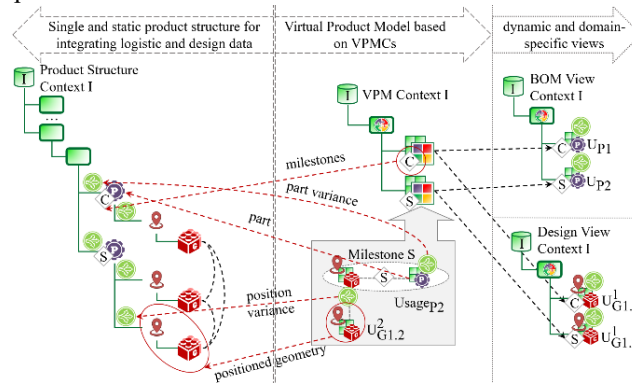











Fig.5. Static Product Structure vs. Domain-Specific and Dynamic Views on VPMCs

The algorithm takes a mapping from *classes*  to nodes  of the EBOM and a context  as an input. For each class of the mapping the corresponding node of the EBOM is determined using the given mapping. For each VPMC-U  of the current class a new part-node  is generated, the validities (variance  and milestone ) are retrieved from the VPMC-U, and assigned to the new part-node. Next a node representing the *first position* 

of the VPMC-U's geometry revision  and the revision itself are appended. The final steps are to iterate over all alternate positions of the VPMC-U's geometry, to create nodes that represent these positions, to set the geometric validity according to the VPMC-U, and to append the geometry revisions.

Algorithm 1 VPM2ProductStructure Transformation

```

1: procedure transformVPMCs2Structure(Mapping: Class → Node, Context context)
2:   for all Class class : mapping.getClasses() do                                ▷ all classes of VPMCs
3:     Node node = mapping.getNode(class)                                       ▷ get node in the structure
4:     for all VPMC-U vpmcu: context.getVPMCUUsages() do
5:       if vpmcu.getVPMC().getClass() == class then                            ▷ check class membership
6:         Node partNode = node.appendPart(vpmcu.getPartUsage(), node)          ▷ part
7:         partNode.setVariance(vpmcu.getPartVariance())                        ▷ part variance
8:         partNode.setValidity(vpmcu.getMilestone())                            ▷ milestone
                                                                                   ▷ append first geometry in first position
9:         Node posNode = partNode.addPositionedGeometry(vpmcu.getGeometryUsage())
                                                                                   ▷ add alternate positions
10:        for all GeometryUsage altGeoUsage: geoUsage.getAlternatives() do
                                                                                   ▷ add position node and geometry revision from geometry-usage
11:          Node altPosNode = partNode.addPositionedGeometry(altGeoUsage)
12:          altPosNode.setVariance(altGeoUsage.getPositionVariance())

```

5 Conclusion

In this paper, we described a data model for managing the development of complex products, where several domain experts take part in the development of new products and each domain has its own and specialized view on an overall product model. We proposed a novel *Virtual Product Model (VPM)* that supports *reusability*, *transparency*, and *continuity*. Each component of our VPM is modeled as a combination of context-free *parts*, *geometries* (Virtual Product Model Component (VPMC)), and their context-specific counterparts, *part usages*, *geometry usages* extended with *milestones* and *variance-expressions* and combined in a VPMC-U. As we pointed out in [13], the VPMCs are extended by domain experts within their specialized views throughout the Product Development Process (PDP). Therefore, the approach presented in this paper allows the derivation of specialized views (BOM or design oriented) on a single product or on a collection of multiple contexts. This reduces the amount of information each domain expert is required to manage.

To summarize, the main contributions of this paper are: (i) an analysis of an industrial use case to determine the minimal set elements needed in a Product Data Management (PDM)-System to support the development process, (ii) a conceptual definition (based on the use case analysis) of VPMCs and their elements to support the release process of design information, and (iv) the derivation of specialized and user-related views on the VPM.

Future work will focus on (i) an agent based model to handle the dependencies between elements of VPMC [3], and (ii) methods for Feature Model Analysis to increase the transparency regarding the reuse of individual components in different products.

References

1. Adolphy, S., Grosser, H., Kirsch, L., Stark, R.: Method for automated structuring of product data and its applications. *Procedia CIRP* 38, 153–158 (2015)
2. Benavides, D., Segura, S., Ruiz-Cortés, A.: Automated analysis of feature models 20 years later: A literature review. *Inf. Syst.* 35(6), 615–636 (2010)
3. Bender, J., Kehl, S., Müller, J.P.: A comparison of agent-based coordination architecture variants for automotive product change management. *Multiagent System Technologies, LNCS*, vol. 9433, pp. 249–267. Springer International Publishing, Cham (2015)
4. Bucciarelli, L.L.: *Designing Engineers. Inside technology*, INSTITUTE OF TECHNOLOGY (1994)
5. Chatras, C., Giard, V., Sali, M.: High variety impacts on bill of materials structure: Carmakers case study. *IFAC-PapersOnLine* 48(3), 1067–1072 (2015)
6. Cheng, H., Chu, X.: A network-based assessment approach for change impacts on complex product. *Journal of Intelligent Manufacturing* 23(4), 1419–1431 (2012)
7. Clarkson, J.P., Simons, C., Eckert, C.: Predicting change propagation in complex design. In: *ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Pittsburgh, Pennsylvania (2001)
8. Cross, N.: *Engineering design methods: Strategies for product design*. Wiley, Chichester, 3. ed., reprinted. edn. (2005)
9. Dr. Kees, M., Seibertz, A.: Compositional variant management and its application in embedded software development (29042010)
10. Galle, P.: The ontology of ideo's fbs model of designing. *Design Studies* 30(4), 321–339 (2009)
11. Hamraz, B., Caldwell, N. H. M., Clarkson, P.J.: A holistic categorization framework for literature on engineering change management. *Systems Engineering* 16(4), 473–505 (2012)
12. Jarratt, T., Eckert, C.M., Caldwell, N., Clarkson, P.J.: Engineering change: an overview and perspective on the literature. *Research in Engineering Design* 22(2), 103–124 (2011)
13. Kehl, S., Stiefel, P., Müller, J.P.: Changes on changes: Towards an agent-based approach for managing complexity in decentralized product development. In: *International Conference on Engineering Design (ICED 15)*. vol. 3, pp. 219–228. Milan, Italy (2015)
14. McMahon, C.A.: Observations on modes of incremental change in design. *Journal of Engineering Design* 5(3), 195–209 (1994)
15. Oldham, K., Kneebone, S., Callot, M., Murton, A., Brimble, R.: Moka-a methodology and tools oriented to knowledge-based engineering. *Proceedings of the Conference on Integration in Manufacturing, Göteborg, Sweden, 6-8 October 1998*. vol. 8, p. 198 (1998)
16. Roucoules, L., Noel, F., Teissandier, D., Lombard, M., Debarbouille, G., Girard, P., Merlo, C., Eynard, B.: Ippop: an open source collaborative design platform to link product, design process and industrial organisation information. In: *6th International Conference on Integrated Design and Manufacturing in Mechanical Engineering, IDMME'06*. p. CDROM (2006)
17. Tekin, E.: A method for traceability and “as-built product structure” in aerospace industry. *Procedia CIRP* 17, 351–355 (2014)
18. Tseng, M.M., Jiao, J.: Mass customization: 25. In: *Handbook of Industrial Engineering*, pp. 684–709. John Wiley & Sons, Inc (2007)