INTERNATIONAL DESIGN CONFERENCE – DESIGN 2020
https://doi.org/10.1017/dsd.2020.303

CUSTOMER SPECIFIC COMPATIBILITY MATRICES FOR FUNCTIONAL INTEGRAL PRODUCT ARCHITECTURES

J. Siebrecht [1,✉], G. Jacobs [1], C. Konrad [1], C. Wyrwich [1] and W. Schäfer [2]

[1] RWTH Aachen University, Germany, [2] University of Siegen, Germany
✉ justus.siebrecht@imse.rwth-aachen.de

Abstract

Supplier of system components face the challenge of customer requirements influencing the property level functional integral product architectures. For this, solution approaches focusing on the re-use of pre-engineered part variants are not applicable. However, to generate a valid product structure, customer-specific properties have to fit modelled product knowledge. Therefore, the approach models a reference class structure and analysis compatibilities on the property level for customer specific inputs concerning explicit product knowledge and constraints.

Keywords: complexity, systems engineering (SE), mass customisation, compatibility, property variance

1. Introduction

Today, companies in mechanical engineering often face a displacement market with rising pressure from the growing international competition (VDMA, 2018; Friedli and Schuh, 2012, p. 11). Especially suppliers of system components for special machinery and plant engineering experience the need to differentiate themselves from the competition. However, they face the additional challenge of developing functional, narrowly defined products, where additional features only partially lead to differentiation in the customer's view. Pre-engineered component variants, whether in the form of platforms, modular systems or series, always represent a trade-off in regarding to the optimal performance for customer specific requirements (Ehrlenspiel and Meerkamm, 2017, p. 884; Friedli and Schuh, 2012, p. 14). Industrial cooperation with component suppliers confirm Andersons statement, that individual combinations of pre-engineered parts are often no longer sufficient to satisfy the customers need for individual developed performance (Anderson and Pine, 1998).

The customers, OEMs and system manufacturers, aim for optimal performance of their own systems, which depends on every used component. Cutbacks in the performance of single components have a direct impact within the overall system and their unique selling points. Component suppliers who only sell based on a fixed components catalogue lose flexibility towards the customer, which often is an exclusion criterion for an order in a displacement market (Franke, 1998).

The fundamental goal for these suppliers is to maximize customer coverage with customer specific products while minimizing internal costs (Ehrlenspiel and Meerkamm, 2017; Feldhusen and Grote, 2013; Gausemeier et al., 2015). The classic approach to solving this challenge is the re-use and combination of fully engineered, pre-designed parts and assembly variants according to the customer's individual requirements (Blees, 2011; Krause and Gebhardt, 2018; Nurcahya, 2007; Wyrwich and Jacobs, 2019; ElMaraghy et al., 2013). However, if customers require optimal functionality,

combinations of pre-defined parts are not sufficient and the supplier has to optimize components on the property level. Since companies are not able to predict customer requirements accurately, no development can occur beforehand. This inability to use pre-engineered parts represent a risk entering into negotiations through incalculable expenses and development iterations. Functional integral products and their interactions on the property level increase this complexity. The results of the subsequent steps in the development process based on customer-specific property values are unpredictable (Katzwinkel et al., 2018; Konrad et al., 2017; Konrad et al., 2019). One of the main results is the constantly growing flood of product data (internal variance) based on customer-specific requirements that must be managed and included in the manufactures portfolio (Feldhusen et al., 2007; ElMaraghy et al., 2013).

This situation requires new approaches in the development process of technical products. The supplier needs the ability to compare customer requirements and customer-specific properties to explicitly modelled product knowledge in order to manage and structure different simultaneous incoming and running offers and development processes. Model Based Systems Engineering (MBSE) presents a useful approach in managing the complexity and needed consistency of the presented challenges. However, a successful implementation is redundant without a clear methodological approach (Gausemeier et al., 2015; Konrad et al., 2019).

This contribution presents an approach to manage the complexity of functional integral product structures through the mapping of modelled product knowledge based on customer properties. The attention lies on the interdependencies on the property level. For the methodological foundation, the approach incorporates parts of existing methods for modular products with the extension into the property level of the product structure.

2. State of the art and challenges

The challenge for suppliers lies in the fact that classical methods to represent possible configurations are based on the assumption, that the supplier is able to predefine the product portfolio as a modular system (Konrad et al., 2017). However, since the customers' requirements cannot be predicted (Otto and Wood, 2001), and pre-engineered variants do not reflect customer requirements accurately enough, existing methods fail to support this configuration (Baumberger, 2007; Konrad et al., 2017). Furthermore, to ensure economic efficiency, processes must react to changes in near real time. This requires a cross-departmental and cross-hierarchical methodology (Grauer et al., 2010a). The large number of interactions at the property level and the lack of methodological support hampers the stringent and profitable development of new configurations.

2.1. Functional integral products

Technical products can be analysed in the context of system theory at different relational dimensions (Göpfert, 1998). The most frequent consideration of relational dimensions is the product architecture. On the one hand, it describes the physical subdivision into assemblies and components and, on the other hand, the overall function as well as the underlying sub functions. On the lowest layer of both views, the product architecture connects the functional and structural view. In system theory, the number and arrangement of relationships between elements in a specific relational dimension, can assume either integral or modular traits. For the two dimensions on the product architecture, this leads to the matrix shown in Figure 1. The prerequisite of many variant management approaches is to achieve a modular product architectures (Firchau and Franke, 2002; Göpfert, 1998; ElMaraghy et al., 2013). This leads to an approximately structural and functional independent development, testing and production of the modules. In this case, a physical modular product architecture is present. This in turn represents integral traits on the functional view of the product, where separate functions depend on several parts and a single part incorporates multiple functions.

Component suppliers face customers, where the requirements less often influence the modular product structure, but rather the optimal realisation of their needs in the functional dimension. The customer specific change of a part property has a multitude of effects on other components via the functional interdependencies. Modification of individual part properties leads to far-reaching, complex consequences for other parts and their properties. Konrad describes this problem and inserts the property level into the product architecture (Konrad et al., 2017), resulting in the simplified product architecture shown in Figure 2. The modular character in the physical structure remains, where parts consist of numerous properties

https://doi.org/10.1017/dsd.2020.303 Published online by Cambridge University Press

while the product architecture assigns each property only to a single part (no. 1). The integral trait lies in the functional, left part of the product architecture. Multiple use of properties in functions (no. 2), as well as the dependence of single functions on several properties (no. 3) are the characteristics.

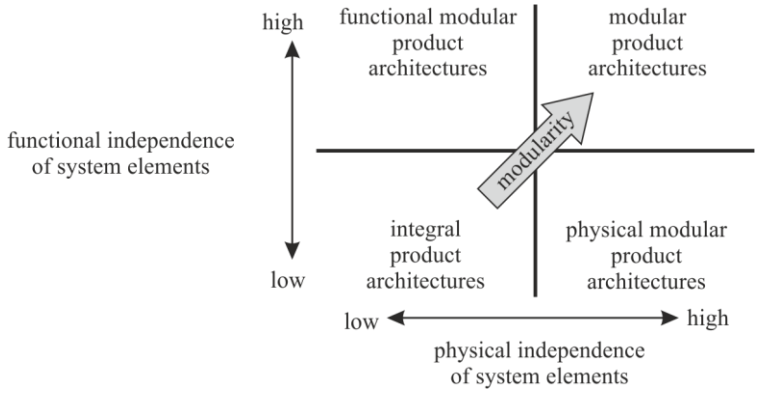

Figure 1. Functional and physical dimension of product architectures (Göpfert, 1998)

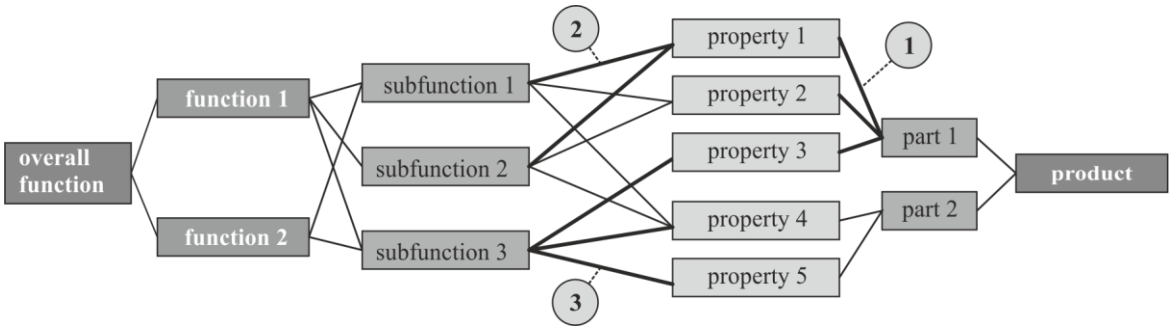

Figure 2. Extended product architecture to the property level (Konrad et al., 2017)

2.2. Reference product structures and configuration matrices

The basic idea of reference product structures (RPS) is the representation of product variants and combinations based on a similar product architectures (Feldhusen and Grote, 2013, p. 795; Feldhusen et al., 2007; Nurcahya, 2007; Jiao and Tseng, 1999). Figure 3, no. 1 - 3 shows the RPS based on the example of a bicycle. The RPS represents the fixed product structure and arrangement of the bicycle through providing default, selection and optional modules. Default modules are always included, whereas the customer chooses optional modules as well as variants from a pre-engineered selection. The RPS also includes customer specific parts, which have to comply with the defined interface. The RPS is especially useful for modular product architectures, where modules are functionally and physically independent. All parts in the RPS have a fixed material number in the ERP system with given costs and production plans and all part properties are set.

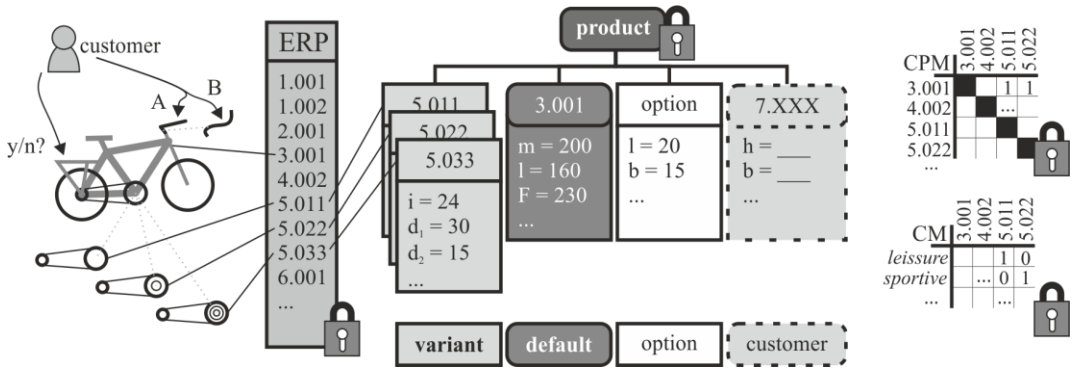

Figure 3. Classic approach for a reference product structure and configuration matrices

https://doi.org/10.1017/dsd.2020.303 Published online by Cambridge University Press

Configuration and compatibility matrices illustrate underlining dependencies in the product and add additional strategic restrictions. The compatibility matrix (CPM) contains the compatibility for a group of similar elements, for example parts in a product, and defines the pairwise occurrence or exclusion of these elements (Figure 3, no. 4). The configuration matrix (CM) links two different groups of elements and thus shows how two views are interconnected. A good example is the connection of a customer requirement for a sportive bicycle with a subset of transmissions and wheel sizes. This results in symmetric CPMs and mostly asymmetric CMs. Most approaches use a combination of CMs and CPMs. Puls for example uses the configuration and compatibility matrices to link the customer view with the technical product view as shown in Figure 4 (Puls, 2003). It can contain part compatibilities that are independent of property values and functional interactions, such as consciously made sales or strategic decisions.

As long as products are combinations of these pre-engineered parts, the RPS represents a useful tool in combination with the information contained in the matrices (Feldhusen and Grote, 2013, p. 802). The reference product structure shows the arrangement and number of elements in the product structure. Configuration and compatibility matrices support and restrict the selection of variants and options concerning the customer as well as technical view. If for example dependencies exist between the two selections in Figure 3, a selection in one part of the product structure can restrict the possible variants in the other or even results in a fixed selection. This allows an easy mapping of compatibilities for modular product architecture with a fixed arrangement of the parts in a fixed timeframe.

3. Challenge regarding creation of variants and solution approach

Customization on the property level result in challenges regarding the number of variants created and the corresponding development processes (Firchau and Franke, 2002; ElMaraghy et al., 2013). A functional integral product architecture amplifies the number of changes made based on customer requirements. Figure 4 shows an example of a functional integral product and the results produced by customer requirements on the property level. Product developers compare customer requirements with existing parts, customize and create new parts, which expand the portfolio. The differentiation between variants and default parts remains since variants have underlying differentiation characteristics (see Figure 4, form 1 or 2). Every change on the property level has the possibility to further change additional parts through the functional dependencies. The result is an unmanageable combinatory solution space and product portfolio. Sometimes to a degree, where the new customer specific development seems easier, faster and more effective, deliberately ignoring existing variants and products (Feldhusen et al., 2007).

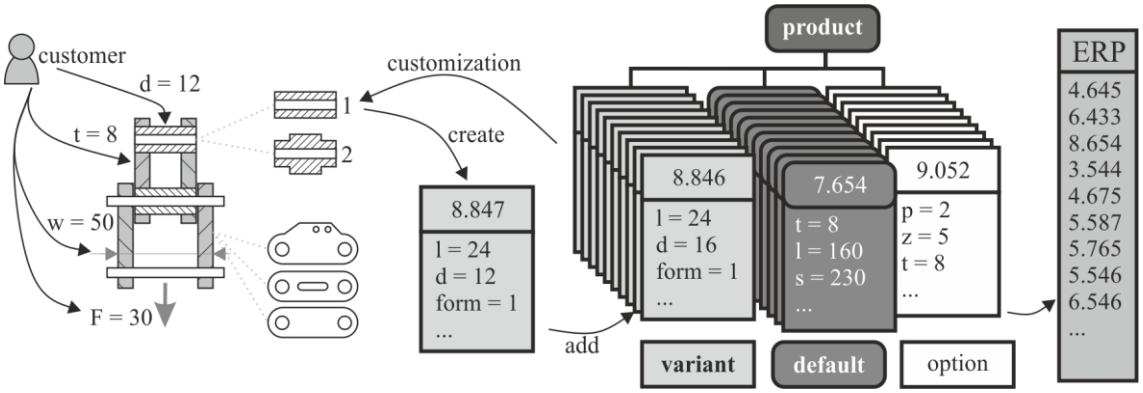

Figure 4. Variance explosion in functional integral products

Although including all engineered customer variants is possible, the rapid growth of customer variants leads to an enormous and growing number of data as well as exponentially more dependencies. For customer specific integral products, universally valid RPS, CM and CPM are almost impossible. The company has to manage all these elements and keep the system up to date to re-use variants in a sales offer or the development process. In the midst of an almost zero reuse rate for variants, the expenses to maintain and check every newly added variant presents no feasible option.

https://doi.org/10.1017/dsd.2020.303 Published online by Cambridge University Press

Nevertheless, it is necessary for suppliers to keep development efforts and iterations in the development process, especially when creating an offer, at a minimum. This concerns application of product knowledge and the mapping of customer requirements to this knowledge, for example functionality, costs or delivery times. Therefore, a solution is necessary which structures the product knowledge to realize a better trade-off between scale and scope principles as well as clearer incorporation of customer requirements. (ElMaraghy et al., 2013; Jiao et al., 2007)
The solution approach aims at reducing the complexity for functional integral product architectures based on customer specific properties when applied by product developers. The focus lies in mapping customer requirements with modelled product knowledge, either in case of a sales offer process or as a starting point for the customer specific development process. The hypotheses of the approach is, that splitting RPS and CPM into a structural part and a functional part will improve the transparency of the product knowledge as well as support the usage when analysing customer requirements. Combined with a strict separation of product knowledge and engineered variants, a clearer processing of customer requirements should be possible.

3.1. Reference class structure

The product structure illustrated so far is not yet suitable for mapping the functional, technical compatibilities, i.e. the integral functional character of the product. The representation of all component variants, as well as the manual linking of deviances, would drive the representation and the maintenance effort into an inappropriate range. In addition, it would not have the desired effect for the reasons given in the introduction. Franke suggests converting fixed variants into parameterized classes, since the use of pre-engineered modules is only viable in a clear state of market leadership (Franke, 1998). This step represents a change from customer specific parts to customer specific instances and standardised classes. These classes allow the desired reuse of product knowledge (Grauer et al., 2010b) instead of the reuse of the results from previous development processes, e.g. parts. In addition, the classes contain the constraints and boundary conditions that describe the variance of possible instantiations, an information that normally is not visible in the RPS.
As defined in section 3, the approach separates structure and functionality. This generalizes the reference product structure to a reference class structure (RCS) as shown in Figure 5. The RPS (on the left) contains numerous variants for different parts (options and variants alike). Analysing and dissecting the variants of parts, based on properties (no. 1) results in classes with similar constraints and characteristics (no. 2). Overlapping properties of these classes are contained in a general class.

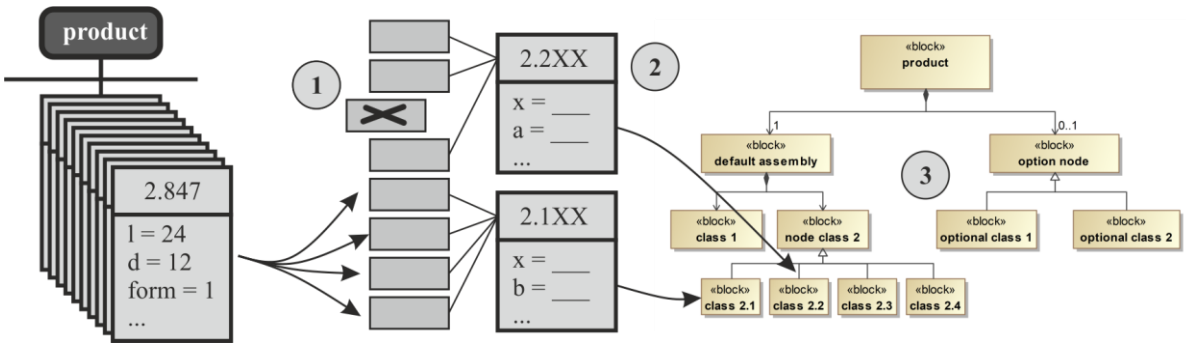

Figure 5. Transformation of variance explosion into structured classes with inheritance

For the later creation of CM and CPM, the approach models the resulting generalizations, specifications and compositions in SysML. The RCS can also contain defaults, options and selections with multiplicity and optional nodes (Figure 5 no. 3). The inclusion of generalizations is the first differentiation from the classes RPS approach.
The second differentiation is the abandonment from fixed, fully engineered variants. The RCS refers to the individual classes, not instances of these classes. Therefore, the portfolio does not result from combinations of pre-defined classes without a full set of values, but from a solution space with interconnected value ranges.

https://doi.org/10.1017/dsd.2020.303 Published online by Cambridge University Press

For a start, the RCS contains an even greater, unmanageable variance of combinations on the property level compared to the RPS. However, it simplifies the representation of classes and their interdependencies. The classes contain the constraints and validity areas that must apply to each instantiated part of the class structure. Figure 6 shows a detail of the RCS and the different constraints, which describe the functionality of the product. Part functions (no. 1) represent the dependency of various properties (values) inside a single part (class) with specified in- and outputs. Boundary constraints (no. 2) describe the validity areas for various property inputs. Crosscutting constraints (no. 3) (Schneeweiss and Hofstedt, 2013) logically connect different areas of the RCS, either defining clear dependencies between values (in- and output) or parent boundary constraints. This allows the assimilation and reuse of previously identified knowledge from various developed products into new configuration processes, which is crucial for an integrated value chain for suppliers (Grauer et al., 2010b). Through a clear differentiation of departments in a company, management can control access rights to different dependencies. This supports a controlled implementation of new rules. As in the classical approach, a cross section department administrates the structure in integration if the product knowledge.

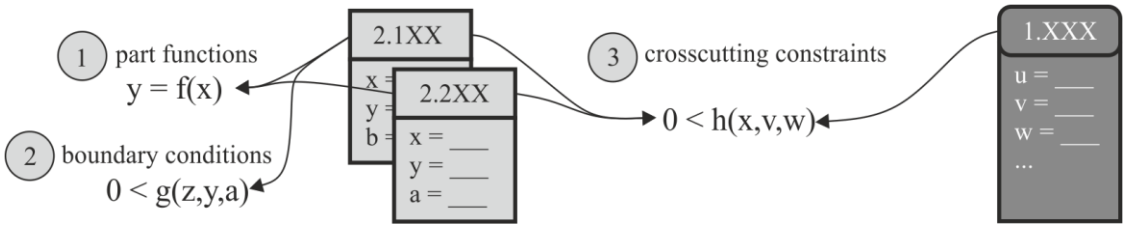

Figure 6. Given constraints between classes of the RCS

3.2. Classic use of configuration and compatibility matrices

The approach uses a bottom-up and top down partial approach, to reduce the solution space. The first part of the solution is a top-down approach consisting of the applicable state of the art methods for higher-level product dependencies.

Figure 7 shows how product dependencies above the parameter level result in configuration and compatibility matrices. On the left side, it presents possible dependencies for a product structure. These represent example classes and their composition as well as generalisations (no. 1) but also dependencies between parts and product structure nodes (no. 2). The dependencies presented on this level are all independent from customer specific properties. These include management decisions, superordinate usage relationships for different industries and sales regions. SysML represents these connections as either generalization, composition or associations. System modellers can extract these relationships as a dependency matrix (no. 3) and form CMs and CPMs for different relationship dimensions through either the three named connections or complex metachain searches. This allows the extraction and export of CM and CPMs (no. 4) based on product knowledge described in SysML in the original intended way.

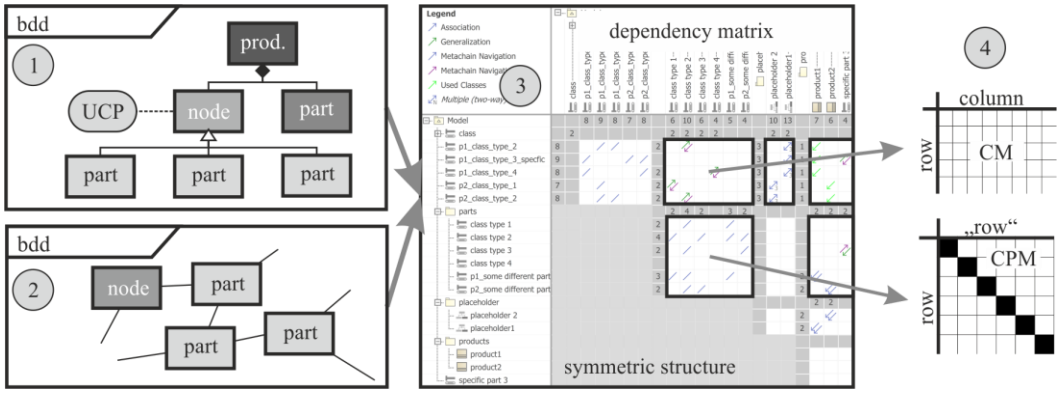

Figure 7. Usage of dependency matrices to extract configuration and compatibility matrices

https://doi.org/10.1017/dsd.2020.303 Published online by Cambridge University Press

3.3. Customer specific configuration and compatibility matrices

The bottom-up part of the approach is shown in Figure 8. For a given, incomplete set of starting properties through customer requirements, the approach instantiates all classes, independent from the position or type (optional, default or selection classes) in the structure. This step simulates and solves all part functions and validates boundary constraints (see Figure 8) for the given properties. An instance list summarises the instances, as well as the validation results. The analysis for all class constraints leads to the overall compliance of a class concerning the given properties. For the selection of class 2, this steps results in the remaining subclasses 2.1XX, 2.2XX and 2.4XX (Figure 8). It is possible to see, that class 2.3XX is not able to comply with three of the shown constraints, which makes the class inapplicable for the given properties and the current customer. Indistinct boundary conditions, for example through the lack of given properties, still represent possible compliance, so long as a constraint is not clearly violated. The usage of SysML and a system modeller supports these steps and allows easy changeability of the constraints and underlying dependencies.

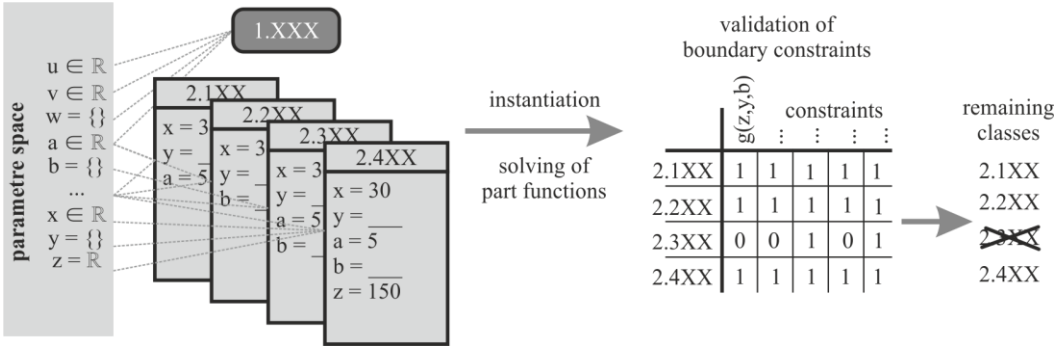

Figure 8. Direct validation check of all classes resulting in Boolean vectors summarising the results of the given constraints

If the direct class validity query is completed, crosscutting constraint checks take place. These check constraints, where numerous classes and their properties are involved in the fulfilment. Therefore, the approach utilises the remaining customer specific instance lists (see Figure 8) as an input for the validation algorithm. The algorithm has access to the underlying crosscutting constraints used in the class structure and generates a matrix for each constraint, as presented in Figure 9, where the classes 2 and 3 have to comply with the constraints *n* and *m*. Each of the instance combinations is an input for the corresponding constraints and therefore results in a Boolean statement about the compatibility of said combination. The algorithm ignores previous dismissed classes, for example class 2.3 in the creation of the matrices.

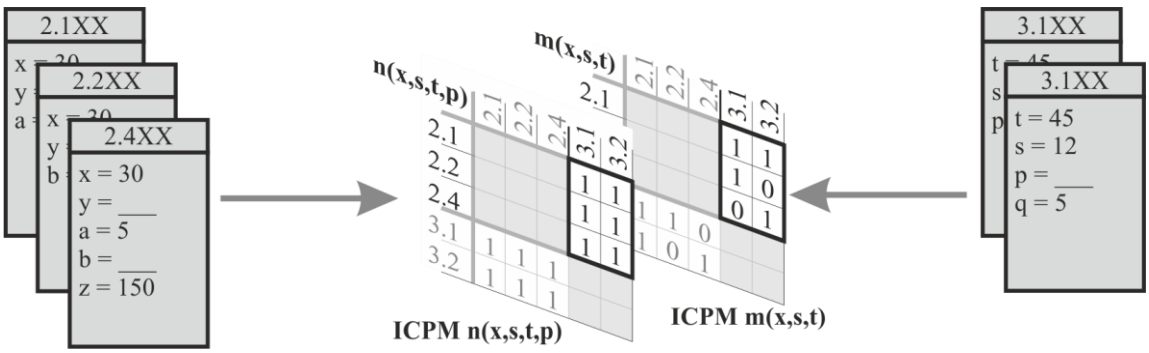

Figure 9. Generation of incompatibility matrices for two given crosscutting constraint

Since only incompatibilities show clear FALSE statement, this step leads to incompatibility matrices (ICPM, see Figure 9). Here, Boolean FALSE represents clearly identifies incompatibilities and Boolean TRUE states that the combination is or might be possible, even though some values might be missing. Figure 10 shows, how the algorithm superimposes all ICPMs (see Figure 9) and CPMs (see

https://doi.org/10.1017/dsd.2020.303 Published online by Cambridge University Press

Figure 7) into a single, overall product compatibility matrix (PCPM). Only classes that have individual compatibilities in all constraints are overall compatible with each other concerning customer specific properties. Figure 10 shows this for two-dimensional matrices but is also mathematically possible for more dimensions. The resulting PCPM now contains the combined information of the class combinations based on customer requirements. The different matrices are checked for internal contradictions and dependencies as proposed by Puls (Puls, 2003), which might result in further reduction of the possibilities.

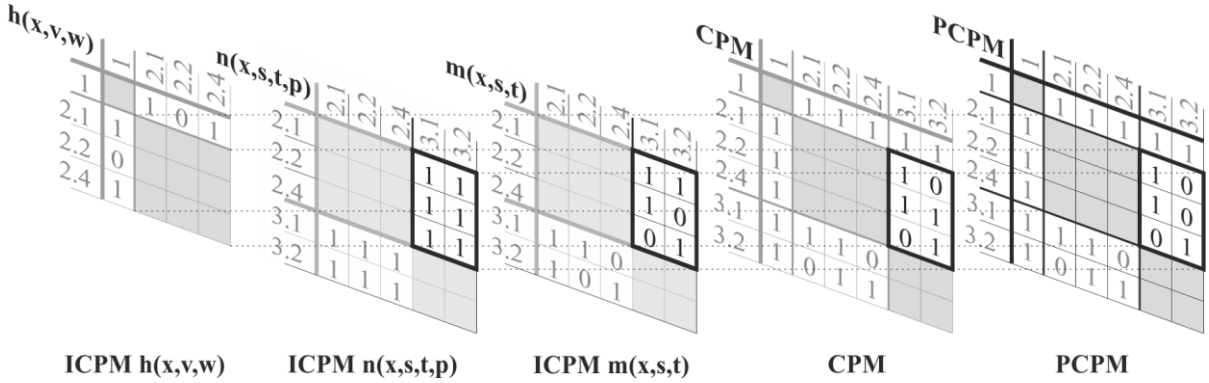

Figure 10. Combination of ICPMs into an overall ICPM

3.4. Application of the presented approach

The presented approach is part of an ongoing research project with industrial partners. The test case is a reduced product portfolio containing two product families. A commercial system modeller contains the underlying dependencies and structure documented in SysML. It further simulates the test process, export of the dependency matrices and controls the main function, calling the constraints given by different departments.

The PCPM and RCS supports the evaluation of constraint compliance across all product elements based on the modelled constraints and used input properties. Therefore, product developers can analyse customer specific inputs for a generic product structure concerning the technical compatibilities in the PCPM. Two possible outcomes remain. On the one hand, no possible solution might remain, which leads to a new and expensive development outside of the modelled product knowledge or a refusal of the sales offer. On the other hand, a number of class combinations is still possible, which represents a starting point for the development process based on modelled knowledge. Solving a positive, but not conclusive result requires further, successively added properties. A product developer gradually fills the remaining parameters (see Figure 8) which in turn results in the reduction of possibilities in the ICPMs (see Figures 8, 9 and 10). To support this completion sequence, weighting of all classes according to a standardization key figure that is based on usage frequency, capitalized costs, and a time relevance factor for each property (Konrad et al., 2017) presents a reasonable approach to find a valid product structure.

The before mentioned strict separation of product knowledge an engineered variants takes place after a complete configuration. The development process only uses the modelled knowledge, but does not use already engineered variants. However, all engineered variants are the basis for a periodically verification and possible changes in the modelled product knowledge. The connection, to transfer a created configuration into a PDM test system and into CAD parts is currently under development.

4. Summary and conclusion

The modelling of the product knowledge in a functional and a structure part allows easy adaptability and an immense reduction of maintenance expenses. The description of classes in the RCS, thereby removing engineered variants from the portfolio, reduces the amount of elements and thereby the complexity and maintenance expenses. Through clearer and fewer dependencies the compatibility matrices is able to handle the different classes and allows the splitting into constraint specific matrices. It is thus possible to

https://doi.org/10.1017/dsd.2020.303 Published online by Cambridge University Press

generate customer specific compatibility matrices based on custom input properties and make a statement about the feasibility of the selected properties concerning a functional integral product structure. This removes the instinct to reuse pre-engineered variants and to start a long chain of change iterations. In addition, the graphically supported environment of SysML supports easy generation of the matrices and changes in the structure. The presented PCPM for functional integral products is equivalent to the CPM in regard of its usage. However, the PCPM does not lead directly to a conclusive product configuration as expected with modular product architectures. Nevertheless, the automated check results is a statement over the compliance of customer requirements with the modelled product knowledge. This presents a possible starting point for the product development. Further analysis of the starting point characteristics can support the differentiation and management of the product in further development steps.

References

Anderson, D.M. and Pine, J. (1998), *Agile product development for mass customization. How to develop and deliver products for mass customization, niche markets, JIT, build-to-order, and flexible manufacturing*, McGraw-Hill, New York.

Baumberger, G.C. (2007), "Methoden zur kundenspezifischen Produktdefinition bei individualisierten Produkten", *Dissertation*, Technische Universität München, München. Lehrstuhl für Produktentwicklung. Online verfügbar unter. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20070910-627396-1-3

Blees, C. (2011), "Eine Methode zur Entwicklung modularer Produktfamilien", *Dissertation*. Technischen Universität Hamburg-Harburg, Hamburg-Harburg. Online verfügbar unter. http://tubdok.tub.tuhh.de/handle/11420/1039

Ehrlenspiel, K. and Meerkamm, H. (2017), "Integrierte Produktentwicklung", Denkabläufe, Methodeneinsatz, Zusammenarbeit. 6., überarbeitete und erweiterte Auflage. Hanser, München.

ElMaraghy, H. et al. (2013), "Product variety management", In: *CIRP Annals*, Vol. 62 No. 2, S. 629-652. https://doi.org/10.1016/j.cirp.2013.05.007

Feldhusen, J. and Grote, K.-H. (Hg.) (2013), Pahl/Beitz Konstruktionslehre. Methoden und Anwendung erfolgreicher Produktentwicklung. 8., vollständig überarbeitete Auflage. Springer Vieweg, Berlin, Heidelberg.

Feldhusen, J., Nurcahya E. and Löwer, M. (2007), "Implementation of a product data model to support variant creation process as a part of product lifecycle management", In: *Product lifecycle management : assessing the industrial relevance. Proceedings of the 4th International Conference on Product Life Cycle Management (PLM'07)*, Bd. 3. Italy. Geneve: Inderscience Enterprises Limited (Product lifecycle management - special publication : PLM-SP), pp. 235-242. Online verfügbar unter. http://publications.rwth-aachen.de/record/112672

Firchau, N. and Franke, H.-J. (2002), "Methoden zur Variantenbeherrschung in der Produktentwicklung", In: Franke, H.-J., Hesselbach J., Huch B. und Firchau, N. (Hg.): Variantenmanagement in der Einzel- und Kleinserienfertigung. Mit 33 Tabellen. München: Hanser, pp. 52-86.

Franke, H. J. (1998), "Produkt-Variantenvielfalt - Ursachen und Methoden zu ihrer Bewältigung", In: Verein Deutscher Ingenieure (VDI) (Hg.): Effektive Entwicklung und Auftragsabwicklung variantenreicher Produkte. Allgemeiner Maschinenbau, Anlagenbau, Fahrzeugtechnik, Tagung Würzburg, 7. und 8. Oktober 1998. Unter Mitarbeit von, M. G. VDI: Gesellschaft Entwicklung Konstruktion Vertrieb. Düsseldorf: VDI Verlag (VDI-Berichte, 1434), pp. 1-13.

Friedli, T. and Schuh, G. (2012), Wettbewerbsfähigkeit der Produktion an Hochlohnstandorten. 2. Aufl. Springer, Berlin Heidelberg.

Gausemeier, J. et al. (2015), "Sytems Engineering in Industrial Practice", Heinz Nixdorf Institute, University of Paderborn, Faculty of Product Egineering; Fraunhofer Institute for Production Technology IPT; UNITY AG. Paderborn.

Göpfert, J. (1998), "Modulare Produktentwicklung", *Zur gemeinsamen Gestaltung von Technik und Organisation*, Gabler, Wiesbaden.

Grauer, M. et al. (2010a), "Real-time enterprise -- schnelles Handeln für produzierende Unternehmen", In: *Wirtschaftsinformatik und Management*, Vol. 2 No. 5, pp. 40-45. https://doi.org/10.1007/BF03248290

Grauer, M. et al. (2010b), "Towards an Integrated Virtual Value Creation Chain in Sheet Metal Forming", In: Dangelmaier, W., Blecken, A. Delius, R. und Klöpfer, S. (Hg.): *Advanced Manufacturing and Sustainable Logistics*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 186-197.

Jiao, J., Simpson, T.W. and Siddique, Z. (2007), "Product family design and platform-based product development: a state-of-the-art review", In: *J Intell Manuf*, Vol. 18 No. 1, pp. 5-29. https://doi.org/10.1007/s10845-007-0003-2

https://doi.org/10.1017/dsd.2020.303 Published online by Cambridge University Press

Jiao, J. and Tseng, M. M. (1999), "A methodology of developing product family architecture for mass customization", In: *J Intell Manuf*, Vol. 10 No. 1, pp. 3-20. https://doi.org/10.1023/A:1008926428533
Katzwinkel, T. et al. (2018), MBSE on parameter level. In: NAFEMS (Hg.): *Proceedings: NAFEMS18 DACH Conference. Berechnung und Simulation: Anwendungen, Entwicklungen, Trends. NAFEMS 18 DACH Conference. Bamberg, 14-16.5.2018. NAFEMS. Grafing, Germany: NAFEMS Deutschland Österreich Schweiz GmbH (NAFEMS Proceeding, 18)*, pp. 129-132. Online verfügbar unter. http://publications.rwth-aachen.de/record/723965
Konrad, C. et al. (2019), "Enabling complexity management through merging business process modeling with MBSE", In: *Procedia CIRP*, Vol. 84, pp. 451-456. https://doi.org/10.1016/j.procir.2019.04.267
Konrad, C. et al. (2017), "Varianzsteuerung integraler Produkte durch den prozessbegleitenden Einsatz von Data-Mining Werkzeugen", In: Köhler, P., Brökel, K. Scharr, G. et al. (Hg.): 15. *Gemeinsames Kolloquium Konstruktionstechnik 2017: Interdisziplinäre Produktentwicklung. Duisburg-Essen*, pp. 213-222. Online verfügbar unter. https://publications.rwth-aachen.de/record/707097
Krause, D. and Gebhardt, N. (2018), "Methodische Entwicklung modularer Produktfamilien", *Hohe Produktvielfalt Beherrschbar Entwickeln*, Springer Vieweg, Hamburg, p. 12.
Nurcahya, E. (2007), "Configuration instead of New Design using Reference Product Structures", In: Krause, F.-L. (Hg.): *The Future of Product Development. Proceedings of the 17th CIRP Design Conference*. Springer, Berlin, Heidelberg, pp. 125-134.
Otto, K.N. and Wood, K.L. (2001), *Product design. Techniques in reverse engineering and new product development*. Prentice Hall, Upper Saddle River.
Puls, C. (2003), "Die Konfigurations- und Vertraeglichkeitsmatrix als Beitrag zum Management von Konfigurationswissen in KMU", ETH Zürich, Zürich.
Schneeweiss, D. and Hofstedt, P. (2013), "FdConfig: A Constraint-Based Interactive Product Configurator", In: Tompits H., Abreu, S., Oetsch J., Puhrer J., Seipel D. und Umeda, M. (Hg.): *Applications of Declarative Programming and Knowledge Management. 19th International Conference, INAP 2011, and 25th Workshop on Logic Programming, WLP 2011*. Viena, Austria. Berlin Heidelberg: Springer (Lecture Notes in Artificial Intelligence, v.7773), pp. 239-255. Online verfügbar unter. http://arxiv.org/pdf/1108.5586v1
VDMA (2018), "Das Chinageschäft der Zukunft", Herausforderungen und Strategien für den deutschen Maschinenbau. Hg. v. VDMA. Online verfügbar unter. https://ea.vdma.org/viewer/-/v2article/render/27009603, zuletzt geprüft am 14.06.2019.
Wyrwich, C. and Jacobs, G. (2019), "Branchenübergreifendes Benchmarking von variantenreichen Produktportfolios auf Basis von Produktstrukturen", In: Stelzer, R. H. und Krzywinski, J. (Hg.): *Entwerfen Entwickeln Erleben in Produktentwicklung und Design 2019*. Entwerfen Entwickeln Erleben (EEE). Dresden Technische Universität Dresden. Dresden: TUDpress (Technisches Design, 11,12). Online verfügbar unter. http://publications.rwth-aachen.de/record/767928

https://doi.org/10.1017/dsd.2020.303 Published online by Cambridge University Press