Cross domain modularization tool: Mechanics, electronics, and software

Askhøj, Christoffer; Christensen, Carsten Keinicke Fjord; Mortensen, Niels Henrik

Published in:
Concurrent Engineering: Research and Applications

Link to article, DOI:
10.1177/1063293X211000331

Publication date:
2021

Document Version
Peer reviewed version

Cross Domain Modularization Tool – Mechanics, Electronics, and Software

Christoffer Askhøj[1], Carsten KF Christensen[1] & Niels Henrik Mortensen[1]

[1]Section of Engineering Design and Product Development, Department of Mechanical Engineering, Technical University of Denmark, Kongens Lyngby, Denmark

1 Abstract

Modularization is a strategy used for handling the demand for external complexity with less internal complexity, which leads to higher profits and more efficient product development processes. However, modularity is often driven in silos, not crossing into the engineering fields of mechanics, electronics, and software. Therefore, we present the MESA (Mechanics, Electronics and Software Architecture) tool – a tool that can be used to visualize modular product architectures across mechanics, electronics, and software. The tool demonstrates how a change in one domain affects the rest and how well aligned the modularity in the different domains is. The tool has been tested in two case companies that were used for case application and has helped provide information for making key design decisions in the development of new product families.

Keywords

Product architecture, modularization, concurrent engineering, product platform, cross domain, mechatronics, architecture strategy

2 Introduction

To a great degree, today's markets demand customization (Nadadur et al., 2012). This has led to the development of larger product portfolios. Often, the development of products is performed sequentially, product by product, resulting in increasing internal complexity, overlapping solutions, and a lack of resources to develop new solutions with a competitive edge (Meyer and Lehnerd, 1997; Wilson and Perumal, 2009). The modularization of products offers a strategy for coping with the increased internal complexity caused by an increased demand for external complexity (Meyer and Utterback, 1993; Robertson and Ulrich, 1998; Sanchez and Collins, 2001).

Over the past 30 years, a significant amount of research has been conducted within the area of modularity (Gauss et al., 2020; Jiao et al., 2007; Otto et al., 2016; Pirmoradi et al., 2014). Scholars have developed frameworks and tools for supporting the development of modular products. However, multiple studies of literature in the field show that the focus on supporting modularity decisions going across domains has been limited for the engineering domains of mechanics and electronics (Gauss et al., 2020; Jiao et al., 2007; Otto et al., 2016; Pirmoradi et al., 2014). Product development in companies is often divided into silos, and efforts to maintain modularity are carried out separately in each domain

or are dominated by the domain with the highest influence or power to take decisions (Gepp et al., 2016; Hehenberger, 2014). The division of development activities between domains increases the risk of introducing contradictory modular designs, thus compromising development and cost efficiency. Hehenberger (2014) put emphasis on the importance of looking across domains when designing mechatronic products, and this need does not decrease when designing for modularity in product families.

When designing product systems and families, the field of concurrent engineering define the tools and processes used to coordinate the product development activities across multiple domains and departments (Prasad, 1996, 1997; Stjepandić et al., 2015). Modularization and modular product architectures act as enablers for concurrent engineering because these help clearly define design boundaries. The decoupling of modules and freezing of interfaces enables the concurrent development of different modules (Prasad, 1999; Stjepandić et al., 2015). If the definition of architecture and its modules extends across mechanics, electronics, and software, this will further enhance the possibility of product developers working concurrently.

The MESA (mechanics, electronics, and software architecture) tool that is presented in the current paper has been developed for mapping modularity dependencies across the engineering domains. In this paper, we test the following hypothesis: *The MESA tool can be used for mapping and communicating modularity dependencies across the engineering domains of mechanics, electronics, and software*. If the MESA tool is able show modularity dependencies across domains, it can be used for input to the improvement of the modular design of product families.

The idea for the tool was first—briefly and with a high level of abstraction—presented by Askhøj et al. (2020). It has been further developed and tested in two case companies that were used for case applications, which we present in the current paper. The structure of the paper is as follows: First, we present the background and research methodology (Section 3), which is followed by the related research in Section 4. In Section 5, we present the MESA tool, and, in Section 6, we describe two applications of the tool in the case companies. Finally, the study culminates in Section 0, which contains a conclusion and discussion, as well as a look at further research avenues.

3 Background and research method

The present research is an extension of the theory of modular product architectures, which states that the internal efficiency related to cost, development speed, and complexity handling can be increased by dividing products into functional units (modules) with stable interfaces (Meyer and Lehnerd, 1997; Robertson and Ulrich, 1998; Ulrich and Eppinger, 2012). A modular product architecture is a strategic decision regarding how to design a portfolio of products (Sanchez, 2013). More precisely, we refer to the work of Mortensen et al. (2016), who defined a modular product architecture as having the following characteristics:

- "[It has] shared core interfaces."
- "Core modules/systems exist in balanced performance steps."
- "The architecture is explicitly prepared for derivative products, and related properties in terms of cost and performance are known."

We use the definition of modularity given by Askhøj and Mortensen (2019), which is based on what Sanchez (2010) called strategic modularity. Modularity is "a one-to-one mapping of a function that is perceived by the customer to be an important source of differentiation onto a single component or subsystem of a component (Sanchez, 2010). Interfaces are designed so that a functional module can be swapped without having to compensate in other areas of the product."

Another important term used in relation to modular product architectures that is also used in the current paper is "product family." A family of products is a set of individual products that share common technology- and address-related market applications/segments (Meyer and Lehnerd, 1997: xi).

The research made in this paper has been conducted in four phases including study of the problem, developing the tool, testing the tool in practice, and evaluation of the usefulness of the tool. The four phases are now described.

3.1 Research Phase 1 – Study of the problem

In Phase 1, modularity practices across mechanics, electronics, and software were studied among four companies from different industries, all of which were producing mechatronic products. This work was published at the NordDesign 2020 conference (Askhøj et al., 2020). The four authors worked in close collaboration with the companies over a period of three years; the authors contributed new methods for analyzing existing product portfolios and developing new modular product architectures while being an integrated part of the development team. From observing the mainly

negative effects from design decisions in one domain onto the two other domains, it became apparent that the companies would benefit from mapping cross-domain modularity in a visual way to increase the understanding of the effects decisions in one domain have on the other. Therefore, it was decided to look into the development of a tool for showing cross-domain modularity dependencies.

3.2 Research Phase 2 – Developing the tool

The purpose of the tool is to help product managers in the design planning process and create awareness of modularity decisions among engineers across engineering domains. It can support presentations at meetings with product and portfolio managers for planning design activities at the portfolio level. Therefore, we find the visual attributes of the tool important. Here, the product family master plan (PFMP) (Harlou, 2006) is a visual way of representing all variants in a family of products. It has already been tested for its ability to communicate the physical composition of a family of products. Therefore, the PFMP practice is used for mapping the mechanical structure in the MESA tool. The interface diagram method by Bruun et al. (2014) is used for visually showing how components inside and across modules are connected, including flows of, for example, energy or information. This tool is also tested for its ability to communicate dependencies in modular product architectures. Our idea is to combine these two practices in one tool, where we add electronic signal handling components, and the software modules that control the product. Signal handling components were in the four companies studied in Phase 1 all composed of PCBs (printed circuit board). The three domains are then connected with lines, as is done in the interface diagram, which represents electrical signals and information flow. A representation of this tool can be seen in Figure 1.

3.3 Research Phase 3 – Testing the tool

Phase 3 included the application of the MESA tool in the two company cases. Two researchers (one for each case) from the Technical University of Denmark (DTU) applied the tool in close collaboration with one technical expert from each domain in the company: a mechanical engineer working with product design, an electrical engineer designing the electrical system of the products, and a software engineer writing the control code. The domain experts all had over five years of experience and provided all the relevant information and documentation required for setting up the tool, and meetings were held to verify that the model was correct. Relevant information included BOMs (bill of

materials) for the variants within a product architecture, 3D models to show how parts are connected and the modular structure of the design, electrical wiring diagrams for showing information flow and electrical input/output handling, and the software control code from which inputs and outputs from software modules were used. It took approximately three weeks of full-time work for one engineer to set up the tool, involving one expert with over five years of experience from each domain in meetings that took place two times a week. In both companies, PFMPs and interface diagrams already existed of the product family under scope, together with an overview of the inputs and outputs in all the software modules. If these overviews were not already made, the task of setting up the MESA tool would be more daunting. Based on the time it took to set up the three other overviews used, it is estimated that it would take approximately two to three months without the overview, full time work of one product developer with the assistance of domain experts. If a company never experienced problems with cross-domain dependencies in the design phase, or if no plans for revising the product design is planned it will probably not be useful to use the MESA tool. In addition, the MESA tool is aimed for the analysis of product families rather than single products development. When the MESA tool was set up, it was presented to product management.

The two companies used for the case application were selected because both carried out modularity projects in collaboration with the section of engineering design and product development at DTU to test new methods and tools for design modular product families. The companies are from different industries. Company 1 is a medium-sized (Muller et al., 2018: 13) enterprise operating in a configure-to-order (CTO) business, while Company 2 is a large engineer-to-order (ETO) enterprise delivering systems for larger systems. Both companies design mechatronic products and conduct in-house development for all three domains (mechanics, electronics, and software).

3.4 Research Phase 4 – Evaluation of the tool

In the final research phase, the tool was evaluated by employees from one of the case companies. The employees were presented the tool, which they helped to set up themselves by giving input for all the domains. Afterwards, they were asked to answer questions related to the value of this new tool. This is described in more detail later.

In the other company, a process of restructuring the organization (management layers and department compositions and responsibilities) that happened after the tool had been implemented made it impossible to conduct a similar evaluation because the people who had helped to set it up were now unavailable.

4 Related research

We have divided the related work into three sections: modularization, mechatronics, and concurrent engineering (CE). First, we discuss modularization, which is the research field that is the most closely related because this research expands on the methods from modularization. Both mechatronics and CE offer frameworks for optimizing the flow in the process of developing mechatronics products together with tools to assist the actual engineering design process.

4.1 Modularization

The tool presented in the current paper is inspired by the PFMP (Harlou, 2006) and the interface diagram (Bruun et al., 2014). In their original forms, one can use these two tools to map the dependencies between the mechanics and electronics domains. However, the PFMP and interface diagram do not include the software domain, and there is no explicit indication of what belongs to the mechanics or electronics domain. These aspects have been addressed in our consolidated extension.

The mapping of product architectures with the interface diagram, and thereby also the MESA tool, find inspiration from the theory of technical systems (Hubka and Eder, 1988), and the generalized model for presenting systems introduced by Hubka and Eder (1988).

Interfaces are of high importance when designing modular product architecture and technical systems and should be subject to explicit attention (Parslov and Mortensen, 2015). Attention is put to the cross-domain interfaces by the MESA tool. Pimmler and Eppinger (1994) defined a functional interface as: “functional transfers of material, energy, information, and spatial relations”(Parslov and Mortensen, 2015: 191). We use this definition when looking at the interfaces across the three domains.

The design structure matrix (DSM), which was first introduced by Steward (1981), is a frequently used tool for assisting modular product design. DSMs have previously been used when designing modular mechatronic products (Alvarez Cabrera et al., 2014; Browning, 2016). However, the software domain is not included in a traditional DSM. For that purpose, one could use the multidomain

matrix (MDM) (Eppinger and Browning, 2012: 8). Uddin et al. (2016) indirectly includes all three domains in what they called a function-state MDM (FS-MDM), even though they did not directly include the associated software. MDMs could be used for mapping dependencies across mechanics, electronics, and software, including the code. Another DSM-based approach used for predicting design change impact across the product is the change propagation method (Clarkson et al., 2004). The work of Clarkson et al. did explicitly include the software domain. This could be included, though, by using some techniques used in an MDM. However, using matrices for designing modularity is not very intuitive or visual when it comes to communicating one's findings to product management. Several authors have reported success in using more visual tools in supporting decision makers (Bruun et al., 2014; Krause et al., 2014; Kvist, 2010; Mortensen et al., 2011; Pedersen, 2010). Furthermore, when products increase in complexity, the clustering of MDMs is most efficiently carried out using clustering algorithms (Otto et al., 2016). The algorithms may give solutions that are not realizable and that become complex to set up when multiple domains are included.

Other researchers focus on the use of indices for improving modular designs of product families (Hölttä-Otto et al., 2012; Simpson et al., 2012). Martin and Ishii (2002) introduced the generational variety index (GVI), a measure for the amount of redesign effort required for future design, and the coupling index (CI), a measure of the coupling among components. Such indices could be used for improving modular design across domains by including cross-domain relations. However, using indices for improving modular designs lack the visual attributes sought by the MESA tool.

Other researchers have developed and tested extensive frameworks/methodologies for developing modular product families. An early contribution was the modular function deployment (MFD), which introduced 12 module drivers (Erixon et al., 1996). When introducing the methodology, a mechatronic product served as the example, and the MFD helped define a strategy for how to divide a product family into modules. Since then, several other researchers have developed frameworks for designing modular product families. Krause et al. (2014) introduced what they called the PKT approach, a methods toolkit for designing modular product families where they present the MIG (modular interface graph). A visual tool for mapping the modular structure of a product architecture and the interfaces. The tool does not include the software domain though. The MIG is also used in Otto et al.

(2016) 13-step methodology for designing a platform concept. Løkkegaard et al. (2018) defined what they called business critical design rules (BCDRs), which comprised a framework for defining a set of "game rules" when introducing new products to assess their impact on product and manufacturing architectures. The visualization of dependencies across the product and manufacturing domain is a key feature of the framework, but they do mainly focus on physical attributes of the products. On a more strategical level, Mortensen et al. (2016) presented the architecture mapping and evaluation (AME) method for calculating the impact of modularity decisions. Decisions that can be supported by the visualization of cross-domain dependencies be the MESA tool.

We have investigated the literature reviews on modularization (Bonvoisin et al., 2016; Fixson, 2007; Gauss et al., 2020; Gershenson et al., 2004; Greve and Krause, 2018; Jiao et al., 2007; Otto et al., 2016; Piran et al., 2016; Pirmoradi et al., 2014; Simpson, 2004). By doing so, we have not been able to find any methods or tools that, in a visual way, are well suited for product management involvement and present cross-module dependencies across the engineering domains of mechanics, electronics, and software.

4.2 Mechatronics

Researchers have provided numerous tools and methods for assisting in the development of mechatronic products (Buur, 1990; De Silva, 2005). With a focus on the design process, Hehenberger (2014) presented an overview of a hierarchical modeling technique to assist in the design of mechatronic products. Hehenberger highlighted that the level of granularity when describing product properties and characteristics is important when using design models. If the product information is too accurate and detailed on a systems level, the drawback of, for instance, the modeling effort may be higher than the benefits of having all this information available. This puts an emphasis on the need to consider the level of detail/granularity when using the MESA tool. More considerations are described in Section 4.

Alvarez Cabrera et al. (2011) presented an architecture model to support the cooperative design of mechatronic products but focused mostly on the process of sharing knowledge between the different engineering domains. Certain researchers, such as Welp and Jansen (2004), have worked with methods used for domain allocation for the different functions in mechatronic products. The focus of most mechatronic methods and frameworks, though, has been on single product design.

Schuh et al. (2016) and Schuh et al. (2019) united the fields of mechatronics and modularization; they developed methods for designing mechatronic modules based on axiomatic design. Weyrich et al. (2011) presented an approach using DSMs for developing what they referred to as solution-neutral modules that can be used for multiple solutions. The focus of these methods is on single modules, and they do not directly support the development of the portfolios of products.

Helbig et al. (2016) presented a model-based method for improving domain collaboration across mechanics, electronics, and software in the design phase of special purpose machinery. The focus is on sharing design information during the design of new products. It does not include the analysis of existing products and focus on the development on single products.

4.3 Concurrent engineering

The field of CE deals with the processes used for developing different parts of products simultaneously. The knowledge regarding how different modules in modular products influence each other is one of the success factors for concurrent development in such products. Prasad (2016) presented an icon-based flowcharting methodology for mapping the concurrency of tasks and activities during product development. Zouari et al. (2015) presented a method for managing domain knowledge versioning in the product development phase so that the evolution of knowledge in a development project can be captured and shared across domains. Mougin et al. (2015) proposed a framework for modeling knowledge transfer in projects with distributed design teams. These are all methods that could be used with the MESA tool for assisting in the concurrent development of modular products, but they do not explicitly support the cross-domain design of modular product families.

4.4 Conclusion on related research

Exiting literature offers several methods and tools for improving modular design, but lack the attributes of visually presenting product architectures when including all three domain of mechanics, electronics, and software. Tools have been developed outside the field of modularization for mapping or communicating dependencies across the three domains, but focus on the analysis of single products and not families of products, or do not include the visual attributes sought with the MESA tool. This is what we attempt to cover with the introduction of the MESA tool. The visualization of cross-domain modularity dependencies for families of products.

5 MESA tool for mapping cross-domain architectures

In Figure 1, a representation of the MESA tool with an example of a clock is shown. The general idea is that each domain is mapped in the same figure, with the electronic in the center, the mechanics to the left, and the software to the right. The domains are connected using lines that represent electrical signals. The arrows on the signals represent the direction in which the signal goes. A signal going out of the software domain is usually used to control the electrical part (often motors) in the mechanical domain, and signals going into the software domain are constituted of information from sensors and the like.

The mechanics domain consists of physical parts, excluding input/output components like PCBs. All variants of a product family are mapped. Different variants of the parts or modules are shown within the kind-of structure, see Figure 1. All product variants within the family of products investigated are represented in the tool. As in the PFMP (the "part view", showing all physical parts), the product is decomposed into module areas, which are further decomposed into submodules or components.

In the electronics domain, all signal-handling (input/output) devices are mapped. All input/output components (often PCB boards) are represented by rectangles. Different boards are shown in different columns, and all signals that go across the board are also physically handled by this board. If some signals from the mechanics domain are not passed through a board, this space is left blank in the corresponding column.

The software domain consists of the associated code used to operate the device. The code is decomposed into chunks of code (modules) that handle certain functionalities. A detailed investigation of the input/output parameters used in the code is necessary to know which signals are processed where. The code processing the signals can both be located on the central computer or directly on the input/output component; in the latter case, the code can be more rigid to change, which was the case in the two case applications.

The MESA tool does not support a functional analysis and functional decomposition of the products. The product family under investigation needs to be decomposed from the top down to a level where all signals to and from the electrical components within a module are separated so that it is possible to find a specific parameter that has a cross-domain impact if changed. There is also the

possibility of only decomposing/separating the signals to a module (the connecting lines in the tool) and only decomposing the product to a level where all functional modules are shown. This, however, requires an overview of all the electrical components that send and receive signals somewhere else. In the current paper, the authors have used the PFMP for decomposing the product families. Another proven methodology is dissecting the products in a family and capturing it decomposition level by level in a table, as in Thevenot and Simpson (2006). The MESA tool is scoped for creating awareness of cross-domain dependencies in the process of redesigning existing modular product architectures. Integrated product architectures are not within the scope of this work.

When the interdependencies of a product are high, this means that a design change has an impact on several other components than on the one that was actually intended to be changed. Modularity seeks to minimize these interdependencies across modules (Gershenson et al., 2003). The main purpose of the MESA tool is to illustrate these interdependencies across domains because doing so will indicate the cascading effects caused by a design change in one domain for the other domains. Later, we present real case examples of such effects through the application of the tool.

The level of granularity or decomposition level is, as mentioned in the "Related Work" section, important to consider when using the tool. Going into too much detail (a high decomposition level) may compromise the provided overview, and one could end up drowning in the details. On the other hand, too low of a decomposition level may result in the occurrence of certain phenomena, such as software redundancy or interdependencies not being displayed, which happened in the application of the tool in Case 2 (see the subsection titled "Observations using the MESA tool").

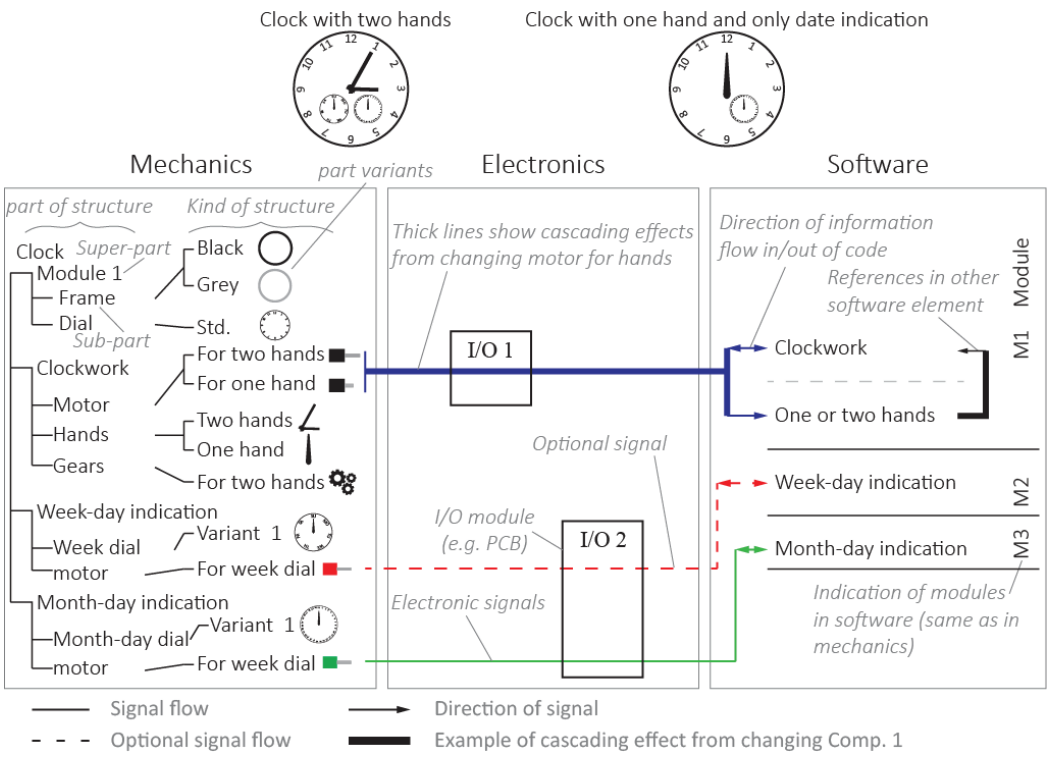

Figure 1: Generic representation of the MESA tool

6 Case applications of the MESA tool

Now, the two applications of the tool are presented.

6.1 Case 1

The company selected for the first application is a Danish small-to-medium-sized enterprise (SME). The company sells approximately 3,000 units per year that are configured when ordered, and its annual revenue is in the order of 20 million EUR. It is a 50-year-old company that has built up an extensive portfolio of hundreds of commercial product variants. The products are mechatronic products sold in a business-to-business context.

Figure 2 shows the MESA tool that was applied for Company 1. The figure only shows a section of the map, including six top-level modules and two submodules. The circles with letters inside them all indicate observations that will be described after the two cases have been presented. In its full form, the map includes 25 physical modules and 56 state machines (state machine is a method for writing the control code, and is described later) compared with the six modules and 25 state machines included in the current paper.

Six modules are identified in the mechanics domain. The kind-of structure (see Figure 2) used shows the variations in the components of different product variants. The electronics domain is represented via input/output (I/O) boards. These I/O boards are PCBs that handle signals to and from sensors and electrical components and that communicate the signals to and from a computer. Three I/O boards are shown in Figure 2: a gas I/O and a main I/O. The optional signals that are added based on the configuration of the product are shown with dotted lines. The software domain uses a state machine practice to write the control code. Using state machines is a common methodology for executing control software for mechatronic products (Wagner et al., 2006). State machines can be activated, and based on the inputs they get from, for example, signals from sensors and calculations, they can obtain a state that can be used by other state machines or that can send a signal to an electrical component. A state machine is comparable to what we call modules in the mechanics domain. Further, as seen in the map, some state machines handle the same functionality, and together, they constitute a higher-level module. The figure also includes how state machines communicate with each other, illustrating how different functionalities in the code are coupled. This allows the user to track how changes cascade through the code.

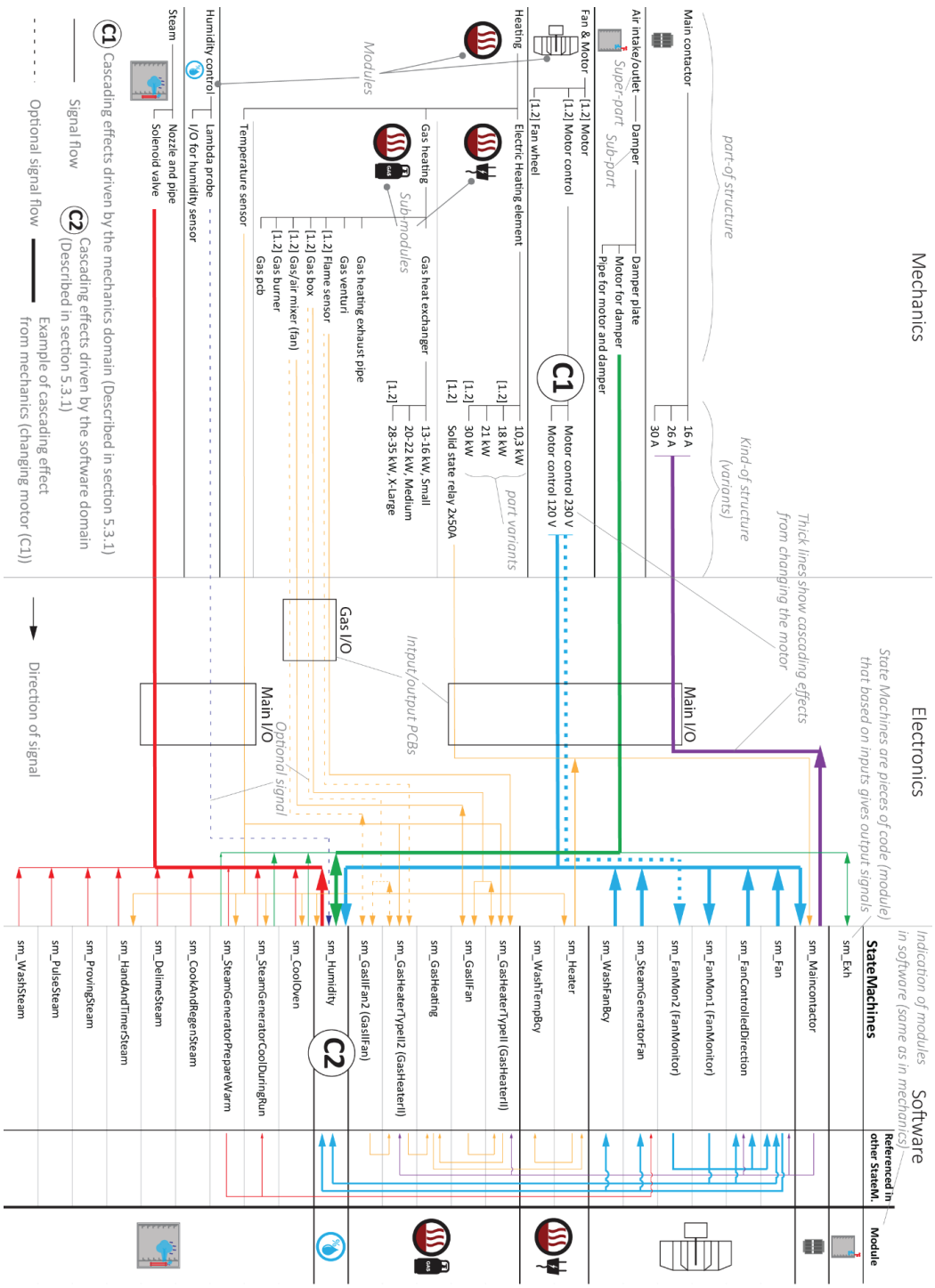

Figure 2: The MESA tool applied for Company 1.

6.2 Case 2

The second case is of a business unit within a large industrial company. The business unit has a turnover of 80 million EUR on average and sells, apart from spares and smaller projects, around four big projects annually as an ETO company. Because of this strategy, their products evolve based on a combination of customer orders and traditional product development. The products are mainly mechatronic and sold to other companies or governments. During the project, the engineers were organized into departments corresponding to the domains, that is, mechanics, electronics, and software. Therefore, the connection between the domains and interaction was of interest. As a part of a bigger analysis of the business unit's products and projects, the MESA tool was used to highlight how customer variation affected the products.

Figure 3 displays a simplified form of the MESA tool's application; it was not possible to show the actual version because of confidentiality restrictions. The level of abstraction compared with Case 1 is equal in the mechanical domain but higher in the electronic and software domain because the controller consists of several I/O boards and PCBs. Each software module consists of numerous state machines.

In Case 2, six mechanical modules are shown, and these are divided into two part-of components (subcomponents). The first part of the component represents the sensor and mechanical interface for the sensor. The other part-of component is the structural assembly of the module. The kind-of structure represents the mechanical variants from one customer project to the next. A single controller represents the electronics domain. This was designed to handle high degrees of variation in terms of interfaces and, therefore, does not change. The software domain is divided into modules, and the interfaces between them are illustrated using the lines on the far right in Figure 3.

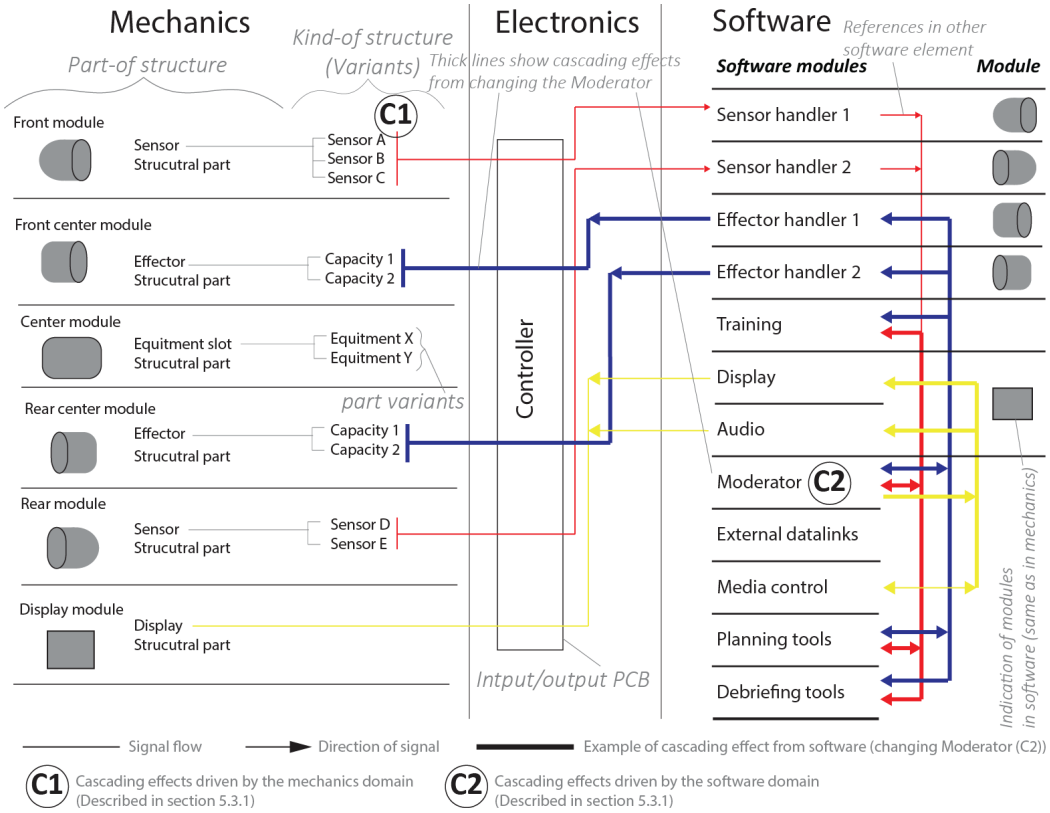

Figure 3: The MESA tool applied for Company 2.

6.3 Observations using the MESA tool

By using the tool in the two cases, we observed the cascading effects. These effects could be seen when observing the impact of a planned design change in the tool. The cascading effects in the two cases were driven by changes in either the software or mechanics; therefore, we divide this phenomenon into two (C1 and C2) when presenting examples from the case application.

6.3.1 Cascading effects

(C1) Cascading effects driven by the mechanics domain

A cascading effect caused by the mechanics domain is the scenario in which a change in one component/module cascades through the system and ends up affecting several other modules in the other domains.

Figure 2 (C1) shows such effects for Case 1. The company considered changing the fan motor control to a different type that uses different signals (in the figure, the cascading effects of this change are marked using thick lines). This would directly affect the eight state machines that are used to control or monitor the motor control. Two of the affected state machines control three other modules, so the

effect of the change cascades from the mechanics into the software and back to three other modules in the mechanics. The changes will also affect almost all modules in the electronics because the components that are changed have signals connecting to the main I/O.

Regarding Case 2, Figure 3 (C1) shows one example of the cascading effects: the changing of a sensor or its position affects the mechanical module and Sensor Handler 1 (software), both of which are supposed to control the sensor's function directly. These cascading effects were seen in cases where the changes cascaded from the sensor handler into four other modules, which were larger and more complex in terms of the number of lines of code; this led to extremely high costs for the company. Additionally, in certain cases, if the electrical connection is different from the one supported by the controller, the cost of developing a new controller or variant exceeded the cost of changing the electrical connection to one that was supported. Therefore, it is crucial to note that the cascading effects of a seemingly small change—thereby, a small expenditure—can carry greater costs than expected in the two other domains.

(C2) Cascading effects driven by the software domain

Using the modular architecture from the mechanics domain, which is connected to the software by signals, the MESA tool helps illustrate how a change in one module in the software domain affects the control of the other modules.

Figure 2 (C2) shows how the humidity module in the software domain controls two modules and receives information from three other modules. Further, this module/state machine communicates with two state machines in another module. Therefore, changing the functionality of the humidity module in the software domain would affect the control of the two other modules directly and three other modules indirectly. One could then determine how the impacted modules are connected to other modules to identify third-order effects.

In Figure 3 (C2), for Case 2, the MESA tool displays the effect of adding functionality to the software module "Moderator." This change has a direct impact and needs the connected software modules to be altered (Training, Debriefing, Planning, and Effector Handler 1). In some cases, it was seen that to support the required changes for functionality, the physical effector needed to be changed too, for example, its angle, placement, and capacity of sensors and effectors in the mechanics domain.

Moreover, in a few cases, this required that changes be made to the controller in the electronics as well, which are all connected in the tool.

6.4 Evaluation of the application of the tool

In both companies, the tool was presented at meetings with product management and the involved domain experts. Both companies reported value in the overview of the cross-domain modularity dependencies. In Case 1, a more detailed evaluation was made. The evaluation included several domain experts: one electrical engineer, one mechanical engineer, and two software engineers, all with over five years of experience. A product manager, with over 10 years of experience, with the task of managing the portfolio of product, was also included. After being presented with the tool, they were given 10 statements and asked to rate their level of agreement (S1–10) on a scale from 1–4, where 4 means strongly agree, 3 means agree, 2 means partly disagree, and 1 means strongly disagree. The statements were constructed based on the researchers' application of the tool and the effects that they thought they could see in using the tool. With questions 1–3 (Q1-3), the participators could supply other written feedback regarding the use of the tool. The statements and questions are as follows:

S1. The MESA tool provides an overview of cascading effects on other modules across the domains of mechanics, electronics, and software when a module is changed in the mechanical domain.
S2. The MESA tool provides an overview of cascading effects on other modules across the domains of mechanics, electronics, and software when a module is changed in the software domain.
S3. An overview of the cascading effects across modules and domains will increase the efficiency of planned design activities.
S4. The MESA tool provides an overview of the redundancy of software code, where there exist multiple pieces of software code to handle the same functionality.
S5. Multiple pieces of software to handle the same product functionality are unnecessary and only add to complexity and development costs.
S6. The MESA tool provides an overview of the degree of centralization or modularization of the signal-handling components (PCBs).
S7. Centralization of signal-handling components reduces direct costs, but modularization increases maintenance efficiency.
S8. The MESA tool provides an overview of the simulation task in relation to separate tests of the modules.
S9. The MESA tool provides an overview of the differences in modular divisions between mechanics and software.
S10. Following the same modular division in mechanics and software increases development efficiency across the two domains.
Q1. What are the challenges with using the MESA tool?
Q2. What are the challenges with setting up the MESA tool?
Q3. Do you see any other benefits with using the MESA tool?

Table 1 shows the answers to statements 1–10 given by the five engineers. Statement 1 (S1) supports the ability to show cascading effects driven by the mechanics domain (C1), S2 supports the ability to show this from the software domain (C2), and S3 is related to the value of showing these effects.

The evaluation showed that there was an overall agreement of the value of this tool and its ability to show cascading effects across domains and modules. Furthermore, there was agreement that some modularity principles (S4–10) can be investigated using the tool. With statement four regarding the ability to show software redundancy, there is a tendency toward partly not agreeing with this. When engineers two and five were asked why they scored low on this, they replied that just because there were multiple pieces of software to handle the same functionality did not mean that they were redundant. They did, however, agree that such areas should be subjected to further investigation.

Table 1: Evaluation of the MESA tool. Scoring by five engineers from case Company 2. Under S1–3 is noted which cascading effects it supports.

	S1 (C1)	S2 (C2)	S3 (C1, C2)	S4	S5	S6	S7	S8	S9	S10
Engineer 1 software	4	3	3	4	4	3	2	3	4	2
Engineer 2 software	3	2	4	2	4	3	3	3	3	2
Engineer 3 electronics	3	3	2	3	2	4	4	3	3	4
Engineer 4 mechanics	4	3	4	3	4	4	4	2	4	4
Engineer 5 product manager	4	4	3	2	2	3	4	4	4	3
Average score	**4**	**3.5**	**3.5**	**2.5**	**3**	**3.5**	**4**	**3**	**4**	**3.5**

Table 2 shows an overview of the extra inputs given by the practitioners from answering questions one to three. Multiple practitioners supported some feedback, while other forms of feedback were unique. The main concern with using the tool is the amount of resources it takes to set it up and maintain it. However, we would claim that making an overview of cross-domain dependencies will always be relatively resourceful, but the impact of forgetting dependencies under a design change can also have a large impact with new design iterations or even quality problems after the products have been sold.

The fact that the tool is a simplified representation of reality is also one of the drivers for achieving an overview. If used together with domain experts, the authors expect that they will know if any important details are not shown. The knowledge from domain experts will also support changes so

that designers are not wrongly stopped from making changes because the impact seems too big. The tool should not be used alone by a single designer but should be seen as more than a tool to communicate impacts across domains under a design change among a cross-domain team.

One of the practitioners stated the need for commitment. We would even claim that commitment is not only needed across departments to fully succeed with a modularization strategy but also that top management commitment and involvement is needed (Sanchez, 2013).

The authors did not consider using the tool for common naming modules and signals, which was suggested by one of the practitioners as an additional benefit. In addition, the possibility of using it for investigating potential service error scenarios was new. The product manager suggested using the tool as "rethinking" the product. When asked about the details on this, he elaborated that knowing all the signals and physical modules in the product could foster new ideas to added functionality, which could be implemented by only adding new software, "a new way of using the product."

Table 2: Evaluation of the MESA tool. Answers from Engineers 1–5 (described in Table 1) to Q1–3. Numbers in right column refer to engineers 1–5.

Challenges with using and setting up the tool	**Stated by which engineers**
Difficult to maintain the overview (keep it up to date)	1, 2, 4
A simplified representation of reality (risk for not including everything)	2, 5
Resourceful to set up the tool	1, 2, 3, 4, 5
There needs to be commitment across departments to modularization and the general awareness of cross-domain dependencies	3
Changes may not be conducted because one can be "scared" of the impact the tool shows the change has, even though the change might not be that big	4, 5
Other benefits related to using the tool	**Stated by which engineers**
Easier knowledge share and understanding across domains	1, 5
Could be used for making common names of modules/signals, etc.	1
Maybe technical service can use it to make an overview of potential error scenarios	1
Useful to all who have a mindset for developing modular products	3
Gives a good overview of what affects what on a higher organizational level	4
Gives a more holistic view of the product that is normally not perceived	5
One could use the tool for rethinking the product by using all the building blocks already there and "only" changing the software	5

7 Discussion and conclusion

When developing mechatronic modular products, the visualization of the modular product architecture for a family of products across the engineering domains of mechanics, electronics, and software can help to illustrate the impact of changes on modules not only in one domain but also across all three. This would be helpful when planning the concurrent development of modules in the families of products.

Because of the lack of research to support modularity across domains and the need from industry to access cross-domain dependencies, we have developed the MESA tool. We then tested the hypothesis that the MESA tool can be used for mapping and communicating modularity dependencies across the engineering domains of mechanics, electronics, and software. This was done by applying the tool in two companies and evaluating its application with the help of senior engineers and product management. The evaluation showed that the MESA tool was able to show cascading effects across domains and modules caused by a design change from both mechanics and software. Knowledge about the dependencies across modules is one of the key factors for improving a modular design (Gershenson et al., 2003). Using the MESA tool, this knowledge is extended beyond a single-domain focus.

The existing research within modularization in engineering design focuses mainly on the mechanics and electronics domains, and no methods or tools directly include the associated software of mechatronic products (Bonvoisin et al., 2016; Gauss et al., 2020; Otto et al., 2016; Pirmoradi et al., 2014). Research has been conducted on modularization and software (Morales, 2014a, 2014b) but mainly focuses on the functional modularity of the software, not so much on the relation the dependencies with mechanics and electronics. Other tools and methods would be better for identifying the dependencies across modules in each of the domains; however, the MESA tool is intended for visualizing modular dependencies across mechanics, electronics, and software, explicitly including the associated software. The tool can be seen as an extension of the existing research on modularity, focusing directly on the dependencies across domains. More precisely, the MESA tool can be seen as an extension of the PFMP (Harlou, 2006) and interface diagram (Bruun et al., 2014).

We have tested the tool within two different companies. The application revealed two types of the cascading effects (C1, 2) that can be observed when using the model. These observations are aimed

at helping in planning improved modular designs or future design updates. The companies reported that the process of using the MESA tool helped in conducting a more precise planning process for the next generation of products in Case 1 and enabled the continuous development of future solutions in Case 2.

The observed effects were achieved by applying the tool to two different sized companies that operate in different businesses. This is not enough to claim generalizability, but it indicated that the observations were not unique to one company. The observations were evaluated in a survey made in Case 1. Five engineers were asked to rate their level of agreement with the tool's ability to show the observed cascading effects, some modularity principles, and the importance of these effects and principles. There was an overall agreement of the importance and ability of the tool to show modularity dependencies across domains and that the tool could be used for investigating modularity principle like the degree of centralization versus modularization of a signal-handling component like I/O boards. However, the ability to show the redundancy of software was rated lower. The practitioners agreed that the tool could be used for searching for areas with software redundancy, but a more detailed analysis of the software was needed to show this.

Regarding the chosen level of the granularity of product decomposition, one should consider choosing a level that allows for the discovery of all cross-module dependencies across domains. It has been described how this must be done to an extent where all components receiving and sending signals in the mechanics domain are divided in each module. In Case 2, the level of decomposition of the software was not high enough to reveal software redundancy. This leads to the conclusion that the software needs to be decomposed to a level where a single functionality is handled in the piece of code that is represented in the tool. This was done in Case 1, and the company was able to locate areas to investigate for software redundancy. Depending on the complexity of the product, different levels of decomposition will lead to different levels of detail (Walden et al., 2015). Therefore, one cannot give a certain level of decomposition that should be used. In future research, a process for determining a suitable level of granularity when decomposing the product family under analysis could be included. These processes have already been described by Hölttä-Otto et al. (2014), for instance.

We have observed two types of cascading effects and four modularity principles related to cross-domain modularity by using the MESA tool. These effects and principles are probably not a

comprehensive definition of such effects when making observations across mechanics, electronics, and software, but these are the ones that had the biggest impact on costs and lead time in the two cases.

By using the MESA tool, one can locate the dependencies between design decisions across domains. This becomes a way of creating awareness of the effects of design decisions from the mechanics domain into the software domain and vice versa. However, it does not support a true integration of the software development processes with product design methods. Object-oriented architecture for graphical user interfaces, such as the model view controller (MVC) and presentation abstraction control (PAC) (Hussey and Carrington, 1997), could perhaps be used or inspire the integration of product development that is not only related to user interface. This could be the subject of further research. We see the MESA tool as one of many enablers for integrating software and product development processes.

Furthermore, to avoid having this as a separate model, the inclusion of a functional layer in the model could be included in future studies.

Funding

The authors received no financial support for the research, authorship, or publication of this article.

Declaration of conflicting interests

The authors declare that there is no conflict of interests.

8 References

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

Alvarez Cabrera AA, Komoto H, Van Beek TJ, et al. (2014) Architecture-centric design approach for multidisciplinary product development. In: *Advances in Product Family and Product Platform Design: Methods and Applications*. Springer New York, pp. 419–447. DOI: 10.1007/978-1-4614-7937-6_17.

Askhøj C and Mortensen NH (2019) Deciding on the total number of product architectures. *Concurrent Engineering*: 1063293X1988896. DOI: 10.1177/1063293X19888968.

Askhøj C, Løkkegaard M, Bertram CA, et al. (2020) Identifying modularity practices across mechanics, electronics and software. In: *Proceedings of NordDesign 2020*, 2020. The Design Society. DOI: 10.35199/NORDDESIGN2020.9.

Bonvoisin J, Halstenberg F, Buchert T, et al. (2016) A systematic literature review on modular product design. *Journal of Engineering Design* 27(7): 488–514. DOI: 10.1080/09544828.2016.1166482.

Browning TR (2016) Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities. *IEEE Transactions on Engineering Management* 63(1): 27–52. DOI: 10.1109/TEM.2015.2491283.

Bruun HPL, Mortensen NH and Harlou U (2014) Interface diagram: Design tool for supporting the development of modularity in complex product systems. *Concurrent Engineering Research and Applications* 22(1): 62–76. DOI: 10.1177/1063293X13516329.

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

Clarkson PJ, Simons C and Eckert C (2004) Predicting change propagation in complex design. *Journal of Mechanical Design, Transactions of the ASME* 126(5). American Society of Mechanical Engineers Digital Collection: 788–797. DOI: 10.1115/1.1765117.
De Silva CW (2005) *Mechatronics : An Integrated Approach*. CRC Press.
Eppinger SD and Browning TR (2012) *Design Structure Matrix Methods and Applications*. MIT Press.
Erixon G, von Yxkull A and Arnström A (1996) Modularity – the Basis for Product and Factory Reengineering. *CIRP Annals* 45(1): 1–6. DOI: 10.1016/S0007-8506(07)63005-4.
Fixson SK (2007) Modularity and Commonality Research: Past Developments and Future Opportunities. *Concurrent Engineering* 15(2): 85–111. DOI: 10.1177/1063293X07078935.
Gauss L, Lacerda DP and Cauchick Miguel PA (2020) Module-based product family design: systematic literature review and meta-synthesis. *Journal of Intelligent Manufacturing*. Springer. DOI: 10.1007/s10845-020-01572-3.
Gepp M, Foehr M and Vollmar J (2016) Standardization, modularization and platform approaches in the engineer-to-order business Review and outlook. In: *2016 Annual Ieee Systems Conference (syscon)* (ed. IEEE), 2016, pp. 237–242.
Gershenson JK, Prasad GJ and Zhang Y (2003) Product modularity: Definitions and benefits. *Journal of Engineering Design* 14(3): 295–313. DOI: 10.1080/0954482031000091068.
Gershenson JK, Prasad GJ and Zhang Y (2004) Product modularity: measures and design methods. *Journal of Engineering Design* 15(1): 33–51. DOI: 10.1080/0954482032000101731.
Greve E and Krause D (2018) An Assessment of Methods to Support the Design of Future Robust Modular Architectures. *International Design Conference - Design 2018*: 335–346. DOI: 10.21278/idc.2018.0249.
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). Elsevier Ltd: 188–197. DOI: 10.1016/j.aei.2014.06.005.
Helbig T, Erler S, Westkämper E, et al. (2016) Modelling Dependencies to Improve the Cross-domain Collaboration in the Engineering Process of Special Purpose Machinery. In: *Procedia CIRP*, 1 January 2016, pp. 393–398. Elsevier B.V. DOI: 10.1016/j.procir.2015.12.123.
Hölttä-Otto K, Chiriac NA, Lysy D, et al. (2012) Comparative analysis of coupling modularity metrics. *Journal of Engineering Design* 23(10–11): 790–806. DOI: 10.1080/09544828.2012.701728.
Hubka V and Eder WE (1988) *Theory of Technical Systems*. *Theory of Technical Systems*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-52121-8.
Hussey A and Carrington D (1997) Using Object-Z to compare the MVC and PAC architectures. In: *Bcs-facs Workshop on Formal Aspects of the Human Computer Interface. Proceedings*, 1997.
Jiao J, Simpson TW and Siddique Z (2007) Product family design and platform-based product development: A state-of-the-art review. *Journal of Intelligent Manufacturing* 18(1): 5–29. DOI: 10.1007/s10845-007-0003-2.
Krause D, Beckmann G, Eilmus S, et al. (2014) Integrated Development of Modular Product Families: A Methods Toolkit. In: *Advances in Product Family and Product Platform Design*. New York, NY: Springer New York, pp. 245–269. DOI: 10.1007/978-1-4614-7937-6_10.
Kvist M (2010) *Product Family Assessment*. Technical University of Denmark. Available at: https://findit.dtu.dk/en/catalog/2389469614 (accessed 17 November 2020).
Løkkegaard M, Mortensen NH and 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. DOI: 10.1080/00207543.2018.1450531.
Martin M V. and Ishii K (2002) Design for variety: Developing standardized and modularized product platform architectures. *Research in Engineering Design* 13(4). Springer Verlag: 213–235. DOI: 10.1007/s00163-002-0020-2.
Meyer MH and Lehnerd AP (1997) *The Power of Product Platforms : Building Value and Cost Leadership*. Free Press.
Meyer MH and Utterback J (1993) The product family and the dynamics of core capability. *Sloan Management Review* 34(3): 29–47.

Morales CO (2014a) A heuristic approach to architectural design of software-intensive product platforms. In: *Advances in Product Family and Product Platform Design: Methods and Applications*. Springer New York, pp. 647–681. DOI: 10.1007/978-1-4614-7937-6_26.
Morales CO (2014b) Design principles for reusable software product platforms. In: *Advances in Product Family and Product Platform Design: Methods and Applications*. Springer New York, pp. 533–558. DOI: 10.1007/978-1-4614-7937-6_21.
Mortensen NH, Hansen CL, Hvam L, et al. (2011) Proactive Modeling of Market, Product and Production Architectures. In: *Proceedings of the 18th International Conference on Engineering Design (ICED)*, 2011, pp. 133–144.
Mortensen NH, Hansen CL, Løkkegaard M, et al. (2016) Assessing the cost saving potential of shared product architectures. *Concurrent Engineering Research and Applications* 24(2): 153–163. DOI: 10.1177/1063293X15624133.
Mougin J, Boujut J-F, Pourroy F, et al. (2015) Modelling knowledge transfer: A knowledge dynamics perspective. *Concurrent Engineering Research and Applications* 23(4): 308–319. DOI: 10.1177/1063293X15592185.
Muller P, Mattes A, Klitou D, et al. (2018) *Annual Report of European SMEs 2017-2018*. Luxembourg. DOI: 10.2873/248745.
Nadadur G, Kim W, Thomson AR, et al. (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*, 12 August 2012, pp. 837–848. American Society of Mechanical Engineers. DOI: 10.1115/DETC2012-70723.
Otto KN, Hölttä-Otto K, Simpson TW, et al. (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. DOI: 10.1115/1.4033654.
Parslov JF and Mortensen NH (2015) Interface definitions in literature: A reality check. *Concurrent Engineering Research and Applications* 23(3): 183–198. DOI: 10.1177/1063293X15580136.
Pedersen R (2010) *Product Platform Modeling: Contributions to the discipline of visual product platform modelling*. Technical University of Denmark. Available at: https://findit.dtu.dk/en/catalog/2389470855 (accessed 17 November 2020).
Pimmler TU and Eppinger SD (1994) Integration Analysis of Product Decompositions. In: *ASME Design Theory and Methodology Conference*, 1994, pp. 343–351.
Piran FAS, Lacerda DP, Antunes JAV, et al. (2016) Modularization strategy: analysis of published articles on production and operations management (1999 to 2013). *International Journal of Advanced Manufacturing Technology* 86(1–4): 507–519. DOI: 10.1007/s00170-015-8221-9.
Pirmoradi Z, Wang GG and Simpson TW (2014) A Review of Recent Literature in Product Family Design and Platform-Based Product Development. In: *Advances in Product Family and Product Platform Design*. New York, NY: Springer New York, pp. 1–46. DOI: 10.1007/978-1-4614-7937-6_1.
Prasad B (1996) *Concurrent Engineering Fundamentals, Volume I: Integrated Product and Process Organization*. Upper Saddle River: Prentice Hall PTR.
Prasad B (1997) *Concurrent Engineering Fundamentals, Volume II: Integrated Product Development.* Goodwin B (ed.). Upper Saddle River: Prentice Hall PTR. DOI: 10.13140/RG.2.1.4710.1527.
Prasad B (1999) Enabling principles of concurrency and simultaneity in concurrent engineering. *AI EDAM* 13(03): 185–204. DOI: 10.1017/S0890060499133055.
Prasad B (Brian) (2016) On mapping tasks during product development. *Concurrent Engineering* 24(2): 105–112. DOI: 10.1177/1063293X15625098.
Robertson D and Ulrich KT (1998) Planning for product platforms. *Sloan Management Review* 39(4): 19–31.
Sanchez R (2010) Creating Modular Platforms for Strategic Flexibility. *Design Management Review* 15(1): 58–67. DOI: 10.1111/j.1948-7169.2004.tb00151.x.
Sanchez R (2013) Building real modularity competence in automotive design, development, production, and after-service. *International Journal of Automotive Technology and Management* 13(3): 204–236. DOI: 10.1504/IJATM.2013.054918.
Sanchez R and Collins RP (2001) Competing - and learning - in modular markets. *Long Range Planning* 34(6): 645–667. DOI: 10.1016/S0024-6301(01)00099-1.

Schuh G, Rudolf S and Breunig S (2016) Modular Platform Design for Mechatronic Systems using Axiomatic Design and Mechatronic Function Modules. In: *Procedia CIRP*, 2016, pp. 701–706. Elsevier B.V. DOI: 10.1016/j.procir.2016.05.035.

Schuh G, Dölle C, Barg S, et al. (2019) Efficient Modular Product Platform Design of Mechatronic Systems. In: *IEEE International Conference on Industrial Engineering and Engineering Management*, 9 January 2019, pp. 1391–1395. IEEE Computer Society. DOI: 10.1109/IEEM.2018.8607714.

Simpson TW (2004) Product platform design and customization: Status and promise. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 18(1): 3–20. DOI: 10.1017/S0890060404040028.

Simpson TW, Bobuk A, Slingerland LA, et al. (2012) From user requirements to commonality specifications: an integrated approach to product family design. *Research in Engineering Design* 23(2): 141–153. DOI: 10.1007/s00163-011-0119-4.

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. DOI: 10.1109/TEM.1981.6448589.

Stjepandić J, Ostrosi E, Fougères AJ, et al. (2015) Modularity and supporting tools and methods. In: *Concurrent Engineering in the 21st Century: Foundations, Developments and Challenges*. Springer International Publishing, pp. 389–420. DOI: 10.1007/978-3-319-13776-6_14.

Thevenot HJ and Simpson TW (2006) Commonality indices for assessing product families. In: *Product Platform and Product Family Design: Methods and Applications*. Springer US, pp. 107–129. DOI: 10.1007/0-387-29197-0_7.

Uddin A, Khan MK, Campean F, et al. (2016) A framework for complex product architecture analysis using an integrated approach. *Concurrent Engineering Research and Applications* 24(3). SAGE Publications Ltd: 195–210. DOI: 10.1177/1063293X16647434.

Ulrich KT and Eppinger SD (2012) *Product Design and Development*. McGraw-Hill/Irwin.

Wagner F, Schmuki R, Wagner T, et al. (2006) *Modeling Software with Finite State Machines: A Practical Approach*. *Modeling Software with Finite State Machines: A Practical Approach*. CRC Press. DOI: 10.1201/9781420013641.

Walden DD, Roedler GJ and Forsberg K (2015) INCOSE Systems Engineering Handbook Version 4: Updating the Reference for Practitioners. *INCOSE International Symposium* 25(1). Wiley: 678–686. DOI: 10.1002/j.2334-5837.2015.00089.x.

Welp EG and Jansen S (2004) Domain allocation in mechatronic products. In: *Design 2004: Proceedings of the 8th International Design Conference*, 2004, pp. 1349–1354.

Weyrich M, Klein P, Laurowski M, et al. (2011) A function-oriented approach for a mechtronic Modularization of a sensor-guided Manufacturing System. In: *56TH INTERNATIONAL SCIENTIFIC COLLOQUIUM*, 2011.

Wilson S and Perumal A (2009) *Waging War on Complexity Costs*. McGraw-Hill Education. DOI: 10.1038/nsmb.3375.

Zouari A, Tollenaere M, Bacha H Ben, et al. (2015) Domain knowledge versioning and aggregation mechanisms in product design processes. *Concurrent Engineering Research and Applications* 23(4): 296–307. DOI: 10.1177/1063293X15591037.