NordDesign 2020
August 11-14, 2020
Kgs. Lyngby, Denmark

IDENTIFYING MODULARITY PRACTICES ACROSS MECHANICS, ELECTRONICS AND SOFTWARE

Christoffer Askhøj[1], Martin Løkkegaard[1], Christian Alexander Bertram[1], Niels Henrik Mortensen[1]

[1]*Technical University of Denmark*
chrask@mek.dtu.dk
mloek@mek.dtu.dk
chalbe@mek.dtu.dk
nhmo@mek.dtu.dk

Abstract

In this paper we present five modularity practices across the domains of mechanics, electronics and software deduced from observations from four companies. The practices are made to help product developers of mechatronic products to assist in finding the most efficient modular division of the products. Furthermore, we present a tool to assist in cross-domain modularity decisions and to help developers follow the five cross-domain modularity practices.

Keywords: modularisation, product architecture, mechatronics

1 Introduction

For decades companies that previously were able to deliver mass produced products with little variance have experienced an increase in both globalisation and demand for customised products (Nadadur, Kim, Thomson, Parkinson, & Simpson, 2012; Pine, 1993). Higher demand for variance means development of more solutions and often this is done sequentially, product-by-product, resulting in overlapping solutions and increased complexity. At worst this leads to costs increasing faster than turnover (M. H. Meyer & Lehnerd, 1997; Wilson & Perumal, 2009). Modular product architectures offers a strategy to cope with increasing complexity and make more profitable products (M. H. Meyer & Utterback, 1993; Robertson & Ulrich, 1998; Sanchez & Collins, 2001).
In recent years, several scholars have developed methods and frameworks supporting engineers in developing modular product architectures (de Weck, 2006; Harlou, 2006; Jung & Simpson, 2016; Krause et al., 2014; Otto et al., 2016). The methods have been used on both mechanical and mechatronic products. However, limited focus has been on modularity practices and effects of modularization across the domains of mechanics, electronics and software. The nature of product development in these three engineering disciplines varies quite a lot, in the way, that they use different supporting tools and methods. Therefore, modularity decisions in one domain could have different and maybe negative impact in one of the other domains.

Often product development is divided into silos i.e. mechanical development, software development, etc. Modularity efforts are then driven separate in each silo/domain, or is dominated by the representatives of the strongest domain (Gepp, Foehr, & Vollmar, 2016; Hehenberger, 2014). Practices for developing modularity across domains could help both design engineers and project managers, to avoid conflicting modularity efforts. In this paper, we presents and discuss five cross-domain practices that can help product developers when designing and re-designing for modularity in mechatronic systems. Furthermore, we present a new tool for mapping mechatronic product architectures to show dependencies across mechanics, electronics and software.
The structure of the paper is as follows: First, we present the background theory and the methodology for the research. The section "State of the art" describes the state in the literature. Following this is the findings of cross-domain modularity practices from four companies. Then we present a tool to assist in cross-domain modularity decisions. In the end, we discuss the findings and finalise with a conclusion.

2 Background and methodology

In this paper we use the definition of mechatronic products introduced by Buur (1990): A technology, which combines mechanics with electronics and information technology to form both functional interaction and spatial integration in components, modules, products and systems. Hence, all products that combine physical mechanical and electrical components that are controlled by a software code.

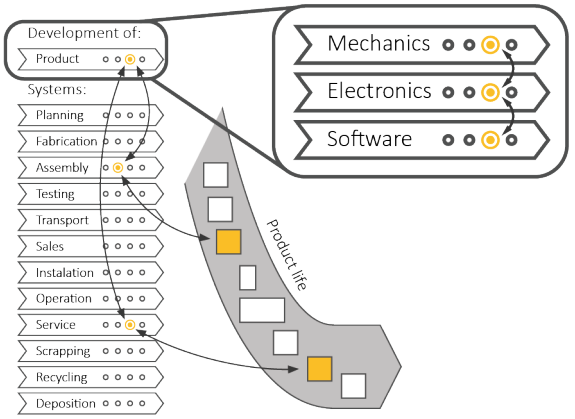

Figure 1: Extension of Theory of Dispositions figure from Olesen (1992)

This research is theoretically founded in the theory of modularisation stating that products can be divided into functional units, called modules, helping a company to increase efficiencies (Marc H. Meyer, 1997; Robertson & Ulrich, 1998; Ulrich & Eppinger, 2012). Essentially the scope of the research can be seen as an extension of the Theory of Dispositions (TD) (Andreasen & Olesen, 1990; Olesen, 1992) in the way that we seek to find practices that deals with effects (dispositions) of modularity decisions taken in one engineering domain into another, see Figure 1. Modularity can be made with different purpose (Erixon, von Yxkull, & Arnström, 1996) and different engineering domains might have different conflicting or non-conflicting inputs to the modularisation strategy. In this research, we cover five key modularity practices that arises when looking across the domains.
The research is exploratory (Karlsson, 2016, Chapter 2) and have been conducted as an action research (Karlsson, 2016, Chapter 7) where the researchers have been observing case companies through close involvement in modularity projects.
Four companies have provided data input for the findings presented in this paper. All have been involved in modularisation research projects with the section of Engineering Design and Product Development (K&P) at the Technical University of Denmark (DTU) over a timespan of two to three years. All companies have been test cases for implementing state of the art tools and methods developed by the K&P section. In this process, the companies have provided data for including: product drawings, sales figures, production setup, organisation structures and cost structures. Meaning that the companies have given full access to their technical, sales and

business data directly from the source being their IT systems. Furthermore, all companies have held weekly status meetings with one of the authors. The cross-domain modularity practices presented in this paper are derived based on insights from these case companies.

2.1 Data input - company descriptions

The four case companies all have different market strategies and product offerings. They range from two small to medium sized enterprises (SME), one in the configure-to-order business and one in Engineer-to-order (ETO) business. Two large companies is included, one mass-production and one ETO company. The products the companies produce range from large one-of-a-kind solutions to smaller technical products, none of which is characterized as consumer goods.
All cross-domain modularisation practices are exemplified with company cases. In all companies, all five practices was observed to some degree, but we only included few examples that display each practice the best.

3 State of the art

The state of the art is divided into two topics: mechatronics and modular product architectures. This research builds on the theory of modular product architectures and therefore the review on this topic is more extensive. The literature that deals with modular architecture development is further divided into 5 groups: product strategy driven models, graphical methods, matrix-based models, mathematical models and modularity frameworks. In each group, we will now discuss how the state of the art deals with architecture development across mechanics, electronics and software and, if any, what effects that might have.

3.1 Modular architectures

Product strategy driven models
The modular function deployment (MFD) methodology developed by Erixon et al. (1996) can be used to systematically develop modular product design. The method does not deal with detailed design and could be used on all sorts of products including mechatronic products as done by Börjesson (2014). The MFD method in its essence does not explicitly deal with any practices of modelling architectures across mechanics and electronics and Börjesson does not directly mention modularity practices across the two domains. In its original form, the MFD methodology does not address relations from mechanics or electronics to the software domain.

Graphical models
Graphical models like the Interface Diagram (Bruun & Mortensen, 2012) or the Product Family Master Plan (PFMP) (Harlou, 2006) are able to address dependencies across mechanics, electronics and software due to the abstraction level the two methods use. Using these models on the three engineering domains of a product could give an overview of differences in modularity. Both methods have been used on mechatronic products, but in their original form do not address dependencies or differences across mechanics or electronics, and the software domain have not to our knowledge been included in their use.

Matrix-based models
The most commonly used matrix-based model in relation to development of modular product architectures would be the Design Structure Matrix (DSM) (Steward, 1981). The DSM have

been used in many different applications including mechatronic products (Alvarez Cabrera, Komoto, Van Beek, & Tomiyama, 2014; Browning, 2016). Algorithms, such as the IGTA algorithm (Borjesson & Hölttä-Otto, 2014), can be used to cluster DSMs and could be used on DSMs that maps products across mechanics, electronics and software to locate modularity synergies across the domains. A certain clustering of a DSM might not be feasible in practice across all three domains, and the task of setting up the DSM extends excessively as the product increase in complexity. Nor is modularity practices across the three domains explicitly presented with DSMs.

Mathematical models
Mathematical models like De Weck's (2006) model for deciding platform extend or the method of Schuh et al. (2017) for contextual design of modular product platforms tends to focus on variants of commercial units within a platform or the extend of a certain platform. They do not explicitly offer any overview of trade-offs between mechanics, electronics and software or describe any practices between the three domains when developing product platforms.

Modularity frameworks
Platform-based frameworks for developing product platforms like the PKT-approach (Krause et al., 2014), the AME (Architecture Mapping and Evaluation) framework (Mortensen, Hansen, Løkkegaard, & Hvam, 2016) or the 13 steps for developing a platform concept by Otto et al. (2016) can be used across engineering domains, and probably should go across domains when developing mechatronic products. However, none of them explicitly covers modularity practices that arise when looking across the three domains discussed herein, nor offer any formal visualisation practices of dependencies across all three domains.

3.2 Mechatronics

Researchers have provided many tools and frameworks for aligning the product development process across the mechanics, electronics and software domains when developing mechatronic products (De Silva, 2005). Welp and Jansen (2004) presented a method for the domain allocation of the functions of products and Hehenberger (2014) gave an overview of a hierarchical process representing different disciplines for the design of mechatronic products. They focus on single product development and not explicitly families of products. Alvares Cabrera et al. (2011) introduced a model, method and tool implementation supporting a corporative design process with exchange of information between domains, but focus more on knowledge management processes than the product development process.
Some researchers have been focusing on modularity within mechatronic products. Schuh et al. (2019) and Schuh et al. (2016) developed methods for designing mechatronic modules based on principles from axiomatic design. They set up axiomatic equations for defining module extend (how much variance can a module cover) and present a four step method for designing mechatronic modules. Their work focus primarily on the development on single modules. Weyrich et al. (2011) presented an approach using DSMs (Design structure matrix) for developing so called solution neutral mechatronic modules that can be used across solutions. However, they do not focus on complete system design.

3.3 State of the art conclusion

No scholars have explicitly presented key practices for modelling product platforms across mechanics, electronics and software. Knowledge of pitfalls and good modularity practices across the domains might be beneficial knowledge when using existing methods, as most products are designed across domains.

Contributions within the field of mechatronic product development, either focus on the process of sharing knowledge between domains or on single products or modules. We see a gap in methods or tools to support product developers when designing modular products on a portfolio level explicitly addressing cross-domain modularity.

4 Cross-domain modularity practices

The five practices presented in this paper were derived from insights from four case companies. They describe some fundamental challenges to consider when working with modularity across domains. However, due to the nature of the research and limitations of the study, more practices might exist. We now present the five cross-domain modularity practices.

Practice 1: Central or decentral electronics

Division into modules does not always follow the same logic in the electronics domain as in mechanics. Electrical signals are often collected into one central PCB (Printed Circuit Board), a so-called I/O board. This means that if an electrical signal from one functional module in the mechanics domain change, the I/O board have to be revised, which again means that an infliction with other modules' design is made. The driver for collecting the signals is cost, but modules that change frequently, might be worth decoupling from the rest, so they can be revised independently. If these modules are not decoupled, development might become slow and an excessive number of variants of these relatively complex components (I/O boards) must be handled.

One company experiences higher efficiency in developing new variants of a frequently changing module, consisting of a user-interface to the product. This module was not connected to the central I/O board. Signals were sent to the main computer through a separated I/O board on the same BUS data-connection as the central I/O board. If the module had not been decoupled from the central I/O board the following complexity would have been introduced:

- More variants of the central I/O (relatively expensive component) would have to be handled and stocked
- The design process would be less efficient because a new layout of the central I/O board would have to be made
- A relatively low selling unit would indirectly inflict design-costs on the top sellers

Practice 2: Same modularity in software and mechanics on critical changing modules

What might seem as small relatively easy-to-handle changes in mechanics or electronics might have big impact in the software domain. If modularity in the software domain follow the mechanical domain in critical areas that are highly coupled changes can be handled more efficiently. One company experienced that the change in the size of a nozzle had big infliction on the control software, because now the dose of water that passes through the nozzle over a certain time is changed. The control software then needed to be calibrated but since the modularity did not follow the logic from the mechanics domain, the impact showed in multiple areas of the software and was almost impossible to manage.

Another company producing processing plants experienced most efficient start-up phases of new plants when the modular structure of software followed the modular structure of the processing plant. The big influence comes when having to tune the plants for operation. If certain modules of the processing plant have to be changed, then the software needs to be changed as well. These changes must be done as efficiently as possible, as the plant cannot run without the controlling software and the longer the start-up phase takes the more it costs. Similar modular structures in the two domains helped fastening up this phase.

Practice 3: Adapting to cross-domain standards

Standards for e.g. testing or certification in one domain, may affect the boundaries for modules in other domains. The functional testing of electrical control units was of the main drivers for cost and lead-time for one mass-producing company. Historically, they tested each new product variant on dedicated equipment. The only way to handle this challenge was to work with modularization across the software-, electronics-, and mechanical domains. To develop a generic testing procedure, and allow test-equipment standardization, a standard testing module had to be defined within the software domain. Within the electronic domain, the design of PCBs had to accommodate a standard test array located at a specific position. Finally, the mechanical domain had to ensure access to the PCB through a standardized interface with a fixed distance and orientation to the test array. Through interface standardization across the three domains, the company reported savings in the range of 30-40% on cost and 40-60% on lead-time for test of new product variants.

Practice 4: Different scaling principles across domains

When scaling up performance of products i.e. increasing capacity or power output, modularity across domains can have a significant impact on the complexity of the job. In one company producing process plants, the engineering department (mechanical domain) drove modularization efforts and thus, the scaling practices for adding extra process equipment to a plant, was relatively well defined. However, when adding extra processes, the integration task within the software- and electronic domains grew almost exponentially, as opposed to the mechanical domain which grew linearly, and was a significant driver of cost and quality issues within the company. This was because modularisation had been optimized only from the perspective of one engineering domain, which lead to some negative trade-offs in the other domains. Had the modularisation strategy been optimized across the domains the company would expect a more linear evolvement of the hours spend on development in software and electronics as the number of process steps increased for the plant, as was seen in mechanical development.

Practice 5: Minimizing variance through the software domain

More of the companies were able to reduce variance significantly in the electronics and mechanics domain. Demands in markets for e.g. differentiation in voltage specification could be handled with regulation of power supply. In this case, it meant reduction of 75% in parts and another company identified a possibility of reducing number of commercial variants in the order of 50% by controlling variance through smart software regulation. If only seen from the mechanical or electronic domain the company would have to introduce commercial variants for each regulation point in the software meaning higher inventory bindings, higher production investments and more production changeovers.

5 Cross-domains architecture tool

We have seen in four different companies that challenges related to design of modularity and design change exists across the domains of mechanics, electronics and software. Streamlining modularity across the three domains is a trade-off between development speed, costs and code efficiency. Electronics is often cheaper to centralize to one PCB when only considering direct costs. However, frequent change in the physical product might overcomplicate the design task with one central I/O board.

Product changes might influence electrical signals that are handled by the software. If the software follows the same modular structure as the mechanics or if there exists a clear overview

of the impact of the change in the code, design changes might be handled more efficiently going from the mechanics to the electronics and finally the software domain.
In Figure 2 we present the MESA (Mechanics, Electronics and Software Architecture) tool. A model that helps designers visualize mechatronic product architectures with focus on identifying cross-domain modularity.
Before going in detail with the tool itself we will link the five cross-domain modularity practices to some requirements for the tool, that have helped in the process of developing the tool.

Link to practice 1: Central or decentral electronics

The choice to centralize or de-centralize electronic signals is highly influenced by costs related to printed circuit boards. The tool should give an overview of these costs. If the tool clearly marked modules that are likely to be changed in the future, it would show the infliction on the signal handling boards (I/O boards).

Link to practice 2: Same modularity in software and mechanics on critical changing modules

The tool should visualize the relations between the mechanics/electronics and software domain, and when locating critical modules that are likely to be changed it will be clear how complex the relations from the mechanic domain into the software domain is, and thereby how aligned the two domains are.

Link to practice 3: Adapting to cross-domain standards

By highlighting modules, signals or code, that needs to follow certain standards, following all relations to the two other domains, could help in visualizing how this standard inflicts the other domains, and if harmonization to this standard across domains should be made.

Link to practice 4: Different scaling principles across domains

Scaling principles could be shown in mechanics in the tool by variance within modules. Then links, made by connecting parts/software in each domains, could give an overview of relations between them and give an overview of where scaling principles influence the other domains and how they are handled in each domain.

Link to practice 5: Minimizing variance through the software domain

In the tool, functional modules with variance in the mechanics domain with no relations to the software domain should be clearly visualized, and these are places to look for possible inclusion of electronic solutions with software-controlled variance instead of mechanical variance.

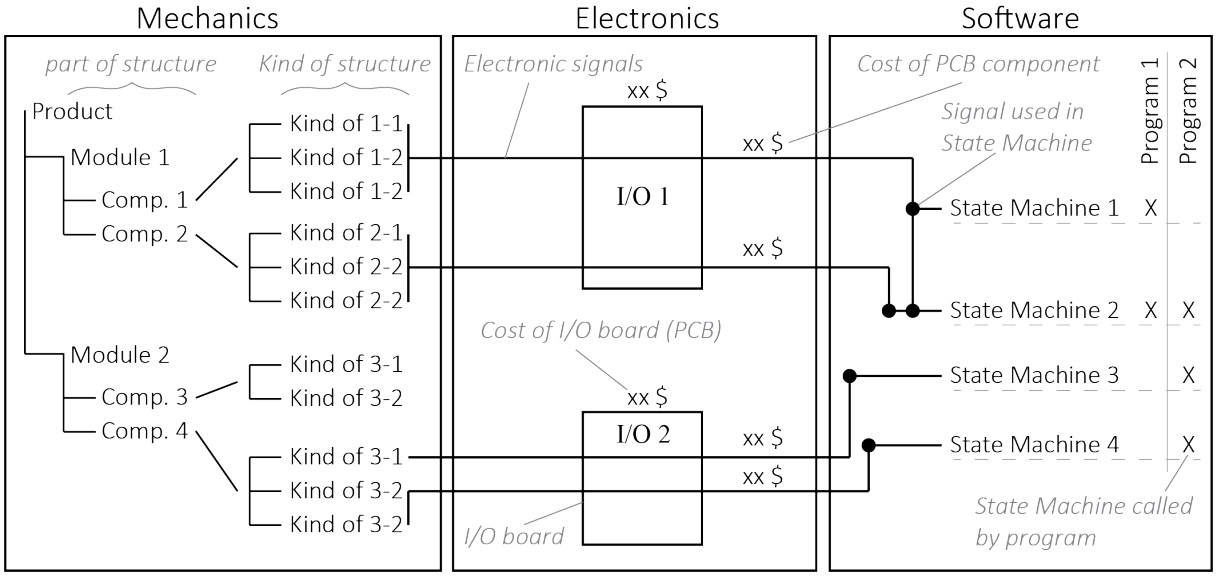

Figure 2. MESA tool (Mechanics, Electronics and Architecture tool)

Besides the modularity practices presented, the tool is inspired by the Product Family Master Plan (PFMP) (Harlou, 2006) and the Interface Diagram (Bruun & Mortensen, 2012). The mechanical platform of the product including part-variance is mapped in the structure shown in Figure 2 to the left in a part of/kind of structure. The idea is then to draw lines from the parts in the mechanical view that either sends or receives electrical signals. The PCB component-cost is noted for each signal and boxes illustrating which I/O board handles the signals are drawn over the lines. With the cost of each I/O board and cost of PCB components, the direct cost of de-centralizing vs centralizing can be estimated. The electrical signals are connected with the software code showing what part of the code handles the signal. In this representation we have used a state machine practice for the code which is a common methodology for making software to control products (Wagner, Schmuki, Wagner, & Wolstenholme, 2006). The structure of the state machines and programs (programs call/activate state machines) presents the architecture of the control code. If the modularity of the mechanics and the software follow each other, you will see a one-to-one mapping of relations from one module in the mechanics domains to one state machine or a few state machines who all handle the same functionality, depending on the complexity of the code.

The function of the MESA tool is to support product developers or managers in giving an overview of modularity and product decisions across mechanics, software and electronics. The input to the model comes from all three development functions and requires a thorough analysis of all the variants of products sold within a product architecture (or similar products architectures). We propose that one person is in charge of filling in product information in the figure to insure consistency in modeling formalism. In addition, this person should have the responsibility of seeking all the product information from experts in each engineering domain. After making a version of the model based on the existing product programme, alternatives can be made to explore how the modularity can be improved across domains.

Experience from the case companies shows that information about cost of PCB components can be difficult to find, especially when third party companies supply the PCBs. Therefore, this information may have to be a best-guess estimate, if the supplier will not deliver cost-breakdowns of PCBs.

6 Discussion

As mentioned previously, existing methods and tools for developing modular product architectures do not explicitly deal with practices for handling modularity across mechanics, electronics and software. Surely, many of the methods are appropriate for developing mechatronic products, however without explicitly concerning cross-domain trade-offs, important synergies or pitfalls might be overlooked, which in the end can sacrifice company earnings.

We did not solve every challenge of developing product architectures across engineering domains, but we have added emphasis on some modularity practices that could be relevant to many companies. Of course, modularity practices between other domains such as manufacturing or supply chain are of high importance when developing modular product architectures. However, work such as Fixson (2005) and Løkkegaard et al.(2018) have already dealt with practices in those domains, and therefore these domains are also not part of the MESA tool presented in this paper.

Regarding the MESA tool, an important task is to find the right level of product breakdown. If the breakdown is too fine-grained, it may compromise the overview. However, the product also need to be broken down at such level that you can distinguish the different electrical signals within each module. This is something that is left to the user to evaluate.

From the observed companies working with modularity we have covered five key practices for developing modularity across the three domains. The practices were deduced from both successful and unsuccessful cases in the companies. We do not claim they represent a complete list of modularity practices across the three domains. However, in the four companies studied in this research they represent the most important practices related to impact on lead-time and earnings. Further research could be made with more companies to discover other practices.

7 Conclusion

In this paper, we have presented five different cross-domain practices that are relevant when designing modular mechatronic products. The practices focus on the trade-offs between the domains of mechanics, electronics and software. The five practices are:

- Central or decentral electronics
- Same modularity in software and mechanics on critical changing modules
- Adapting to cross-domain standards
- Different scaling principles across domains
- Minimizing variance through the software domain

They were deduced by observing four different case companies that have been part of research projects at the section of K&P at DTU over a time span of two to three years.

Building on the observations of modularisation practices across mechanics, electronics and software and methods from other scholars we have also presented the MESA tool for visualizing mechatronic architectures across mechanics, electronics and software. The tool is supposed to assist product developers on modularity decisions by giving an overview of relations between mechanics, electronics and software. The tool still needs testing to validate its ability to help practitioners to follow modularity practices across domains.

In addition, domains such as manufacturing and supply chain might have an effect that could inflict with some of the modularity practices across mechanics, electronics and software. This did not seem to be the case in the companies observed in this paper, but it could be subject to further research.

References

Alvarez Cabrera, A. A., Komoto, H., Van Beek, T. J., & Tomiyama, T. (2014). Architecture-centric design approach for multidisciplinary product development. In *Advances in Product Family and Product Platform Design: Methods and Applications* (pp. 419–447). Springer New York. https://doi.org/10.1007/978-1-4614-7937-6_17

Alvarez Cabrera, A. A., Woestenenk, K., & Tomiyama, T. (2011). An architecture model to support cooperative design for mechatronic products: A control design case. *Mechatronics*, *21*(3), 534–547. https://doi.org/10.1016/j.mechatronics.2011.01.009

Andreasen, M. M., & Olesen, J. (1990). The Concept of Disposition. *Journal of Engineering Design*, *1*(1).

Börjesson, F. (2014). Modular Function Deployment Applied to a Cordless Handheld Vacuum. In *Advances in Product Family and Product Platform Design* (pp. 605–623). New York, NY: Springer New York. https://doi.org/10.1007/978-1-4614-7937-6_24

Borjesson, F., & Hölttä-Otto, K. (2014). A module generation algorithm for product architecture based on component interactions and strategic drivers. *Research in Engineering Design*, *25*(1), 31–51. https://doi.org/10.1007/s00163-013-0164-2

Browning, T. R. (2016). Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities. *IEEE Transactions on Engineering Management*, *63*(1), 27–52. https://doi.org/10.1109/TEM.2015.2491283

Bruun, H. P. L., & Mortensen, N. H. (2012). Visual product architecture modelling for structuring data in a PLM system. *IFIP AICT - Advances in Information and Communication Technology*, *388*, 598–611. https://doi.org/10.1007/978-3-642-35758-9

Buur, J. (1990). *A theoretical approach to mechatronics design*. Technical University of Denmark.

De Silva, C. W. (2005). *Mechatronics : an integrated approach*. CRC Press.

de Weck, O. L. (2006). Determining Product Platform Extent. In *Product Platform and Product Family Design* (pp. 241–301). New York, NY: Springer US. https://doi.org/10.1007/0-387-29197-0_12

Erixon, G., von Yxkull, A., & Arnström, A. (1996). Modularity – the Basis for Product and Factory Reengineering. *CIRP Annals*, *45*(1), 1–6. https://doi.org/10.1016/S0007-8506(07)63005-4

Fixson, S. K. (2005). Product architecture assessment: a tool to link product, process, and supply chain design decisions. *Journal of Operations Management*, *23*(3–4), 345–369. https://doi.org/10.1016/j.jom.2004.08.006

Gepp, M., Foehr, M., & Vollmar, J. (2016). Standardization, modularization and platform approaches in the engineer-to-order business Review and outlook. In IEEE (Ed.), *2016 Annual Ieee Systems Conference (syscon)* (pp. 237–242).

Harlou, U. (2006). *Developing product families based on architectures : contribution to a theory of product families*. Department of Mechanical Engineering, Technical University of Denmark.

Hehenberger, P. (2014). Perspectives on hierarchical modeling in mechatronic design. *Advanced Engineering Informatics*, *28*(3), 188–197. https://doi.org/10.1016/j.aei.2014.06.005

Jung, S., & Simpson, T. W. (2016). An integrated approach to product family redesign using commonality and variety metrics. *Research in Engineering Design*, *27*(4), 391–412. https://doi.org/10.1007/s00163-016-0224-5

Karlsson, C. (2016). *Research methods for operations management*. Routledge.

Krause, D., Beckmann, G., Eilmus, S., Gebhardt, N., Jonas, H., & Rettberg, R. (2014). Integrated Development of Modular Product Families: A Methods Toolkit. In *Advances in Product Family and Product Platform Design* (pp. 245–269). New York, NY: Springer

New York. https://doi.org/10.1007/978-1-4614-7937-6_10
Løkkegaard, M., Mortensen, N. H., & Hvam, L. (2018). Using business critical design rules to frame new architecture introduction in multi-architecture portfolios. *International Journal of Production Research*, *56*(24), 7313–7329. https://doi.org/10.1080/00207543.2018.1450531
Meyer, M. H., & Lehnerd, A. P. (1997). *The power of product platforms : building value and cost leadership*. Free Press.
Meyer, M. H., & Utterback, J. (1993). The product family and the dynamics of core capability. *Sloan Management Review*, *34*(3), 29–47.
Meyer, Marc H. (1997). Revitalize your product lines through continuous platform renewal. *Research Technology Management*, *40*(2), 17–28. https://doi.org/10.1080/08956308.1997.11671113
Mortensen, N. H., Hansen, C. L., Løkkegaard, M., & Hvam, L. (2016). Assessing the cost saving potential of shared product architectures. *Concurrent Engineering Research and Applications*, *24*(2), 153–163. https://doi.org/10.1177/1063293X15624133
Nadadur, G., Kim, W., Thomson, A. R., Parkinson, M. B., & Simpson, T. W. (2012). Strategic Product Design for Multiple Global Markets. In *Volume 7: 9th International Conference on Design Education; 24th International Conference on Design Theory and Methodology* (Vol. 7, pp. 837–848). American Society of Mechanical Engineers. https://doi.org/10.1115/DETC2012-70723
Olesen, J. (1992). *Concurrent development in manufacturing : based on dispositional mechanisms*. Technical University of Denmark.
Otto, K. N., Hölttä-Otto, K., Simpson, T. W., Krause, D., Ripperda, S., & Ki Moon, S. (2016). Global Views on Modular Design Research: Linking Alternative Methods to Support Modular Product Family Concept Development. *Journal of Mechanical Design*, *138*(7), 071101 1-16. https://doi.org/10.1115/1.4033654
Pine, B. (1993). *Mass Customization - The New Frontier in Business Competition*. Harvard Business School Press.
Robertson, D., & Ulrich, K. T. (1998). Planning for product platforms. *Sloan Management Review*, *39*(4), 19–31.
Sanchez, R., & Collins, R. P. (2001). Competing - and learning - in modular markets. *Long Range Planning*, *34*(6), 645–667. https://doi.org/10.1016/S0024-6301(01)00099-1
Schuh, G., Dölle, C., Barg, S., Kuhn, M., & Breunig, S. (2019). Efficient Modular Product Platform Design of Mechatronic Systems. In *IEEE International Conference on Industrial Engineering and Engineering Management* (Vol. 2019-Decem, pp. 1391–1395). IEEE Computer Society. https://doi.org/10.1109/IEEM.2018.8607714
Schuh, G., Riesener, M., Barg, S., & Lauf, H. (2017). Methodology for the contextual design of a modular product platform concept. In *21ST INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED17* (Vol. 3, pp. 1–10). the Design Society.
Schuh, Guenther, Rudolf, S., & Breunig, S. (2016). Modular Platform Design for Mechatronic Systems using Axiomatic Design and Mechatronic Function Modules. In *Procedia CIRP* (Vol. 50, pp. 701–706). Elsevier B.V. https://doi.org/10.1016/j.procir.2016.05.035
Steward, D. V. (1981). DESIGN STRUCTURE SYSTEM: A METHOD FOR MANAGING THE DESIGN OF COMPLEX SYSTEMS. *IEEE Transactions on Engineering Management*, *EM-28*(3), 71–74. https://doi.org/10.1109/TEM.1981.6448589
Ulrich, K. T., & Eppinger, S. D. (2012). *Product design and development*. McGraw-Hill/Irwin.
Wagner, F., Schmuki, R., Wagner, T., & Wolstenholme, P. (2006). *Modeling software with finite state machines: A practical approach*. *Modeling Software with Finite State Machines: A Practical Approach*. CRC Press. https://doi.org/10.1201/9781420013641
Welp, E. G., & Jansen, S. (2004). Domain allocation in mechatronic products. In *Design 2004:*

Proceedings of the 8th International Design Conference (pp. 1349–1354).
Weyrich, M., Klein, P., Laurowski, M., & Wang, Y. (2011). *A function-oriented approach for a mechtronic Modularization of a sensor-guided Manufacturing System*.
Wilson, S., & Perumal, A. (2009). *Waging war on complexity costs*. McGraw-Hill Education. https://doi.org/10.1038/nsmb.3375