Modularization strategies using multiple product views

Heiner Kesper

1 Introduction

Today's industry is facing a major conflict. On the one hand, increasing numbers of competitors and product substitutes force enterprises to fulfill more and more individual customer demands, resulting in large numbers of variants at small numbers of units. On the other hand, increasing cost pressure and global sourcing require lean and efficient production methods, demanding small numbers of variants at large numbers of units.

A variety of strategies to solve this problem, such as building block, modularization, or platform strategies, have been developed and established in the past. But how to choose a strategy, which is appropriate to the current situation and on the current conditions? How to consider the increasing number of domains integrated into product development processes? And especially, how to adapt it to the wide variety of requirements and constraints originating in these domains? Answering these questions poses a main challenge to most companies, no matter which industry they are operating in.

Aim of the presented research work is to provide an integrated modularization strategy, which focuses on the actual situation and the specific requirements of automotive manufacturers and automotive suppliers. Any such strategy has to reach across product variants and product types to identify carry-over potential, while considering the relevant demands of involved domains.

2 Practical approach to modularization

Modularizing a product means to combine its components into replaceable modules. Such modules possess characteristics that might be turned into advantageous effects. Apart from the situation at hand, it depends on the particular product and the way of modularizing it, whether these characteristics can actually be turned into advantageous effects. Desired effects, however, base on the following characteristics of modules.

- Modules can be developed, manufactured, and supplied separately.
- Modules can be combined into various product variants.
- Modules can be carried over into other products types.
- Modules can be tested separately.

Potentials for modularizing products result from analyzing their structures, i.e. the interdependencies between their components. Apart from strength-based dynamic graphs, the

presented approach utilizes *Design Structure Matrices (DSM)* (STEWARD 1981, PIMMLER & EPPINGER 1994, BROWNING 2001), *Domain Mapping Matrices (DMM)* (DANILOVIC & SANDKULL 2005, DANILOVIC 2006, DANILOVIC & BROWNING 2007), and *Multiple-Domain Matrices (MDM)* (EICHINGER ET AL. 2006, MAURER & LINDEMANN 2007, MAURER 2007) to capture, depict, and analyze those structures.

2.1 Modeling systems

Graphs, like shown in Figure 2-1, depict systems (products) as elements (components) connected by interdependencies (change impacts) and are particularly suitable for visualizing structures, because they represent a system's overall structure in a way that can be grasped intuitively.

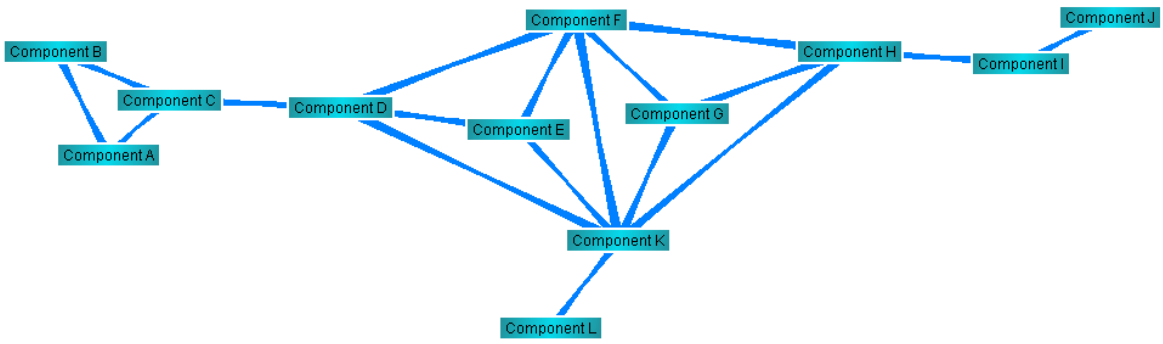

Figure 2-1 Graph representation of an exemplary product structure

Modularization aims at combining a product's components in a way that minimizes interfaces between resulting modules, or generally spoken, to combine a system's elements in a way that minimizes dependencies between resulting clusters (e.g., creating an organizational structure that reduces information exchange between departments). Minimizing interfaces means both minimizing the absolute number of interfaces and minimizing the relative number of interfaces (i.e. the ratio of a module's external dependencies to its internal dependencies).

Of course, not only the (absolute and relative) number of interfaces, but also the importance of interfaces has to be considered to derive practical results. This was effortlessly possible, as the presented analysis process itself is highly systematic and automated. Involving the quality of interfaces (in addition to the quantity of interfaces) in the overall process may, however, be difficult, because it requires assessing dependencies completely and consistently.

Regarding the number of interfaces is, nevertheless, of great importance for several reasons. At first, every interface has to be defined when building a module, or to be more precise, every interface has to be defined *to* build a module. Depending on the circumstances and the type of dependency, defining interfaces has (completely) different meanings. Generally, the number of interfaces connecting a module determines the expenses for defining them. The intended ability to adapt existing modules or to design new modules without causing impacts on the remaining product is another reason to reduce interfaces. As it will not always be

possible to predefine all interfaces, changing module components may still affect product components. The fewer interfaces connect a module to the remaining product, the fewer effects will potentially occur. Apart from that, fewer interfaces result in a stronger canalization of occurring change effects and in an easier handling of them. And even if all interfaces could be entirely defined, it was worth minimizing their number, because every interface means restrictions to the adaptability of a module itself.

The exemplary graph from above supports the claim that graphs are clear system representations, which can be grasped intuitively. Against this, graphs of complex systems may get very confusing. Great numbers of elements and dependencies in extensive and highly connected systems can hardly be distinguished and hence prevent gaining any detailed insights. This is one reason for using an additional method for handling structures, the so-called design structure matrix or DSM.

DSMs depict systems in a matrix alignment. Elements correspond to the rows and columns of a DSM, their interdependencies to the indications in the matrix fields (possible weightings are represented by numerical values). Diagonal elements are faded out, because reflexive dependencies are not intended. Figure 2-2 shows the DSM that belongs to the graph of Figure 2-1.

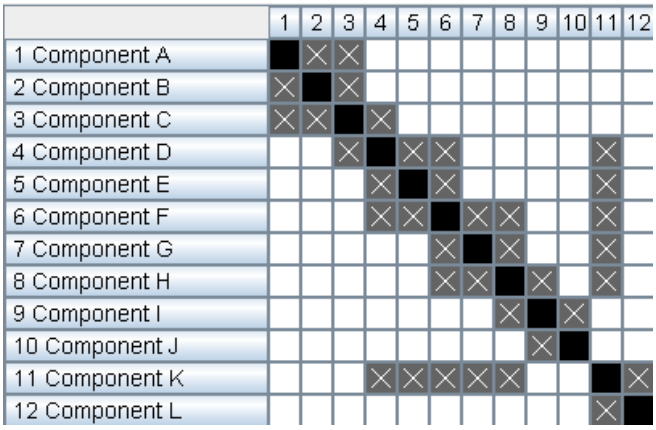

Figure 2-2 Matrix representation of an exemplary product structure

As complex systems may comprise several types of elements, they may have to be represented by various DSMs. A specific type of elements is called a domain. Like explained above, DSMs possess dependencies between elements of a specific type, i.e. elements of a specific domain. Capturing dependencies between elements of different domains requires a second kind of matrix, the so-called domain mapping matrix or DMM.

DMMs are composed similar to DSMs, only they do not have to be symmetric, because the represented domains may include different numbers of elements. Apart from that, they do not possess faded out matrix cells, since reflexive dependencies are precluded by the fact that different domains cannot include identical elements.

All system describing DSMs and DMMs are arranged in an overall matrix, called multiple domain matrix or MDM. MDMs, like depicted in Figure 2-3, represent powerful and comprehensive methods to capture, depict, and analyze complex systems.

	Components	Costs	Persons
Components	…may induce changes of…	…contribute to…	
Costs		…may limit…	
Persons	…design…	…take responsibility for…	…may have to communicate with..

Figure 2-3 Exemplary MDM

2.2 Deriving additional views

A particular type of elements may possess different types of dependencies. A particular type of dependencies, however, may result from different circumstances. E.g., change impacts on the components from the exemplary structure above may result from geometries components share, functions they fulfill, or costs they contribute to. Dependencies due to specific circumstances can either be directly acquired or computed from already captured dependencies due to other circumstances.

No matter which circumstances dependencies originate from, and no matter if they are directly acquired or derived from other dependencies, they should be captured in separate matrices, located in the corresponding sections of a MDM. Capturing dependencies due to different circumstances separately simplifies data acquisition, maintains comprehensibility throughout all project stages, and enables analyzing them separately. It has to be noted that these matrices possess the same elements (e.g., components) *and* the same type of dependencies (e.g., change impacts). Only the dependencies result from different circumstances, or literally spoken, from different *views* on the system.

Experience shows that most views cannot be captured directly. E.g., experts from industrial practice are indeed able to determine, whether dependencies due to geometric constraints exist between product components. But even they are hardly able to tell, whether the same components depend on each other, owing to cost or manufacturing constraints. Capturing dependencies is, furthermore, complicated by the fact that it must be absolutely complete and consistent. It is, however, not necessary to capture all views directly, because many of them can be computed from other matrices, i.e. other DSMs and/or DMMs.

If the underlying information of these matrices can be acquired easily and reliably, views of high data quality can be generated. Figure 2-4 illustrates the basic principle of computing views by an example. If elements of one type (e.g., components) relate to (e.g., contribute to) an element of another type (e.g., costs), they might depend on each other. In this example, components A and B both contribute to cost A and might consequently depend on each other. E.g., modifying component A might increase its component cost. In case a total cost target had been defined for cost A, component B could be adapted to reduce its component cost and consequently balance component costs and reach the predefined cost target respectively.

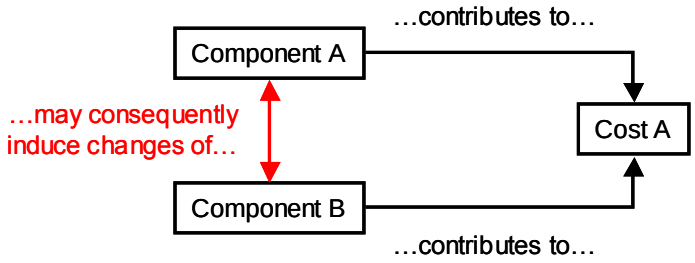

Figure 2-4 Basic principle of deriving change impacts

The initial information for computing alternative views (the allocation of components to costs, in this case) as well as the resulting view (the components' change impacts) is located in the belonging sections of the MDM (see Figure 2-5).

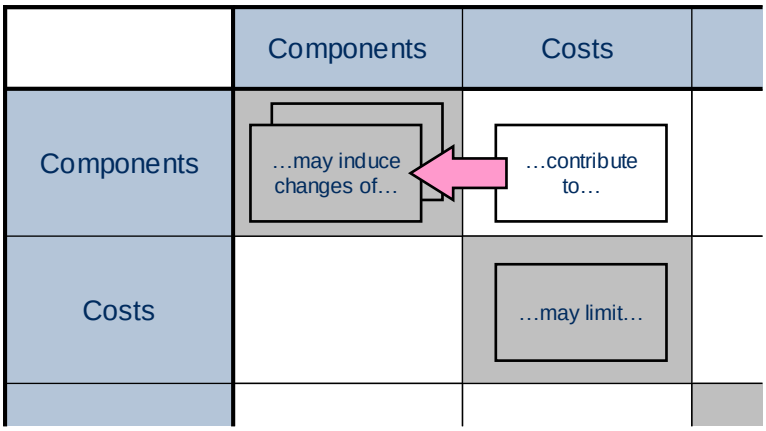

Figure 2-5 Initial information for and resulting information of deriving views in the exemplary MDM

2.3 Single view clustering

Like explained above, modularizing systems means aggregating included elements in an appropriate and efficiently realizable way. The presented approach bases on the analysis of

system structures, i.e. the identification and assessment of structural characteristics. Depending on the regarded system and the desired application, few relevant system characteristics have to be selected. In case of the presented approach, so-called clusters are the only characteristics, which have been considered as being capable for product modularization.

Clusters are groups of elements, which have many internal dependencies and few external dependencies. Modules, on the other hand, are groups of components, which are likely affected by internal changes and unlikely affected by external changes (interfaces). Clusters consequently correspond to potential modules (they are nothing but a more abstract and general description of the same objects).

Thus, systems are modularized by clustering the corresponding matrices, i.e. by systematically rearranging included elements until their dependencies form clusters on the matrix diagonal. Figure 2-6 shows a simple exemplary structure before and after such systematic rearrangement.

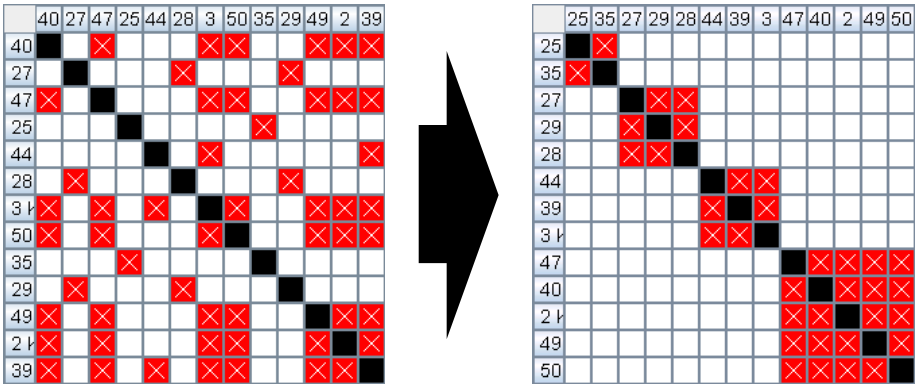

Figure 2-6 Clustering matrices

Clustering matrices bases on established methods, which have been applied to many and diverse problems from theory and practice (see STEWARD 1981, BROWNING 2001, and MAURER 2007). It aims at identifying and evaluating clusters as well as deciding on the suitability and relevance of existing views for upcoming analyses. The detection of clusters itself is an automated process, carried out by the LOOMEO software tool[1]. Combining and splitting up clusters into realizable modules cannot be automated at all, as it is subject to many and diverse practical constraints. The clustering process is consequently a combination of automated and manual methods that depend on each other.

2.4 Multiple view clustering

Exclusively clustering a single matrix or clustering several matrices separately meant exclusively considering a single aspect or several aspects separately. Producing

[1] The LOOMEO software tool is provided by TESEON GmbH

comprehensible and thus practical results, however, means considering relevant aspects simultaneously. In this respect, optimizing product structures could be compared to optimizing a Rubik cube. The problem in optimizing a Rubik cube is to optimize all sides simultaneously. Trying to do so by regarding only one side of the cube will not be successful, because it is highly improbable to optimize the remaining sides by chance. Optimizing each side separately will not lead to any success either, owing to the fact that resulting impacts on other sides can neither be foreseen nor seen. The only way to solve the "Rubik cube problem" is to consider all sides simultaneously. Considering all aspects of a dependency structure simultaneously is reached by, metaphorically speaking, overlying the corresponding matrices and looking through them. Figure 2-7 clarifies this proceeding by a graphical presentation.

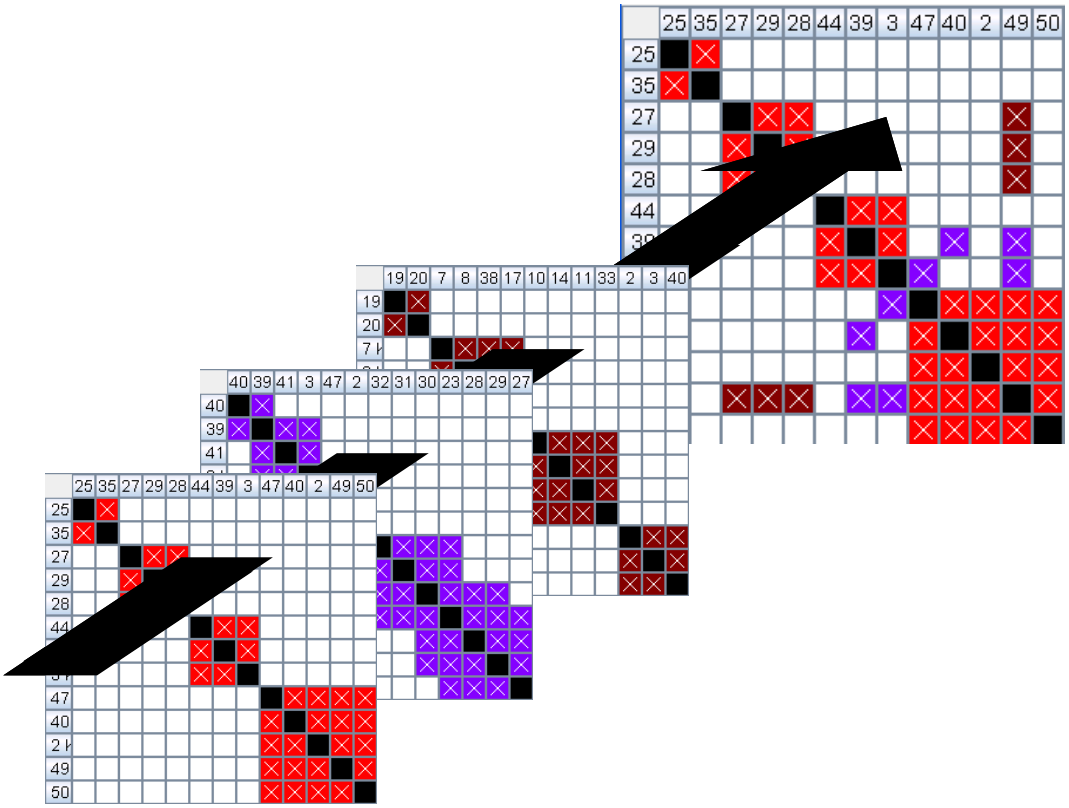

Figure 2-7 Overlying matrices to carry out comprehensive analyses

The resulting aggregated view comprises all aspects of interest and can be clustered in the following. If desired, one of the underlying views can be designated as the initial view, i.e. the order of elements of one view is kept up. As a result, the suggested modules of the initial view remain visible during the overlying and analyses process. Thus, dependencies originating in additional views, which are in conflict with these modules, can easily be identified. These dependencies are the actual objects of interest, because they represent the interfaces of potential modules and have to be controlled to practically implement them.

3 Verification

The presented approach was applied and refined in a consultancy project from industrial practice.

The Diesel Systems (DS) division of Robert Bosch GmbH develops, produces, and distributes Common Rail Diesel Systems. High-pressure pumps represent essential components of these systems, as they are in charge of delivering a required fuel flow at a defined pressure level of currently up to 2000 bar. A variety of additional functions increases the complexity of the pumps, which are mounted to the engines of over 100 OEMs Bosch DS call their customers. Providing individual customer interfaces and especially adapting them to changing requirements make great demands on developing competitive high pressure pumps.

TESEON GmbH supports complexity management by a variety of innovative methods and tools. In cooperation with Bosch DS, TESEON developed a modularization concept, which allows offering customized high-pressure pumps at economically justifiable numbers of component variants.

The project focused on investigating two pump types, which base on partly different concepts. CP1H contains three pistons that are driven by an eccentric shaft and a housing that possesses internal piping of low pressure and high-pressure fuel. CP4.1 contains one piston that is driven by a camshaft and a housing that possesses internal piping of low pressure but not high-pressure fuel. Aim of the project was to derive pump specific and unspecific potentials for modularizations by identifying (groups of) components, which are suitable for being carried over.

Produced analysis results allowed identifying groups of lowly integrated components, which are particularly capable for being combined into modules. Depending on included components, modules can be reused in additional product variants and product types. Apart from that, structurally isolated and lowly integrated single components could be identified. Such components represent potential carry over parts, as they can effortlessly be modified to be transferred to other product variants or product types, too.

Potential modules and carry over parts offer additional degrees of freedom to development, manufacturing, and planning processes, i.e. the responsibilities for modules and parts can be distributed more freely across Bosch's international development and manufacturing network both regarded fuel pumps are integrated into. This opportunity may result in cost savings, quality improvements, and time reductions as well as a more reliable capacity and risk management.

Additional benefit arises from the possibility to test functionally integrated modules separately, which allows identifying errors at an early stage of the manufacturing process. Thus, cheaper sub-assemblies are rejected instead of more expensive end products. This effect is of major importance to Bosch DS, as the great demands on high pressure fuel pumps require testing every single assembly during the production process.

The carried out project could proof the need for comprehensive modularization strategies and the capability of the applied method, as the suggested modules actually depend on the

considered views and combination of views respectively. Derived potentials have been evaluated by experts from the concerned departments and will be implemented in upcoming development projects.

4 Conclusion and outlook

Product differentiation and diversification has been permanently increasing in the past and will be further increasing in future in most industries. It is promoted by opening up new markets, being challenged by new competitors, and facing new requirements. Modularization strategies generally offer a wide range of possibilities to reduce the resulting diversity of variants and to control the complexity it is accompanied by.

At the same time, the complexity of development processes increases, as the number of domains that are integrated into product development and the often conflicting requirements that originate from them increase. Currently existing modularization strategies, however, do not take this fact into account adequately, as they merely aim at aspects of single domains. The presented modularization strategy allows considering aspects from various domains and thus identifying potentials, which do not come into conflict with any well-founded constraints originating in these domains.

The developed approach for modularizing products extends the applicability of existing approaches by integrating established methods of Design Structure Matrices, Domain Mapping Matrices, and Multiple-Domain Matrices. This allows considering aspects from different domains of relevance and hence carrying out comprehensive analyses. The capability of such comprehensive analyses for identifying carry-over potentials in complex product designs could be proven in an extensive use case from industrial practice.

The underlying process is generic and thus suitable for many and divers applications. The application in further projects from different industries represents required future work to confirm present insights and to refine and improve the developed proceeding.

5 References

BROWNING, T. (2001):
Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions.
IEEE Transactions on Engineering Management 48 (2001) 3, pp 292-306.

DANILOVIC, M.; SANDKULL, B. (2005):
The use of dependence structure matrix and domain mapping matrix in managing uncertainty in multiple project situations.
International journal of project management 23 (2005), pp 193-203.

DANILOVIC, M.; BROWNING, T. (2007):
Managing complex product development projects with Design Structure Matrices and Domain Mapping Matrices.
International journal of project management 25 (2007), pp 300-314.

DANILOVIC, M. (2006):
Bring your suppliers into your projects – Managing the design of work packages in product development.
Journal of Purchasing & Supply Management 12 (2006), pp 246-257.

EICHINGER, M.; MAURER, M.; PULM, U.; LINDEMANN, U. (2006):
Extending Design Structure Matrices and Domain Mapping Matrices by Multiple Design Structure Matrices. In: Proceedings of the 8th Biennial Conference on Engineering Systems Design and Analysis (ASME-ESDA06), Torino.
Torino, Italy: ASME 2006.

MAURER, M.; LINDEMANN, U. (2007):
Facing Multi-Domain Complexity in Product Development.
Cidad Working Paper Series 3 (2007) 1, pp 1-12.

MAURER, M. (2007):
Structural Awareness in Complex Product Design.
München: TU, Diss. 2007.

PIMMLER, T. U.; EPPINGER, S. D. (1994):
Integration Analysis of Product Decompositions. In: Proceedings of the 1994 ASME Design Theory and Methodology Conference, Minneapolis.
Minneapolis, USA: ASME 1994.

STEWARD, D. (1981):
The Design Structure System: A Method for Managing the Design of Complex Systems.
IEEE Transaction on Engineering Management 28 (1981) 3, pp 79-83.