TKK Dissertations 10
Espoo 2005

MODULAR PRODUCT PLATFORM DESIGN

Doctoral Dissertation

Katja Hölttä-Otto

Helsinki University of Technology
Department of Mechanical Engineering
Machine Design

TKK Dissertations 10
Espoo 2005

MODULAR PRODUCT PLATFORM DESIGN

Doctoral Dissertation

Katja Hölttä-Otto

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Mechanical Engineering for public examination and debate in Auditorium K at Helsinki University of Technology (Espoo, Finland) on the 12th of August, 2005, at 12 noon.

Helsinki University of Technology
Department of Mechanical Engineering
Machine Design

Teknillinen korkeakoulu
Konetekniikan osasto
Koneensuunnittelu

Distribution:
Helsinki University of Technology
Department of Mechanical Engineering
Machine Design
P.O. Box 4100 (Otakaari 4)
FI - 02015 TKK
FINLAND
Tel. +358-9-451 3551
fax +358-9-451 3549
URL: http://www.machina.hut.fi/
E-mail: katja.holtta-otto@tkk.fi

ISBN 951-22-7766-2
ISBN 951-22-7767-0 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)
URL: http://lib.tkk.fi/Diss/2005/isbn9512277670/

TKK-DISS-2027

Otamedia Oy
Espoo 2005

HELSINKI UNIVERSITY OF TECHNOLOGY P.O. BOX 1000, FI-02015 TKK http://www.tkk.fi	ABSTRACT OF DOCTORAL DISSERTATION

Author Katja Hölttä-Otto

Name of the dissertation

Modular Product Platform Design

Date of manuscript 7.6.2005	Date of the dissertation 12.8.2005
☐ Monograph	☑ Article dissertation (summary + original articles)

Department	Mechanical Engineering
Laboratory	Machine Design
Field of research	Product Development
Opponent(s)	Prof. Asko Riitahuhta, Prof. Kristin Wood
Supervisor	Prof. Kalevi Ekman
(Instructor)	Prof. Thomas Roemer, Prof. Christopher Magee

Abstract

Modular product platforms, sets of common modules that are shared among a product family, can bring cost savings and enable introduction of multiple product variants quicker than without platforms. This thesis describes the current state of modular platform design and identifies gaps in the current state. The gaps were identified through application of three existing methods and by testing their usability and reliability on engineers and engineering students. Existing platform or modular design methods either are meant for (a) single products, (b) identify only module "cores" leaving the final module boundary definition to the designer, and (c) use only a limited set of evaluation criteria.

I introduce a clustering algorithm for common module identification that takes into account possible degrees of commonality. This new algorithm can be applied both at physical and functional domains and at any, and even mixed, levels of hierarchy. Furthermore, the algorithm is not limited to a single measure for commonality analysis.

To select the candidate modules for the algorithm, a key discriminator is how difficult the interfaces become. I developed an interface complexity metric based on minimizing redesign in case of a design change. The metric is based on multiple expert interviews during two case studies. The new approach was to look at the interface complexity as described by the material, energy, and information flows flowing through the interface.

Finally, I introduce a multi criteria platform scorecard for improved evaluation of modular platforms. It helps a company focus on their strategy and benchmark one's own platform to the competitors'.

These tools add to the modular platform development process by filling in the gaps identified. The tools are described in the context of the entire platform design process, and the validity of the methods and applicability to platform design is shown through industrial case studies and examples.

Keywords Modularity, platform design, product architecture

Number of pages 65	ISBN (printed) 951-22-7766-2
ISBN (pdf) 951-22-7767-0	ISBN (others)
ISSN (printed) 1795-2239	ISSN (pdf) 1795-4584

Publisher Helsinki University of Technology, Laboratory of Machine Design

Print distribution Helsinki University of Technology, Laboratory of Machine Design

☑ The dissertation can be read at http://lib.tkk.fi/Diss/ 2005/isbn9512277670/

ABSTRACT

Modular product platforms, sets of common modules that are shared among a product family, can bring cost savings and enable introduction of multiple product variants quicker than without platforms. In this thesis I show how to define common platform modules with interfaces that require as little redesign effort as possible as well as how to choose a platform alternative that is well aligned with the company strategy. The focus of the thesis is on electro-mechanical products of medium complexity.

This thesis describes the current state of modular platform design and identifies gaps in the current state. The gaps were identified through application of three existing methods and by testing their usability and reliability on engineers and engineering students. Existing platform or modular design methods either are meant for (a) single products, (b) identify only module "cores" leaving the final module boundary definition to the designer, and (c) use only a limited set of evaluation criteria.

I introduce a tool for common module identification in a product family that can be used in conjunction with other methods to take into account the entire product family and not just single products. I introduce a clustering algorithm for common module identification that takes into account possible degrees of commonality. In addition this new algorithm can be applied both at physical and functional domains and at any, and even mixed, levels of hierarchy. Furthermore, the algorithm is not limited to a single measure for commonality analysis.

The tool alone identifies alternative common modules. To select these, a key discriminator is how difficult the interfaces become. I also developed an interface complexity metric based on minimizing redesign in case of a design change. The metric is based on multiple expert interviews during two case studies. This metric aids in the interface definition. The new approach was to look at the interface complexity as described by the material, energy, and information flows flowing through the interface.

Finally, I introduce a multi criteria platform scorecard for improved evaluation of modular platforms. It helps a company focus on their strategy and benchmark one's own platform to the competitors'. It also serves as a communication tool for upper management as well as between different stakeholders.

These tools add to the modular platform development process by filling in the gaps identified. The tools are described in the context of the entire platform design process, and the validity of the methods and applicability to platform design is shown through industrial case studies and examples.

PREFACE

This research started in April 2001 as a part of the research project FineMed[1] at Helsinki University of Technology (TKK). FineMed was a TEKES (a Finnish National Technology Agency[2]) funded project on product development and concepting in fine mechanics. The original thesis topic was to "develop tools to improve cooperative R&D in small and medium sized companies". The idea was to use modularity tools in collaborative product development since modules, according to the literature, enable independent development due to well defined interfaces. It was, however, quickly discovered that modularity was not a mature enough area to be used as such, but more research was needed. This lead to re-scoping of the research area into improving modularity and product architecture in general.

I would like to thank both TKK Laboratory of Machine Design and Massachusetts Institute of Technology (MIT) Center for Innovation in Product Development (CIPD) for supporting me during my studies. In addition, I would like to give my gratitude to TEKES, Graduate School for Concurrent Mechanical Engineering (GSCME), Tekniikan edistämissäätiö (TES), and Emil Aaltosen Säätiö for their financial support.

During my research I was lucky to work with three professors: Kalevi Ekman at TKK, who has always believed in me and given me all the support and opportunities needed; Thomas Roemer, who was my first host at MIT and taught me how to do scientific work; and Chris Magee, my second MIT host who taught me scientific rigor in data analysis.

I would also like to thank Professors Mogens Myrup Andreassen from Denmark University of Technology and Kristin Wood from the University of Texas at Austin for reviewing and commenting on this thesis in such thoughtful and detailed manner.

I would like to thank the companies and helpful experts that I worked with during my thesis work. The close contact to companies has helped make the tools easier to adapt in industry.

I am writing this preface a little over 4 years after starting this work. These four years would not have been as productive, interesting, inspiring, and fun, if there weren't for my friends and colleagues both at TKK and MIT. Thank you all!

I must also thank my mom, dad, sisters and brother as well as all the friends outside work who listened me stress about the work and who wonderfully kept me in touch with other things in life.

And finally, I would like to thank Kevin. The word everything does not begin to cover what I mean by Everything. Kiitos.

Katja Hölttä-Otto Watertown, June 6, 2005

[1] http://www.machina.hut.fi/finemed/
[2] http://www.tekes.fi/eng/

CONTENTS

LIST OF PUBLICATIONS

I Hölttä, K., Suh, E. S., & de Weck, Olivier. Trade-off between modularity and performance for engineered systems and products. *In Proc of International Conference on Engineering Design*. Melbourne. August 15-18, 2005. (to appear)

II Holtta K. & Salonen M. Comparing three modularity methods. *In Proc of ASME Design Engineering Technical Conferences*. Chicago, IL. September 2-6, 2003.

III Holtta, K., Tang V., & Seering W. Modularizing product architectures using dendrograms. *In Proc of International Conference on Engineering Design*. Stockholm. August 19-21, 2003. (a continuation submitted to RED Jan/Feb 04)

IV Holtta, K. & Otto K. Incorporating design complexity measures in architectural assessment. *In Proc of ASME Design Engineering Technical Conferences*. Chicago, IL. September 2-6, 2003.

V Holtta, K. & Otto, K. Incorporating design effort complexity measures in product architectural design and assessment. *Design Studies.* (to appear)

VI Otto, K. & Holtta, K. A multi-criteria framework for screening preliminary product platform concepts. *In Proc of ASME Design Engineering Technical Conferences*. Salt Lake City, UT. September 28 - October 2, 2004. (also to appear in *Journal for Intelligent Manufacturing*)

Additional publications not included in the thesis:

Hölttä K. Comparative analysis of product modularization methods. NordDesign. August 18-20, 2004. Tampere, Finland.

Holtta, K. & Magee C. Estimating factors affecting project task size in product development – an empirical study. IEEE Transactions on Engineering Management (conditionally accepted)

AUTHOR'S CONTRIBUTION

Publication I: Performance tradeoff. This work is done together with Professor de Weck, Eun Suk Suh and with the help of 2 undergraduate students. Research concept, literature search and the data analysis as well as most of the writing are done by the author.

Publication II: Modularity method comparison. The experiment design and the data gathering and analysis are done solely by the author. Mikko Salonen helped with the experiments and co-wrote the article.

Publication III: Module commonality, dendrograms. This work, including the idea development, data gathering, analysis, writing, etc., is done almost entirely by the author, but the original idea and its development was a joint brainstorming effort of Victor Tang and the author. Professor Seering and Dr. Otto advised in the writing.

Publications IV and V: Interface complexity. Publication IV introduces the metric, but the metric is modified and improved with a second case study in Publication V. Both are done primarily by the author with advice from Dr. Otto. The second case study in Publication V is done together with Hannu Valo, a master's student.

Publication IV: Platform assessment scorecard. This work is done together with Dr. Otto. The author was responsible for the literature review of existing metrics, transformation of single product metrics into platform metrics, and the case example. The author also was responsible for the most of the writing.

NOMENCLATURE

D	Distance between two modules
DSM	Design structure matrix
DSP	decision support problem
m_k	index of the last component in the k^{th} module
M_m	total number of modules in the product
M	number of output types in a product family
MFD	Modular function deployment
MIM	Module indication matrix
n_k	index of the first component in k^{th} module
N_c	number of components in the product
N	number of input types in a product family
OPM	Object-process methodology
PC	personal computer
PD	product development
PPCEM	product platform concept exploration method
QFD	Quality function deployment
R_{ij}	value of the i^{th} row and j^{th} column element in the modularity matrix.
R&D	research and development
UML	Unified modeling language
x_k^i	size of input type *k* for module *i* in product *I*
x_k^j	size of input type *k* for module *j* in product *J*
y_k^i	size of output type *k* for module *i* in product *I*
y_k^i	size of output type *k* for module *j* in product *J*
$W_{k_{IN}}$	weight of the input type *k*
$W_{k_{OUT}}$	weight of the output type *k*

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Background

In the product development (PD) literature, tools and methods are often described as if PD is a unique process from a clean sheet to a new one-of-the-kind product. They are also described as if development can be done for a single standalone product. These *clean sheet designs*, however, are not as common place as *derivative* product development. A typical project is more likely a derivative i.e. modification project of an older product. For example at General Electric 85% of development projects are modification projects [122]. These modifications form product generations over time. In addition, companies often add new parallel products to their product line to form a product family. The multiple lines of Nokia cell phones or Volkswagen vehicles, including the Skoda and Audi brands, are good examples. Further, these products are rarely started with a clean sheet but are based on something that exists already, something based on the company's core competence. Henderson and Clark [46] dub these derivative products as incremental or modular innovations. Figure 1 schematically illustrates the idea of the multi-product world.

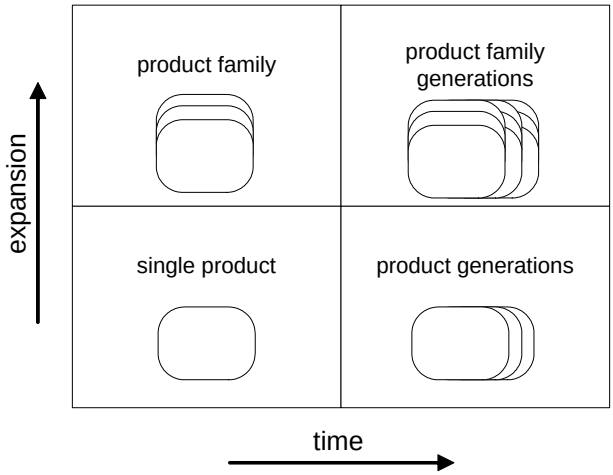


Figure 1 Single products are rare.

Moreover, instead of a single product PD process and organization, I will argue that there should also be a multi-project, platform process and organization for platform projects. Platform projects develop the base from which the multiple products can be derived later in single product PD processes. The platform projects should go hand-in-hand with the strategic planning of the company. These are different from derivative projects, where the multiple variants are developed, which can be run in the product development organization with tighter time-to-market demands. Wheelwright and Clark [121] also support the separation of platform and derivative projects in order to remain competitive.

Tatikonda [113] surveyed 108 platform and derivative projects in 51 companies and found that the platform and derivative projects are and should be significantly different. For example, platform projects involve more technology development and generally aimed at newer markets than derivative products. He found, interestingly, that companies

employ the same management strategies regardless of the project type. This may be due to the fact that many product development models describe the PD process as a single process, where the platform planning happens in the planning or concept development phase of the product development. I will claim that a platform processes should be considered different from derivative product development processes. There may be, in fact, multiple platform projects for different technologies, for example. The platform projects should be part of strategic planning and technology development, and the platform projects should not happen in the tight schedule of a PD project due to the different properties of a platform project. The platform projects will provide the basis, a set of technology platforms, etc., for the actual product development processes (Figure 2).

In principal the process steps for platform and derivative product development are similar, but the inputs and outputs are different. The platform process may start with a clean sheet, or an existing product family, whereas the derivative product development process starts with a platform, the outcome of the platform process. The output of a platform project is a platform, not a product, whereas the output of a derivative project is a product to be launched in to the market, based on the platform.

Nokia, for example, has separate platform and product development processes[3]. The Nokia platforms represent technologies (e.g. Bluetooth) as well as design rules for the mechanical design. The platforms then form a basis for the product ideas that can be developed based on one or multiple platforms.

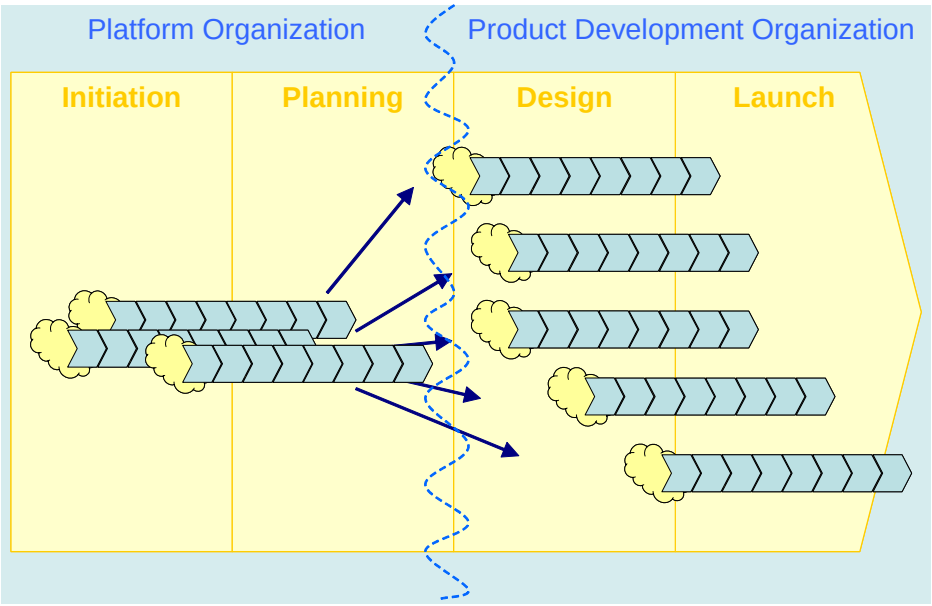


Figure 2 Three platform processes and five derivative product development processes overlaid with a traditional process.

A typical PD project consists of phases. I have adapted a PD process model from Ulrich and Eppinger [117] for this thesis (Figure 3). I will apply this modified model for the platform development process. The last phases of the process here do not mean that the platform is launched as a product to the customer, but that the platform organization delivers the platform to the product development organization. Similarly the 'after sales'

[3] Based on several personal conversations and personal observations of the products

refers to the phase where the product development organization may give modification suggestions to the platform core modules.

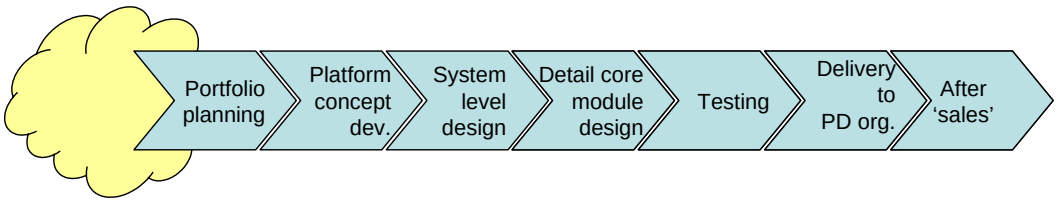


Figure 3 A typical product development process [adapted from 117].

This thesis focuses mainly on the beginning phases (Figure 4) of the process since these are the phases when portfolio, platform, and architectural decisions are made.

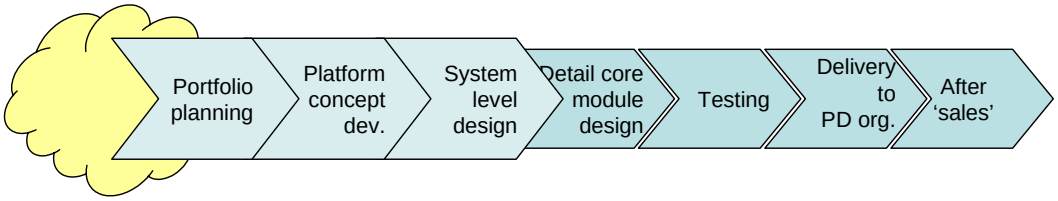


Figure 4 The focus of the thesis is on the beginning phases of a development project.

The beginning is a crucial part of the PD process. After the system level design, up to 80% of the product's cost is determined [20]. By the end of this phase a company has committed to certain solutions involving specific technologies and configuration of the product. This commitment ties up the investment and makes it increasingly more difficult to make changes to the product's design. [9] Also Kaplan *et al*. [60] show how error prevention in the early phase can cost as little as 6% of the cost of error correcting toward the end of the product process. Therefore, it is important to have good systematic methods that produce good results at the system level design phase.

There are many strategies for proper product architecture design and platform development. This thesis describes one especially for modular product platforms.

1.2 Objectives

In this thesis I will show how to define common platform modules with easy to redesign interfaces as well as how to choose a platform alternative that is well aligned with the company strategy.

The key idea is to develop a methodology that is well founded and yet easy to use in practice. I will identify the gaps in the methods and develop new methods to make the modular platform development process more complete. The research addresses the following four questions:

1. What are the biggest gaps in the modular platform development methods to date?
2. How can module interface complexity be described quantitatively?
3. How can the identification of common module candidates for modular platform design be improved?
4. How to evaluate the "goodness" of a modular platform and its fit to the over all company strategy?

The result of this thesis will be a better understanding of the modular platform development process as a whole, as well as improved methods for platform development and evaluation.

1.3 Theoretical Approach

In a larger context this thesis is part of *design science*. Hubka and Eder [54] define design science as "a system of logically related knowledge, which should contain and organize the complete knowledge about and for designing". This research is in full accordance with this definition in that this thesis presents a framework of new and improved methods for design, specifically architectural design.

Hubka and Eder [54] further state that the goal of design science is to improve the situation in the design area and eliminate existing problems. This second criterion of design science describes the approach in this thesis. I start by analyzing existing methods, identify their strengths and weaknesses, and then build the new methods on that.

Design research, according to Blessing *et al*. [10] includes three stages: (1) Descriptive study I, where the goal is to identify factors that lead or prevent success; (2) Prescriptive study, where a method or theory is developed based on the results of the first stage; and (3) Descriptive study II, where the methods are applied, and the contribution to success is analyzed. Typically, a research project focuses on one to two stages. This research starts in stage 1 and ends at stage 2. The primary research method in these stages is case study research. All the analysis of past and newly developed methods is done with case study products from real companies. Yin describes case study research as an iterative three step process shown in Figure 5. This process is used in this thesis.

To ensure the validity of the developed methods, the four tests of validity are applied as defined by Yin [125]: construct, internal and external validity, and reliability. The construct validity of research can be assured by using multiple sources of information. Internal validity is the most difficult to prove in case study research according to Yin. The internal validation approach used here is *explanation building* and *pattern matching*. Both tactics are analytical tactics of making sure that the conclusions drawn from the case studies follow logically from the data gathered. The external validity has to do with the generalizabilty of the results. In case of case study research the results are often generalized analytically as opposed to statistically as in other types of research. Also the statistical generalization is used in this work when possible. The analytical generalizability is similar to the repeatability of the results. The repeatability can be tested using the same approach for more than one case study and checking whether results are the same. This approach is used in this work. The fourth type of validity, reliability, is similar to the external validity in that both can be measured as the repeatability of the research. Reliability, however, is about the repeatability of a single case. A case study should produce similar results independent of the person doing it. This is difficult in practice, since an industrial case study can rarely be fully replicated, but the aim here is to minimize the biases e.g. repeating the case study procedure with multiple people from each case study company.

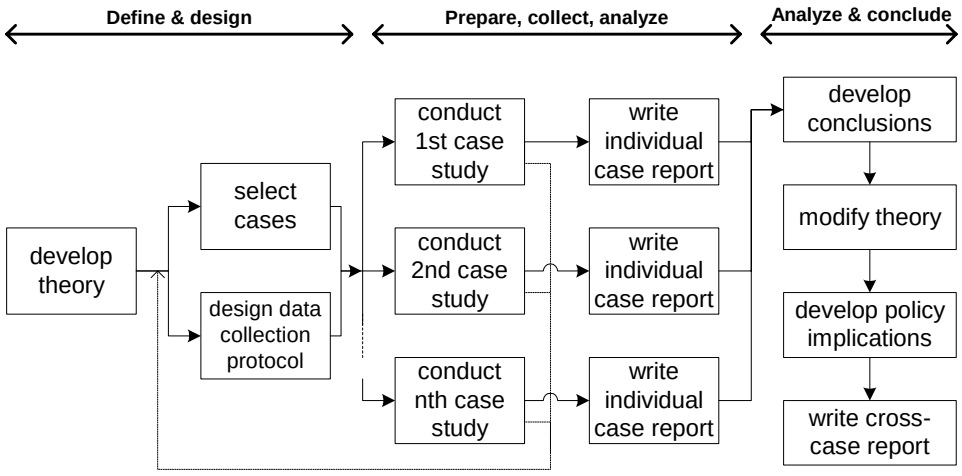


Figure 5 Case study method picture adapted from [125].

1.4 Scope of the Thesis

The focus of this thesis is on a modular approach to product platform design. Modular design cuts through product architecture, product platforms, etc., but those issues are dealt with in this thesis only in where they relate to modular platform design.

This thesis deals with modular product platforms. There are also other types of valid platforms, but they are not the primary topic in this thesis. Further, product platforms affect and are affected by multiple facets of the business process. This thesis focuses on the engineering side of modular product platforms – how to design and develop modular platforms. Company strategy is considered, mainly in the platform evaluation phase, since it is closely linked with product platforms, but the business strategy issues are not the foci in this research.

The theoretical part of this thesis is general to any product, but since the case studies used to test and validate the methods are electro-mechanical products of medium complexity, I cannot claim proven applicability of this work beyond these product types.

1.5 Outline of the Thesis

This thesis consists of three main parts. The first part is the theoretical foundation of the research area. This establishes the fundamental underpinning of the research. The second part is the literature, of which the thesis can be viewed as a continuation. The third part is the actual contribution of the thesis – a methodology consisting of three separate tools to design and evaluate product platforms. The figure below illustrates the outline of the thesis (Figure 6).

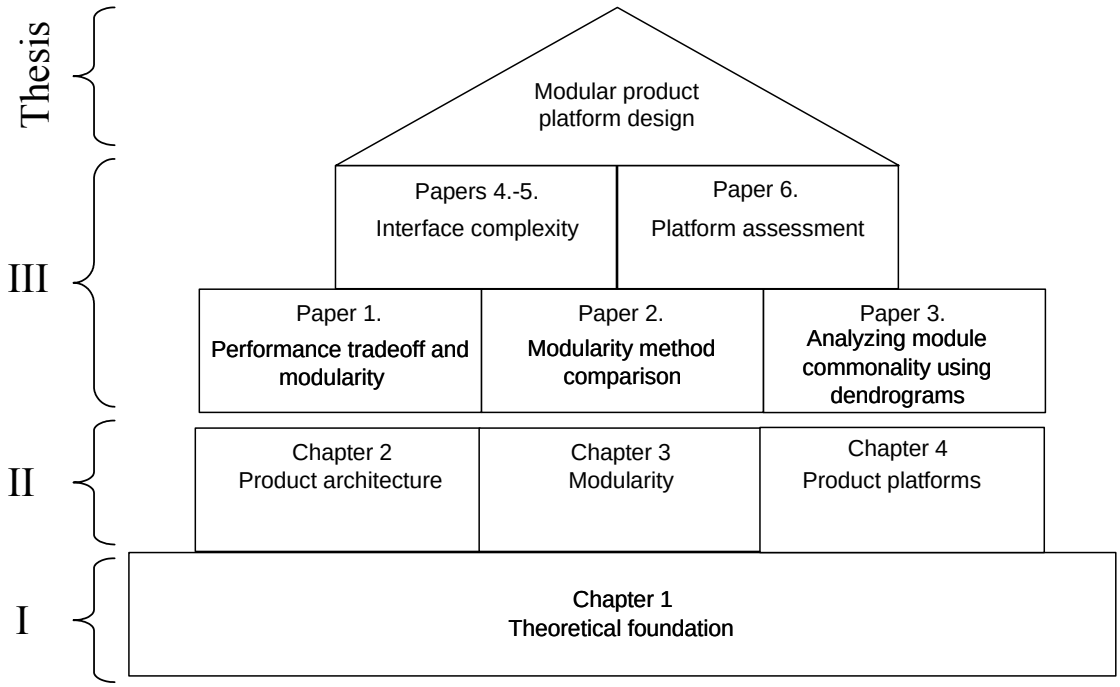


Figure 6 Outline of the thesis.

Chapter 1, Introduction, forms the theoretical basis for this research. It describes the theoretical approach used as well as the scope of the thesis. Chapters 2 through 4 introduce the state-of-the-art of research in platform architecture and other relevant topics. The first publication is summarized in Chapter 3. This thesis is a continuation of this platform architecture research. Chapter 5 summarizes 4 of the articles that form the main contribution of this thesis. These articles describe platform development methods. Chapter 6 summarizes the 6th article on evaluating platform architectures. And finally, Chapter 7 concludes the major findings of this research.

1.6 Original Features

This thesis includes methods for improved product platform development. The following features are believed to be original:

1. Literature review of platform development, modularity, and product architecture, with concentration on how to develop and evaluate platforms, modularity, and architecture and identifying gaps in the current methods. The main contributions are:
    a. Evaluation of 6 different architectural representations.
    b. Comparison of modularity methods based on actually using the methods on 6 products including a repeatability analysis with 40 participants
    c. Performance tradeoff analysis between modular and integral architectures using quantitative examples. The earlier work in this area is only qualitative.

2. A new metric to analyze and describe interface complexity quantitatively based on the interface type. The complexity is based on the ease of redesigning adjacent

modules if an interface changes. The new approach here is to look at the interface complexity as described by the material, energy, and information flows flowing through the interface. The metric is evaluated by using it on two case studies.

3. A new quantitative method to evaluate module commonality. Unlike most measures before, this method sees platform and component commonality analysis not as a binary, common/not common choice, but as a more complex decision of degree of commonality at the functional level. The method is also flexible in that it can compare functional commonality within and across products at the same time. Further, the method is not restricted to comparing functional commonality at a single level on the function hierarchy, but it can compare commonality across the hierarchies. In addition, the method can handle functions described by multiple and different units of measurement. The method is shown to work through real examples.

4. A platform evaluation scorecard that includes a comprehensive set of metrics. This is a new compilation of existing as well as new metrics to evaluate platform "goodness". The existing metrics were modified to fit for platform design and not only single product design. The usability of the tool is shown via an example.

2 PRODUCT ARCHITECTURE

In this chapter I will first define the term architecture for the purpose of this thesis and then discuss the multiple ways of representing a product or system architecture.

2.1 Definitions

Merriam-Webster on-line dictionary [120] has one potentially relevant definition of architecture:

> *The manner in which the components of a computer or computer system are organized and integrated*

Ulrich [118] defines architecture as:

> *The scheme by which the functions of a product are allocated to physical components*

This definition recognizes that a product can be realized through alternative architectures. The US Department of Defense, on the other hand, use more life cycle thinking in their definition of architecture:

> *The structure of components, their relationships and the principles and guidelines governing their design and evolution over time.* (CJCSI 3170.01D)

Maier and Rechtin [69] have a systems approach and include the process in their definition:

> *The structure (in terms of components, connections, and constraints) of a product, process, or element.*

Crawley *et al*. [19] give a similar definition for system architecture, but instead of physical components they refer to entities that could be functions, physical or non-physical "components", etc.:

> *System architecture is an abstract description of the entities of a system and the relationships between those entities.*

The definitions deal either with the physical structure of a product, the abstract representation of the system components, or the mapping between the two. The common theme in all these definitions is the arrangement of elements of a product. The last definition is the most abstract and therefore also less restrictive than the other definitions. I will use a definition adapted from the last definition by Crawley *et al*. in this thesis while still recognizing the existence of alternative architectures of a product as in Ulrich's definition:

System architecture is an abstract description of the entities of a system and the relationships between those entities and the scheme by which these entities are mapped to larger physical or non-physical sub-systems of a system.

2.2 Representation

There are multiple ways of representing a product, or system, architecture. I will shortly present here a few models commonly found in architecture literature and argue why a specific representation is chosen for this thesis. All the representations concentrate on the physical (components or sub-systems) or functional (product functions) decomposition [62].

2.2.1 Six Architectural Models

The simplest way of representing an architecture is probably a hierarchical tree structure. In a hierarchical tree, a system is decomposed to sub-systems and the system architecture can be looked at different levels of abstraction. Figure 7 shows how a system and its sub-systems can be represented as a tree structure.

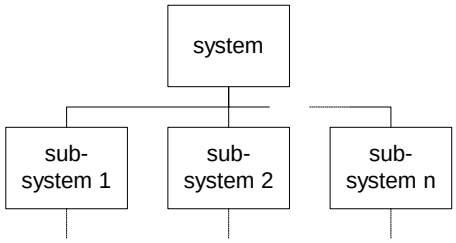


Figure 7 A hierarchical tree structure.

Hubka and Eder describe an *organ structure, especially developed* to represent a (technical) system, which principally means a machine that does work. An organ, according to them, is "a system that realizes a given internal function" [53]. They do not, however, present a specific symbolic way of representing an organ structure. An organ is similar to what others call simply a function, something that the product does. Pahl and Beitz represent architectures as functional decomposition block diagrams of all the product's functions. Alternatively the functional decomposition could be replaced by a physical decomposition into components (and sub-systems). These *function structures* include all the material, energy, and information flows as arrows between the functional blocks (Figure 8). [89] The division of the flows makes the function structures suitable mainly for electromechanical products.

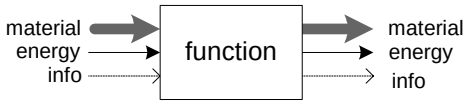


Figure 8 A single function block of a function structure with basic flow types.

IDEF0 [57] is another way of modeling a system. It was originally developed to model processes. This is similar to the function structures in that in IDEF0 the functions are also presented as blocks, and there are inputs and outputs to and from the functions (Figure 9). Alternatively the function could be replaced by a component (or a sub-system). These inputs and outputs are, however, not decomposed into different types, but instead two more input arrows are added in addition to the basic function input. These are

a *control* arrow to represent a controlling element and a *mechanism* arrow to represent the tool or resource performing the function.

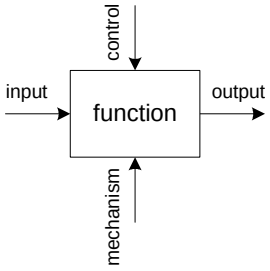


Figure 9 Basic structural unit of an IDEF0-diagram.

Another popular way of representing architecture is a *design structure matrix* (DSM) [117]. It was originally developed for modeling organizations. The DSM is analogous to the function structure, but here, the functions are presented as row and column headers of the matrix instead of the function boxes and the connections between the functions are shown in the matrix (Figure 10). The connection mark ("1" in Figure 10) indicates that the function on the row depends on the function on the column. For example, in the figure below, functions 2 and 3 depend on function 1 and function 1 depends on function 3. The marks can, as shown by Pimmler and Eppinger [94], also be divided into spatial (S), material (M), information (I), and energy (E) interactions, as shown on the right in Figure 10. In addition to functions the rows and columns could represent components, tasks, team members, etc.

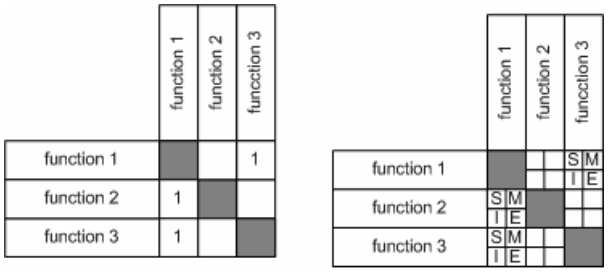


Figure 10 A design structure matrix.

So far, the architectural models have described either the functions or components (or sub-systems) of a product or system and their interconnections. Object-process methodology (OPM) [25] was developed to include both aspects into the model at once. Sub-systems can be presented as objects (parts, and other elements involved in the system) and functions as processes (Figure 11). The objects are represented with rectangles and processes with ellipses. In addition, the links between the objects and processes can be represented with multiple symbols. To indicate that an object performs a process, the object is connected to the process with a connector that has a black circle at the process end. A white circle indicates that the object is part of the process but not the agent performing it. In addition states (▭), effects (→), and aggregations (•) can be represented in the OPM diagram.

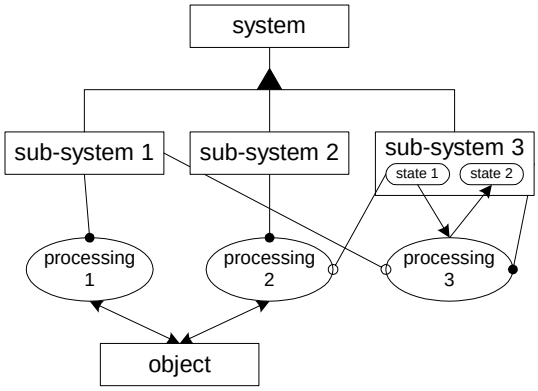


Figure 11 Object-process diagram.

The OPM is a close cousin of *unified modeling language* (UML), the last approach to modeling an architecture described here. This language was originally developed for software design, but it can be used to model also non-software systems. Below is an exemplary *structural* UML diagram of a product (Figure 12). In addition there are other views to describe an architecture. These other views make the UML the most general model, but some views, e.g. states, are modeled also by the OPM diagram. Here, the *classes* represent the basic concepts of a system (similar to components of a product) and each class can have a set of *attributes* and *operations* to describe their properties and the alternative functions each class can do. The relation can be any verb that describes the respective roles of the two classes. The relations between the classes can describe e.g. how one class controls, consists of, or reads the other class.

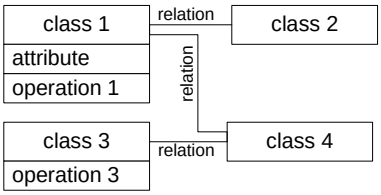


Figure 12 UML diagram.

2.2.2 Comparison of the Six Architectural Models

It is not agreed which of these architectural representations is the best or most suitable for a specific case. The different architecture representations are best suited for different purposes. The focus of this thesis is on products and their functions and structure, and not for example on a process or use case of a product, in which case the choice of method would be different.

I will analyze the models here from nine different aspects:

1. Whether both functions and actual elements of a product can be represented. The element here refers to a component of a physical system or e.g. a block of code in a software system.
2. Whether the user of the product can be included in the model.
3. Whether the surroundings of the product can be included in the model.

4. Whether the model can differentiate between different interface types or complexities.
5. Whether the model is static or dynamic i.e. whether the model can show different views or states of the system (dynamic) or just a single state and view (static).
6. Whether the model is suitable for modeling service and software architectures in addition to electromechanical architectures.
7. Whether the model can be used in the early phases of the development process.
8. What aspects of the product are visualized with the model.
9. Whether the model is compatible to other models i.e. if one representation is transferable to another without any additional information.

To better illustrate the differences between the models, I will use each method to model the same product: a water bottle. To keep the models simple, I will only model three functions of a water bottle: holding water, directing water to mouth, and sealing container if needed. In addition, of course, the water bottle includes functions such as show contents, show how much left, enable holding, etc. Figure 13 illustrates the three functions of a water bottle using the six architectural representations introduced.

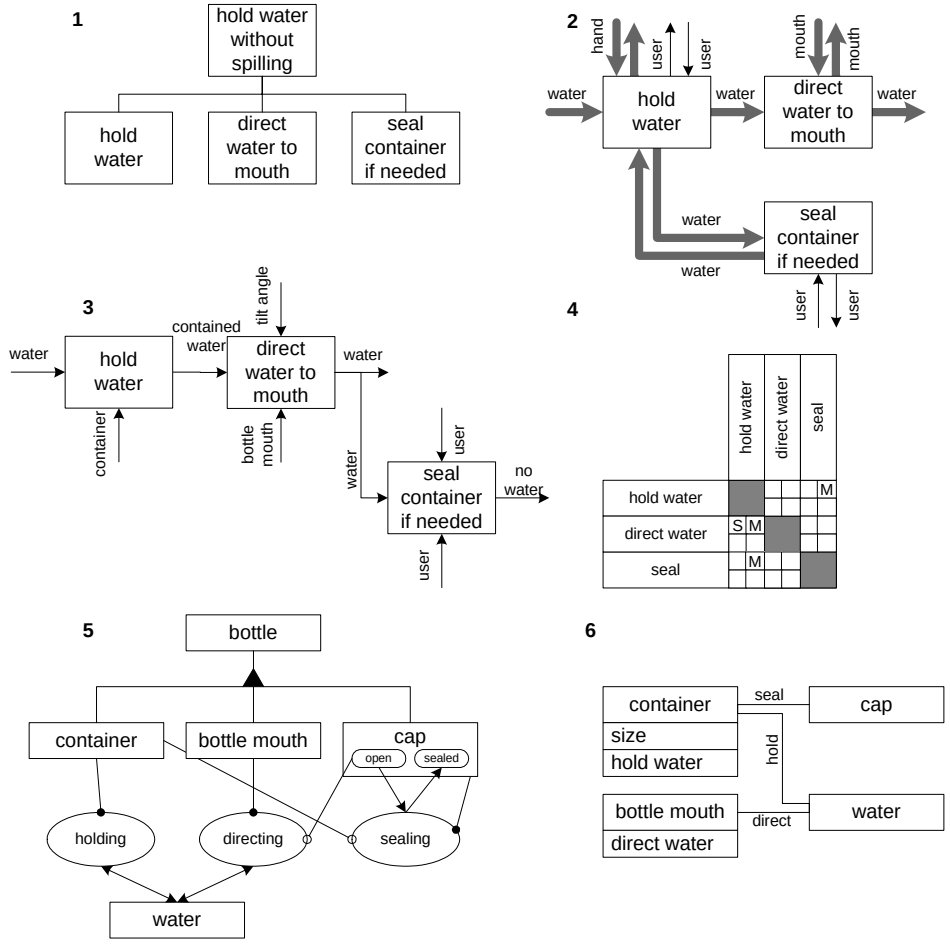


Figure 13 A water bottle modeled using six different architectural representations.

Looking at the hierarchical tree structure of the water bottle, we see that it only shows the functions of the product but does not reveal any details about their relations. Alternatively the tree structure could contain the elements of the bottle, but still leaving out the relations between them.

Function structure can also have either the functions (as shown above) or the elements of a product. In addition to the tree structure, we now have connections and connection types between the functions to represent how the functions relate to one another. Water flows from the hold water function to the outside of the system through direct water functions. In addition, one can see that water can also enter the seal function, but the flow of water stops there and returns to the hold water function. Further, the function structure includes also connections to the outside of the system. In the water bottle example, a user is seen as a hand and force holding the bottle, mouth touching the direct water part of the product, and user sealing the bottle with the cap.

The IDEF0 includes the functions or elements of the product as well and their relations. It does not, however, detail the interaction types, or flows (water), in the architecture like function structures. IDEF0 representation can include the user and possible other connections to the outside of the product it self. This is presented as control and mechanism arrows. For example, the user is both the mechanism that seals the bottle and controls that the sealing is done.

The DSM is very similar to the function structure model. The main difference to the function structures, in addition to the spatial interaction, is that connections to the outside of the system are not shown in the DSM. In addition, the matrix format of the DSM enables easy reorganization of the architecture using matrix manipulation. Most algorithms, however, are for the binary DSM, where the interactions are not separated into the four categories.

OPM, as discussed above, enables simultaneous modeling of functions and elements of a system, unlike the function structure and DSM methods, for example. In addition, the surroundings and the user of the system can easily be included in the diagram. In the water bottle example the object the system (bottle) operates on is water. Water is not part of the system itself but can be included in the model. Further, dynamic aspects such as states of the system can be included in OPM. The bottle cap can be sealed or not, for example. This modeling method provides a more complete description than the others so far.

The structural UML model of the water bottle also includes the functions and the elements of the system. The elements are presented as classes and the functions and their operations. Note that now the operations and relations are very similar. Each relation could be replaced by “acts upon” to avoid use of same verb as in operations, but the above notation is chosen for clarity. If a class had more operations, i.e. an element had more functions, these operations would be divided to different relation lines to different classes. In addition, relation lines could describe e.g. aggregations such as in the OPM diagram. Further, one could include attributes for each class, e.g. the water container could have size as an attribute. If one were to draw different diagrams for the other views, many more features of the system could be described, but not in a single diagram.

All six architectural representations have benefits and drawbacks and a choice on which to use depends on the situation what in the most suitable for a specific purpose. Most of these methods were used at some point of this research. During the research I

identified certain key features that an architectural representation must have for platform development purposes. Similar to customer needs in a KANO chart, the need for an architectural representation can be divided into both basic needs that are “must” and special features that make the method better than expected. The basic needs include the ability to illustrate functions/elements of the product and their relations. A special feature that was also important in this research was the capability to show the interfaces in detail between the functions/elements in the structure. This stemmed from the need for improved understanding and design of interfaces and linking the customer requirements into the functions for commonality analysis. This will be discussed later in the thesis.

Further, since the focus of the thesis is in the early phases of development, it is important to have an architectural model that can be created when only customer requirements are known, not the components of the system. In addition, it is sometimes important to be able to represent an architecture of an existing system with an abstract model in order to not tie the thinking to the existing solutions. This opens possibilities for fundamental [31] redesign of the product architecture. For example, if a company making a water bottle was in fact in the business of providing water containers that one could easily drink from and seal if necessary, then the same functions could be realized by multiple product types (Figure 14) or combinations of different attributes of the different product types. In order to keep these possibilities open and enable platform commonality across different product types, an abstract model with no components is needed.

Figure 14 Five alternative water containers with functions: hold water, direct water to mouth, and seal container if needed.

Additional good features of an architectural representation are its abilities to visualize the product design problem as well as to provide clues about the product context (e.g. user and surroundings).

In summary, during the course of platform design, different methods may be needed, or a combination of them to benefit from the power of the methods. For this to be easy and effective, it is desirable that a method is compatible or easily transferable to another method. As an example the function structure is maybe easier to read than a DSM, but a DSM is quick to manipulate. Table 1 summarizes the main features of the six models giving clues to a designer which method to use in a specific situation.

Table 1 Comparison of architectural representation methods.

	Hierarchical tree	Function structure	IDEF0	DSM	OPM	UML
Functions / elements	F or E	F or E	F or E	F or E	F & E	F & E
user	-	+	+	-	+	+
surroundings	-	+	+	-	+	+
Interface types	-	+	-	+	-	+
static/dynamic	static	static	static	static	dynamic	dynamic
Suitable also for service and sw architectures	(+)	-	-	(+)	+	+
Can be used at early phases	(+)	+	+	+	(+)	(+)
Visualization	levels of hierarchy	functional layout, interf. types	functional layout	interface connectivity	objects involved	objects involved
Compatible w/ methods	-	DSM (IDEF0)	(DSM)	Function str. (IDEF0)	(UML)	(OPM)

In this thesis I will analyze product architectures and develop systematic methods for architectural design. I will choose function structures as the primary architecture representation method since it has the most features needed but not a lot of additional information to clutter the presentation in my work and since it is transferable to other representations with minimum effort. The key issues that lead to the selection of the function structure as the main method were (1) the function structure can be drawn at the early phases. The minimum information needed are customer requirements; (2) The function structure is an abstract demonstration that enables comparison of different product types (e.g. in a product family); (3) The function structure includes a separation of interface types and includes units of measurement related to the customer requirements; (4) Representing dynamics aspects, i.e. the states of the system or different view points, were not found to add significant value. Also DSM would have been a good choice for (1), (2), and (4), but the function structure is more visual and has the most information needed for the interface definition. Also OPM could be used as well as the DSM, but similarly the interface type separation is not as suitable for this research as in the function structure model.

Further, Kurfman *et al.* have shown function structures to be reasonably repeatable [66]. They also show that function structures result in quasi-unique product representations and that the functional basis vocabulary improves the functional modeling by making it more repeatable and by determining a level of decomposition. They also show that the method works for both redesign and an original product, but benefits are clearer with redesign projects [67].

3 MODULARITY

Modularity has become very popular in academia in recent years even though it has existed for at least 30 years [31] and the idea of hierarchical systems consisting of semi-independent sub-systems was brought up by Simon [101] already in 1962. Several companies have adopted modular thinking in various industries such as Boeing, Chrysler, Ford, Motorola, Swatch, Microsoft, Conti Tires, etc. [85]. In this chapter I will first define how the term module is used in this thesis, and then discuss different measures for measuring the degree of modularity of a product. I will end with tying modularity to product architecture and discussing the advantages and disadvantages of modularity.

3.1 Module Definitions

Gershenson *et al*. [36] note in their literature review that there is no agreement on the definition of modularity. There is some agreement that a "more modular product is one with more modules that are closer to the *ideal* module". But the definition of an ideal module is not agreed upon. This is largely due to the definition of a module being related to the benefits sought from modularity.

O'Grady [85] defines "hard" and "soft" modules. "Hard" modules are physical assemblable modules and "soft" modules have limited physical presence e.g. software, service, financial products, insurance, etc. In this thesis I do not make a separation between the two, both are considered equally modules. This choice is since a single product can consist of both types of modules, and therefore the architectural analysis is complete only if both types of modules are considered.

Mattson and Magleby divide modularity into three categories: design, manufacturing, and customer modularity [71]. Also Gershenson categorizes modules into the design and manufacturing, as well as the end-of-life modularities.

Independent of the life cycle phase or purpose of modularity, Merriam Webster [120] has two relevant definitions for a module:

1. *a : any in a series of standardized units for use together: as (1) : a unit of furniture or architecture (2) : an educational unit which covers a single subject or topic*
   b : a usually packaged functional assembly of electronic components for use with other such assemblies
2. *an independently-operable unit that is a part of the total structure of a space vehicle*

Hubka and Eder [53] define a modular design as "connecting the constructional elements into suitable groups from which many variants of technical systems can be assembled". Salhieh and Kamrani [98] define module as "building block that can be grouped with other building blocks to form a variety of products". They also add that modules perform discrete functions, and modular design emphasizes minimization of interactions between components. Also Camuffo [15], Dahmus *et al*. [21], Pahl [90], as well as Ulrich and Eppinger [117] have a similar definition. They all define a module as a chunk of a product with an identifiable function.

The above definitions are mainly based on the functionality of a module. Another common way of defining a module is a more abstract definition such as that of Otto and Wood [87]: "product modules are defined as integral physical product substructures that have a one-to-one correspondence with a subset of a product's functional model". Also Stone *et al*. [109] use a very similar definition derived from Ulrich's definition of architecture.

Ericsson and Erixon [27] add that in addition to the similarity between the physical and functional architecture of a product, a module should have minimal interaction with other modules or the rest of the system. This strong connectivity within a sub-system and loose connectivity between sub-systems was discussed by Simon [101] quite early. Baldwin and Clark [5] define a module as "a unit whose structural elements are powerfully connected among themselves and relatively weakly connected to elements in other units". Also Suh [111] considers the connectivity of the module to the rest of the system in his definition where a module is a row in his design matrix.

The module definition used in this thesis is adapted from the above sources:

> *A module is an independent building block of a larger system with a specific function and well-defined interfaces.*

In addition, a module has fairly loose connections to the rest of the system allowing an independent development, outsourcing, manufacturing, recycling, etc. of the module as long as the interconnections at the interfaces are carefully considered. This definition is general to different product types and gives a definition that helps identify modules to benefit from the facts listed in Section 3.4, Advantages and Disadvantages of Modularity.

3.2 Modularity Measures

When talking about modularity, a question arises: how modular is a product platform? In order to quantify modularity, many measures have been developed, but the answer is not trivial as pointed out by Gershenson *et al*. [35]. They conducted a study where groups of independent students, engineers, product development managers, and researchers had to evaluate the degree of modularity of 10 consumer products. Interestingly, there was no statistical significance to the answers i.e. there was no agreement on what was more modular than another.

There are many attempts in the modularity literature to measure the degree of modularity [e.g. 2, 37, 55, 70, 71, 84, and 106]. Guo and Gershenson [44] developed a new metric by first studying eight existing metrics [43], including their own, and then validating their improved metric through experiments. This metric is in line with the module definition used in this thesis. It measures the intra- (first term) and inter-module (second term) connectivity in a modularity matrix, such as a DSM:

$$Modularity = \frac{\sum_{k=1}^{M_m} \frac{\sum_{i=n_k}^{m_k}\sum_{j=n_k}^{m_k} R_{ij}}{(m_k - n_k + 1)^2} - \sum_{k=1}^{M_m} \frac{\sum_{i=n_k}^{m_k}(\sum_{j=1}^{n_k-1} R_{ij} + \sum_{j=m_k+1}^{N_c} R_{ij})}{(m_k - n_k + 1)(N_c - m_k + n_k - 1)}}{M_m}$$, where

n_k = index of the first component in k^{th} module
m_k = index of the last component in the k^{th} module
M_m = total number of modules in the product
N_c = total number of components in the product
R_{ij} = the value of the i^{th} row and j^{th} column element in the modularity matrix.

As found also by Guo and Gershenson [43], modularity measures are very different and give different results on the degree of modularity. Most measures deal with physical components, but a few can be extended to the abstract design phase by replacing the components with functions. About half of the metrics are designed for a specific application, such as recycling or supply chain management, and the other half are metrics to calculate the degree of modularity in general or in terms of connectivity. If the purpose of modularizing is to separate outsourced components into modules, a supply chain specific metric (e.g. [55]) is the most suitable, and when the goal is to develop independent modules, a metric based on the connectivity (e.g. [44]) of the module is more appropriate. The metric by Guo and Gershenson is used in this thesis since it is in line with the modularity definition in this paper (Publication I).

3.3 Modular Architectures

There are many ways of categorizing architectures. One common way is to divide architectures into modular and integral architectures. In reality a fully modular or fully integral architectures are rare and almost all architectures are somewhere in between.

Modular architecture has functionally de-coupled interfaces between components [118]. In practice this often also leads to an architecture is one where the functional elements in the function structure are mapped one-to-one to the components (or elements to be more general) of the product. This is because in order for a component to be an independent module, it needs to interact as little as possible with the other components, and this is achievable, for example, when each component has only one function, or at least no functions are shared between components. Typical examples architectures that have close to one-to-one mapping between functions and components and that are at the modular end of the modular-integral scale include a mechanical pencil, a personal desk top computer (PC), and the Swiss army knife (Figure 15).

Figure 15 Examples of modular products.

An integral architecture is the opposite of a modular architecture. An integral architecture has coupled interfaces between components [118]. An integral architecture tends to have more complex (non one-to-one) mapping from functional elements in the function structure to the components (or elements) of the product. Typical examples of architectures at the integral end of the modular-integral scale, where it is hard to identify

what part of a product performs which function, include an old fashioned pencil, a laptop PC, and a hunting knife (Figure 16).

Figure 16 Examples of integral products.

3.4 Advantages and Disadvantages of Modularity

Modularity brings both advantages and disadvantages. Modularity often means using the same module in multiple products enabling a large variety of products while using less different component types than if the different products did not share common modules. This *multiple-systems modularity* [79] brings scale and scope advantages such as reduced capital requirements, and economies in parts sourcing [5, 85, 118]. On the downside, modularity may lead to excess costs due to over design [42, 64], inefficient performance [26, 124], and too many common modules may result in loss of brand identity [61, 118].

Modules are also helpful in design re-use [79, 104] since already designed modules with well defined interfaces can be used again in other designs. This applies to software products as well as hardware [6]. Design re-use can lead to reduced cycle time, which in turn results in e.g. increased revenue due to increased market penetration as a result of being first to market, success in time sensitive markets, and shorter time to market increases accuracy of meeting customer needs [74].

A well designed product architecture can help the management of product change and upgrades, product variety, and components standardization [85, 118]. Product change, upgrade, and variety can be achieved by replacing a module in a system without other changes to the overall product, or product platform [27, 117]. In addition, from a *single-system* [79] point of view, a well defined module, in terms of simple interfaces, can ease project management due to decoupling of tasks and providing design freedom within a module [85, 118]. Modularity also makes a complex product architecture appear simpler and therefore easier to manage [27].

The above advantages are useful in the design phase of a product. Fixson [30, 32] and Miller [79] outline how modularity has impact at the different phases of a product's development lifecycle. Fixson says modularity can also have different effects depending on the stakeholder; e.g. a supplier's cost might actually rise when the manufacturer applies a certain module regime to reduce its costs. Also Pahl and Beitz [89] discuss the advantages and limitations from different stakeholders' points of view (manufacturer and user). Coulter *et al*. [18] also found similar results. They introduce a *limiting factor* i.e. "a characteristic of an existing product for which a change in value of the characteristic (or a change of the characteristic itself) to another value in the feasible design space would result in increased achievement of product goal targets". In their case it can be e.g. one part in a module that makes the entire module non-recyclable. In addition, Newcomb [84] as well as Allen and Carlson-Skalak [2] view modularity at the end of a products

lifecycle - as a tool to ease the disassembly and recycling of the product. Also Riitahuhta and Andreasen [96] and Dahmus and Otto [22] discuss the life cycle benefits of modularity. These tradeoffs between different stakeholders and lifecycle phases must be considered when designing modular products.

Another trade-off in modularity is the trade-off between performance (e.g. high efficiency, low weight) and the business oriented benefits (e.g. high product variety, flexibility) that can be achieved with modular designs [1, 20, 26, 124]. This is discussed in detail in Publication I. Whitney [124] points out, that especially high power mechanical products, as opposed to low power signal processor type products, would benefit from more integral design if technical performance is high priority. A more modular product is likely, but not necessarily, to be larger, heavier and less energy efficient than a product with integral architecture [119, 124]. Also side effects are harder to control. Whitney compares complex electro-mechanical-optical products to VSLI that can be considered fully modular, and in line with Suh's [111] design axioms. Mechanical parts have a "multi-function character" partly due to basic physics (material contains also energy, rotating axle transmits shear loads and rotational energy) and partly due to "design economy". Also Gonzalez-Zugasti and Otto [38] show that some performance is sacrificed to obtain goals of the individual products that are created for a platform.

I show here an illustrative example of the technical performance trade-off using a simple truss example. There are two different architectures for a simple one triangle truss. The first consists of three identical beams of same length, cross section profile, and material, i.e. is a modular structure, and the second is an integral one piece structure (Figure 17). Both triangles have a vertical load of 50N. This causes a compressive force of approximately 29 N to the two angular beams and a tension force of approximately 14 N to the horizontal beam. For the first structure, where all beams are identical, they are chosen according to the most critical requirement, i.e. the two angular beams. The horizontal beam is therefore over designed (larger diameter than needed) and the structure is heavier than if the structure was more optimized such as the second integral one piece truss where the lower section is thinner than the upper parts of the structure. Clearly, modularity makes a product heavier.

On the other hand, if the load or the load direction were to change, the modular truss structure is quicker to adapt to the new requirements. Modularity makes a product more flexible toward change.

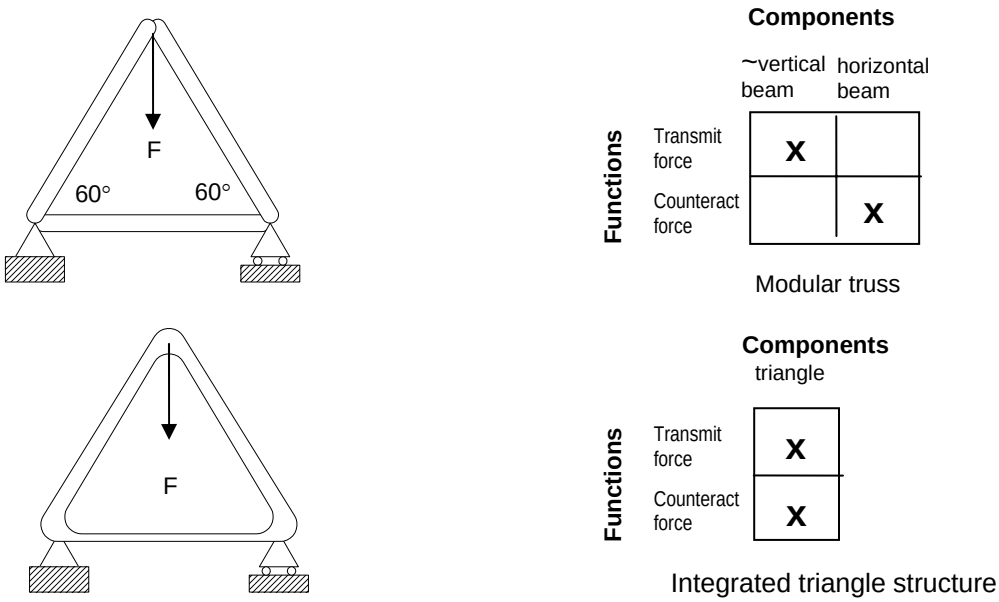


Figure 17 Modular and integral truss.

In addition we investigated the degree of modularity compared to the performance level, mainly in terms of power consumption and weight limit, of an architecture using two product pairs as an example: a cellular phone and a desk phone, and a laptop and a desktop computer. We found that the more performance constraint products (cellular phone and laptop computer) are more integral than the non-performance critical counterparts (desk phone and desk top computer). Other examples of products where the modular product is (or would be) heavier are a car and an electronic calculator [20]. The cellular phone example is discussed also in [7]. More details and the modularity calculations are in Publication I.

Gershenson concludes in his literature review that even though there is agreement on the benefits of modularity, there's no large scale validation of it. He adds that there is no research on how long modularity brings benefits and when it causes diminishing returns. [36] Kusiak [68], on the other hand, argues that the full potential of modularity is not realized, and the research should continue in the area.

4 PRODUCT PLATFORMS

So far I have discussed product architecture and modular architectures. These form the basis for an effective platform design. This chapter will define the concept of platform and how it is used in this thesis, discuss the benefits of modular platforms, and introduce the state-of-the-art of platform method research to date.

4.1 Definitions

Meyer and Lehnerd [78] define a platform as a "set of common components, modules, or parts from which a stream of derivative products can be efficiently created and launched". Muffato [81] defines platform similarly as: "a relatively large set of product components that are physically connected as a stable sub-assembly and are common to different final models". Also Ulrich and Eppinger [117] share a similar definition.

McGrath [75] and Otto and Wood [87] have a more general definition, where platform is a collection of common elements (not just physical components), especially the underlying technology, that are implemented across a range of products. Simpson *et al*. [102] have an even more general platform definition: "the set of parameters (common parameters), features and/or components that remain constant from product to product, within a given product family".

The more general definitions enable platforming in design, manufacturing and assembly, and product phases. The last definition by Simpson *et al*. takes into account that platforms can be either module or scale based [87, 103]. Since this thesis focuses on module based platforms, the platform definition used here is one that is suitable for module based platforms. The definition used in this thesis is derived from the above sources:

> *Platform is the common set of physical or non-physical modules from which multiple products can be derived*

This definition is in line with the literature and industry practice. In addition there are many valid platform definitions regarding the interface between the product and the manufacturing system e.g. the assembly coordinates or welding points of a product. But these are outside the scope of this thesis. This definition also supports the product development process framework in section 1.1, Background .

4.2 Benefits

The benefits of platforms are similar to the benefits of modularity since modules are often used to create either modular platforms or product variants by adding a module to a platform. A classic example of a successful use of platforms is the Sony Walkman story [99]. They were able to create more variants and faster than any competitor. Also Volkswagen has outperformed its competitors in terms of selling the most vehicles based on their platforms [95]. Platform projects also enable later derivative projects that are much shorter in duration than the platform projects [52]. Derivative products are more likely to succeed than totally new products as shown by the Association of National

Advertisers, who found that 27 % of product line extensions fail; whereas 31 % of new products introduced into existing categories fail; and a very high 46 % of new products introduced in new categories fail [4]. Good platform can enable a set of successful product variants. Meltzer [76] claims that product families and platforms can be used as a tool to accelerate new product development since developing a derivative product based on a platform is faster than developing a completely new product. However, Roemer and Fixson point out the limits of this strategy and warn of potential lead time increases under commonality [97]. The faster development time applies also in the context software development [6, 112]. Muffato [81] discusses the benefits of automotive companies adopting platform strategies and claims that even though there has already been success in shortened lead times, among other benefits, there is room for improvement. Also other success stories can be found in literature [21, 27, 78, 87, 126].

Meyer and Lehnerd [78] discuss the benefits of product platforms: scale advantages etc. They also introduce a list of metrics to measure platform performance (in dollars). The metrics are based on the "business" performance (to use Whitney's [124] term) of a platform and the costs of developing it. However, Krishnan and Gupta point out that the platform development costs are, in general, a very small percentage of the total life cycle costs. They suggest that the cost of using an over designed part in order to have an identical module instead of two (or more) variations will end up costing much more than the original platform investments. Also Moore *et al*. stress the importance of considering both the fixed and variable costs of platforms [80].

4.3 Methods

There are several methods for designing a platform. Simpson *et al*. find that there are two types of platform design methods: (1) top-down and (2) bottom up. [102] Another way to characterize the two approaches is that the top-down approach is more business and the bottom up approach more technically oriented. Yet a third way of categorizing platform design methods is to distinguish between module based and scale based platforming [87, 103]. Scale based platforms are platforms where products share the functionality but are all at different performance levels. Examples include: Pratt & Whitney jet engines [87] and Black & Decker universal motors [78]. Module based platforms, on the other hand, are products that share common modules but may have different functionality. Examples of this include Sony Walkman [99] and Black & Decker tools [110]. I will use this last categorization in this chapter since the methods are often suitable for either scale or module based platforms.

4.3.1 Scale Based Platform Methods

Several researchers [e.g. 47, 77, 82, and 102] are developing optimization based methods for designing a platform. Simpson *et al.* [102] introduce a method called Product Platform Concept Exploration Method (PPCEM). They use decision support problem (DSP) to try to design a platform by minimizing performance loss and maximizing commonality. Their method starts with market segment grid from Meyer and Lehnerd [78]. Messac *et al*. [77] also start with the market segment grid. They show a method to provide decision support in designing product families. Messac *et al*. start with the assumption that the common platform components are known and then identify parameters that designer can effect as well as noise. They include a step for robustness,

but do not show it, then they use physical programming to formulate and solve the problem. [77] Similarly, Hernandez *et al.* [47] show a compromise DSP approach for designing robust product families. They too start with a presumption that common components are known. Hernandez *et al*. focus on production costs of the platforms. Nayak tries to define platforms based on minimizing the variations of corresponding design variables in different products of a product family. Also he uses DSP to optimize the platform. [82]

Conner Seepersad *et al.* show a quantitative method to decide on a number of product platforms, or number of common components, for a family of absorption chillers. They also use compromise DSP. [16] In later work Conner Seepersad *et al*. add a utility based method that takes into account the evolving markets [17]. The changing market is added as an expected utility of predetermined possible scenarios (each scenario is given a certain probability of occurrence).

These methods concentrate on scalable common functionality. This is important, but outside the scope of modular product platforms, the thesis topic.

4.3.2 Module Based Platform Methods

This thesis focuses on module based platforms. Researchers approach module based platform design from many viewpoints. Moore *et al*. [80], for example, use conjoint analysis to determine a product platform. Siddique and Rosen [100], on the other hand, describe a method to design platform from an existing set of products by looking at the commonalities in the assembly process. Gonzalez-Zugasti *et al*. [40] introduce an iterative method for optimizing platform design based on minimizing cost. In another work, Gonzalez-Zugasti *et al.* [39] developed a method to assess the value of a platform. They use a real options approach to determine a path to choose when developing an initial platform and possible variants/derivatives in the future. Also Steuer and Whitcomb [108] use real options to assess the value of a platform. Steuer and Whitcomb focus on market uncertainty instead of technical uncertainty. Most of these methods concentrate on evaluating a platform, once the platform modules have been chosen.

A few authors have developed matrix based methods for platform design. Fujita *et al*. [33] introduce a way of using Quality Function Deployment QFD [45] for product families. They assign a zero weight to a customer requirement that does not exist in a specific model but exists in at least one member of a product family to be able to use the same matrix for multiple products in a family. Also Martin and Ishii [70] developed a QFD based method for developing platforms. They aim to minimize the connectivity and future redesign of the architecture with a help of modularity metric introduced earlier. Dahmus *et al*. [21] also use matrix approach. They focus on defining platform modules based on common functionality. Sudjianto and Otto [110] introduce a similar method as Dahmus *et al*. to design multi-brand product platforms based on shape and color schemes rather than on technical attributes. These methods address the choice of common modules for platforms, but the methods presume the modules are predefined.

There are multiple ways of determining the degree of commonality in a platform. Fellini *et al*. [28] introduce a method to choose common components for a platform while trying to optimize both the commonality and the performance. In previous work, Fellini [28] introduced a pareto optimizing method to decide how many design variables to share among two products of a family with a given acceptable performance loss. Nelson *et al*.

[83], on the other hand, use pareto fronts to decide on the degree of commonality between products in a product family. They optimize the performance of a single product and the degree of commonality.

In addition to the actual design methods above, De Weck *et al.* [24] have developed a method for deciding the number of platforms based on sales volumes and performance at market segments. And Georgiopoulos *et al*. [34] show how to determine how much to produce each of the product variants in a platform.

All of the methods presume that the modules and common elements to be shared for the platform are decided or provide only weak guidance as to how to do that. These methods are useful in optimizing the platform or deciding the number of platforms, but they lack detailed advice for the design engineer who is designing the platform – making decisions about the interfaces, common elements, etc. The following chapters will look more into specific tool for architecture design.

5 DESIGNING AN ARCHITECTURE

This chapter is the main part of this thesis and contains the main research contribution. In this chapter I will introduce an approach for designing a "good" product architecture using a modular design approach. The goodness can be assessed in terms of *ilities* [19]: properties such as upgradability, serviceability, flexibility, etc. This will be covered in Chapter 6, Evaluating Platform Architectures. The idea is to develop product family architectures that enable product variety by designing an architecture consisting of independent modules that are defined in accordance with the company's modular strategy. In this section, I first introduce three modularity methods, present results of the comparison of them, and then move on to discussing the improvements I have developed to overcome the weaknesses of the current methods.

5.1 Modularity Methods

Modularity definitions and methods depend on the purpose of modularity. For example at what point do we want benefits of modularity – during the design phase (design reuse etc.) or at the end of life of a product (recycling etc.)? Fixson [31] and Gershenson [36] also support this. I focus on the architecture and design phase but try not to ignore other aspects. The methods introduced here were chosen since they are well established in academia and used in industry.

5.1.1 Function Structure Heuristics

Stone *et al*. developed a *function structure heuristic* method, based on Pahl and Beitz's function structures [89] introduced in Section 2.2. Stone *et al*. separate modules from a single product's function structure by finding the dominant flow, branching flows, or conversion-transmission function pairs (Figure 18) [109]. Zamirowski and Otto [126] present three additional heuristics to find modules across products in a product family. They find similar and repetitive functions within a single product, common functions across products, and unique functions that are found only in one product within the product family and separate them as modules. These three product family heuristics are similar to the component standardization strategies by Perera *et al*. [93]. In addition to these, McAdams *et al*. [72] separate causally linked function pairs as modules, but since all modularity methods are used in their original form, this is left out of this study. A good tutorial of the method is given by Otto and Wood [87].

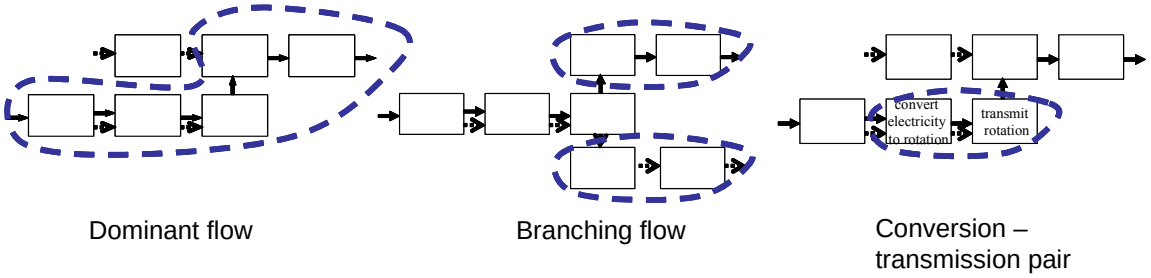


Figure 18 Function structure heuristics.

To apply the function structure heuristics method, one starts with a function structure, and then considers the many possible alternative modules that can be defined

by grouping functions according to the heuristics. The heuristics define possible modules; it is up to the designer to choose the "sensible" modules. Further, the heuristics are maximal heuristics. They state only that one should not define modules larger than indicated. Any module defined by a dominant flow as a serial chain of functions, for example, can be subdivided in any way and still be consistent with the heuristics. As such, the approach provides modularity suggestions only; it is not a deterministic algorithm. Therefore, designer insight and good judgment can enter the process; this is either a benefit or a problem, depending upon one's perspective.

These heuristics apply to single products and the three family heuristics to product families of similar products. The method can be applied for both module based and scale based platforms, but the most common use is with module based platforms. The main modularization criteria considered in the function structure heuristic method are functionality and module interfaces. Other criteria such as business or strategy related factors are not represented in the function structure heuristic method but, instead, enter through designer judgment in where the rules get applied. Otto [86] presents a method based on the functions structure heuristics that includes also steps for customer segmentation and profit estimation.

5.1.2 Modular Function Deployment

Modular function deployment (MFD) [27] is also based on functional decomposition, such as functions structure heuristic method, but in this method, modularity drivers other than functionality are considered. MFD is designed to modularize a single product at a time. There are twelve modularity drivers in MFD (Figure 19). One or a few modularity drivers are chosen according to the firm's strategy. Ericsson and Erixon [27] offer a good tutorial on the method.

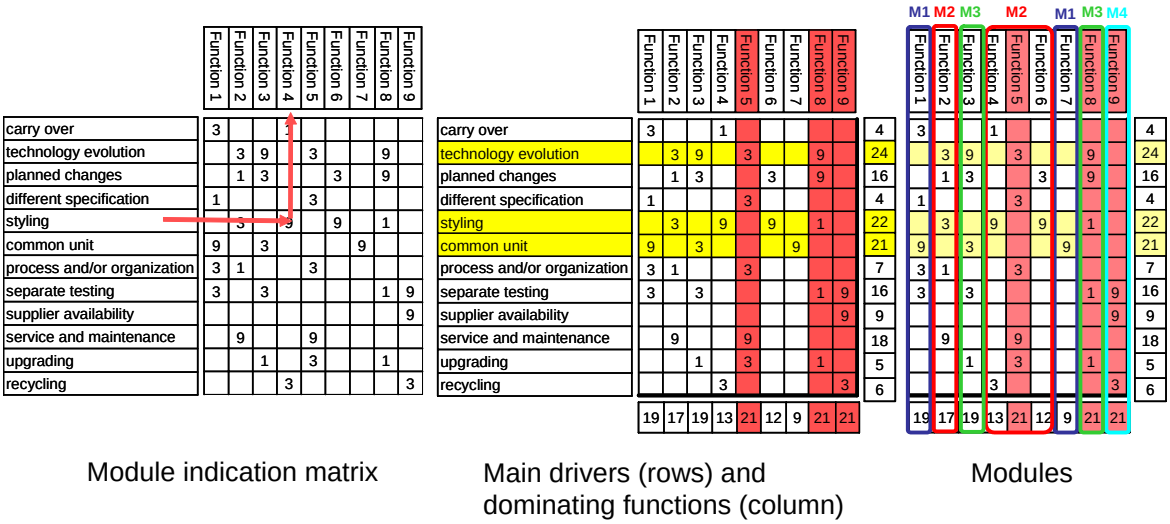


Figure 19 Main steps of modular function deployment.

MFD is similar to QFD, but here modularity drivers are mapped against functions instead of customer requirements in a matrix (Figure 19). The grouping into modules is started by the functions receiving the highest summed scores (dominating functions, see Figure 19); and the functions dominated by the same modularity drivers are good candidates for a module according to this method. The number of modules according to

MFD is approximately the square root of the number of parts or assembly operations. The estimate is based on optimizing the assembly lead time of the whole product. [27]

Stake [107] and Blackenfelt [8] show how MFD and DSM can be integrated in the grouping phase. Blackenfelt builds a strategic DSM using simplified modularity drivers from the MFD. He suggests using also a functional DSM in conjunction with the strategic MFD [8] to systematize the grouping phase in the MFD. In addition, MFD has a step for interface design that considers form, fixation principles, number of contact surfaces and attachments, as well as the number of energy connection points, material flow, and signals. It relies more on the intuition of engineers than presentation of a systematic method to locate and choose cut-off points for modules, which again is either a benefit or a problem, depending upon one's perspective.

5.1.3 Design Structure Matrix

A DSM [117] is typically used to organize product development tasks or teams to minimize unnecessary design iterations and thus help manage and speed up the development process. The DSM can also be used to define modules within a single product's architecture. In the component or function based DSM, also called architecture DSM, components or functions are placed on the row and column headers of the matrix. Components or functions are then mapped against each other and their interactions are marked in the matrix. One can also present spatial, energy, information, and material interactions of components or functions in a DSM as shown by Pimmler and Eppinger in [94] and also by Blackenfelt in [8]. The interactions can be represented with coupling coefficients -2, -1, 0, 1, or 2 depending on the strength of the relation and whether the relation is beneficial or undesired.

Once functions or components and their interactions are placed in the DSM, a clustering algorithm can be applied to group the functions or components so that the interactions within clusters are maximized and between the clusters minimized. The formed clusters are possible module candidates (Figure 20). There are many algorithms and one can develop one's own to suit the needs of a specific case. The basic idea of a clustering algorithm is to reorder the rows and columns so that all marks are as close to the diagonal as possible or form a tight cluster with other marks. The algorithm used in this study is developed by Thebeau [114]. This was chosen because it is a well defined computerized algorithm. The algorithm can result in overlapping modules or it may leave a function out of the final clustering, in which case it is up to the designer to decide how to handle them. The overlapping section could be for example duplicated and placed in both modules or forced to be only in one of the modules where the algorithm suggested it could be. For more about the component based DSM method, refer to [14].

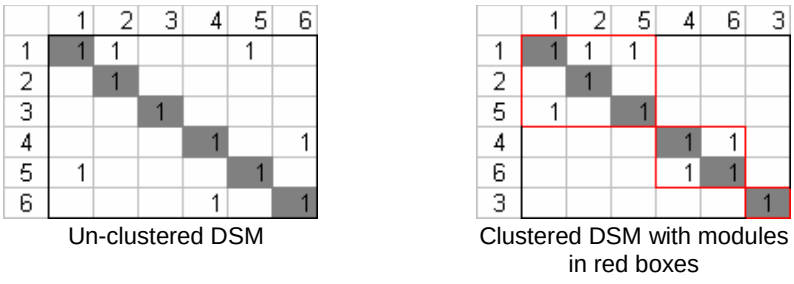


Figure 20 An exemplary DSM.

The DSM is designed especially for quick rearranging of the architecture based on the interface interactions. The method concentrates on the interfaces of the modules to simplify the design process and the apparent complexity of the product architecture. The component based DSM could be combined with the task and team DSMs to include the modularization in the rest of the design process planning. The method leaves more business oriented factors and product functionality up to the designer's judgment after first simplifying the architecture.

5.1.4 Comparative Analysis of the Methods

The goal of the comparison of the three methods introduced above is to get a user's perspective on how easy the methods are to use, how well they work for specific cases, and how repeatable they are. In addition, weaknesses were identified as basis for future work. The comparison and the results in this section are described in more detail in Publication II and [50].

The starting assumption was that the methods would give similar results for the modular architecture or at least identify a few key modules in the same way. This, however, ended up not being the case. Surprisingly the methods, tested on a total of 6 products in two case studies, gave practically no common solutions as to how to divide a structure into modules.

Further, the methods were applied on two families of two products, but since the methods are designed for single products, they did not identify common modules across the products in a family, thus sub-optimizing the family in order to optimize a single product [Publication II, 50]. This is interesting since one of the key goals of modularity is to gain scale and scope advantages by sharing components and thereby creating variety with less components in a product family. The common module heuristic helped the function structure heuristics to perform best in terms of finding the most commonalities between products.

All methods identify certain groups of functions, that should be combined into a module, in some particular way, but they do not agree on how many other functions these so-called *module cores* should have. The electro-mechanical products in the case studies all had a drive unit. One observation is that all methods identify the drive unit as a module (module core). The drive unit is typically a central part of a product and all methods suggest it should be bundled up as a module. However, the methods do not agree on the size of the module, i.e. what functions should be included in the drive unit module.

The different results are mainly due to the different assumptions of each method and the most suitable should be chosen according to the goal of the company. The function structure heuristics aim for simple interfaces (branching flow) and grouping sets of key functions into modules (dominant flow, conversion-transmission pair); The MFD, on the other hand, does not look at the interfaces between the functions but concentrates on the strategic aspects, possible benefits, of modularity such as ease of maintenance and reuse, which in turn are ignored by the other methods. The DSM is more similar to the function structure heuristics in that both aim for simple interfaces. The DSM, however, is run by a computer and it cannot identify the key functions of a product. In fact, the DSM can suggest overlapping or functionally infeasible solutions. This brings us to the subjectivity of the methods.

The repeatability of the three methods was analyzed by having 2 groups of 20 graduate students and engineers perform the methods in two separate case studies. The experiment set up is described in detail in Publication II and [50]. The goal was to analyze the subjectivity and objectivity of the methods. The more subjective a method, the less repeatable it is. The repeatability in the percentage of functions grouped in the same way. The DSM was left out of the repeatability study since it is a computer run algorithm. However, it is also not 100% objective since the algorithm [114] depends on the original order of rows in the matrix. The results are summarized in Table 2.

Table 2 The repeatability of modularity methods.

	Case 1	Case 2
Function Structure Heuristics		
Conversion transmission	90%	90%
Branching	80%	75%
Dominant	75%	60%
Function Structure Heuristics (family)		
Repetitive	81%	84%
Common	70%	63%
Unique	86%	83%
Modular Function Deployment		
	68%	85%

We see that the repeatability of each method is reasonable, but there is a disappointing lack of objectivity so great care must be taken if any are to be used. Conversion-transmission pair of the function structure heuristics has the highest repeatability. This is due to the clear definition of the rule. The unique function heuristic scored high on repeatability for the same reason. On the other side of the spectrum, the dominant flow heuristic received a low repeatability percentage due to its vague hard-to-understand definition, according to the research participants. The difference between the repetitive and common function structure heuristics is that the former is applied within a single product and the latter across different products. The results show how the choice of modules becomes more difficult when the choices must be made for a product family instead of just a single product.

The difference in the repeatability of the MFD in the two case studies is explained by the fact that in case study 1 the participants were not familiar with the product and in case study 2 the participants did not only know the product better but also had a chance to take it apart prior to the modularization [50]. The results suggest that the MFD may lead toward the existing solution if the engineer is familiar with it.

There is a correlation between the repeatability of the methods and the types of modules suggested. The more subjective a method the more feasible modules it suggests. This is due to the engineer's strong influence in the process. This can either be good or bad. One downside is that an engineer may be biased toward a solution e.g. the existing one. On the other hand, a more objective method can give more novel solutions, but they may not always be feasible such as a module with an axle and circuit board parts.

The ease of use is subjective. The heuristics require studying of the definitions, but the execution requires only a pen and paper, or a simple commonly used software such as some of the MS Office programs. The MFD requires interviewing several stakeholders in a company and is therefore more laborious to perform than the other two. The DSM

requires a clustering algorithm and some software, but once the un-clustered DSM is fed into the algorithm the clustering of even a larger matrix is immediate. The ease of use is a secondary goal of these methods and no more analysis is done in regard to it.

The difficulty of use and the weaknesses in repeatability are due to two main reasons:

1. The methods have insufficient rules on how to decide where an interface (module boundary) should be located.
2. The methods are designed mainly for single products.

The first weakness of the modularity methods needs improving. The definition of modularity almost always includes simple interfaces and isolated units, but the methods, except the DSM, do not address this rigorously enough. And even the DSM algorithm treats every interface connection as equivalent, which in unlikely the case. The function structure heuristics aim for simple interfaces, but since the heuristics are maximal heuristics and the rules fairly broad, many alternative modules are possible without violating the heuristics. The MFD has a step for interface design, but it is not detailed and does not address how to actually decide how to cluster some of the functions into modules if the module driver profiles are only weakly similar and to more than one module. Clearly, a better method for interface design is needed. Section 5.2, Flexible Interface Design, will introduce a method for this.

The second weakness is a problem only in designing multiple products. I will argue, however, that a company should be developing multiple products – product platforms and variants. The existing methods, for most part, optimize each product of the family and not the family (or platform). The main deficiency is in identifying the common parts of the family. DSM does not have a step for this at all. MFD has one driver out of many to identify common units across products, but this driver presumes that the commonality is predetermined. The function structure heuristics include three heuristics designed specifically for product family design, but as shown in Table 2, the repeatability is the poorest when trying to identify the commonalities across products. Thevenot and Simpson [115] also call for more specific definitions of commonality in their analysis of commonality metrics for platform design. A better method is needed for identifying common modules across products in a product family for platform design. Section 5.3, Identifying Common Modules, will introduce a new method for this.

5.2 Flexible Interface Design

Redesign is unavoidable. A product will need to be redesigned during the design iterations and later as new versions of the product are designed. Thomke [116] suggests modularity as one tool for improved flexibility, but modularity alone is not enough if the interfaces are not properly designed. Also Tatikonda [113] supports separating dependencies between module interfaces. As discussed earlier, a weakness of the modularity methods to date is the lack of interface design. Some simple heuristics exist, such as calculating the number of connection between modules [8, 12], but as Fixson [30] points out, different interactions have different intensities.

I developed a metric to assess the degree of complexity of different interface types. The metric is used in addition to a modularization method to determine module

boundaries. The metric is based on minimizing the redesign effort, if an adjacent module were to change. This robustness of a module to change makes the architecture more flexible in terms of design upgrades and other changes. The idea is similar as in [49], where an interface workload is mapped in a DSM based on the owner of the interface, except that in my approach we look at the flows at the interface, regardless of the owner.

The redesign effort metric introduced here is not meant for deciding the number or size of the modules alone. Our metric along with others, such as assembly-ability [11, 41], suppliers [56], team size [13], etc., are all important criteria to use in such a multi-criterion decision. In addition, emergent properties such as cost, weight, and performance type criteria must be considered [39, 42, 64, 119, 123]. The purpose of this metric is to help choose module boundaries so that future changes are as effortless as possible.

The metric is based on the interaction types at the interfaces. The interactions are the flow types such as solid material, gas, electrical energy (Table 3).

Table 3 Interface complexity values (per 1% change in the original flow value) for different flow types at two different companies. Values are not expected to be general across companies. (* is an average of sub-category metric values)

Flow category	Sub-category	Company 1 Injectors	Company 2 Sensors
Material	Solid	1.2	-
	Gas	-	0.3
Energy	Acoustic	3.1	-
	Electrical	1.2	0.5
	Mechanical General* Rot torque Translation	 1.0 1.0 1.0	-
	Pneumatic	1.3	-
	Thermal	1.8	0.3
Info	General*	0.8	1.0
	Content	1.2	1.7
	Bandwidth	0.4	0.2

The metric was developed and tested in two industrial case studies. The first case study involved two companies jointly developing medical injectors and the second a single company developing industrial process sensors. We interviewed multiple design engineers and system experts at the two companies. We asked them to evaluate the redesign effort needed if a flow at an interface were to change by a certain percentage. E.g. *How much redesign compared to the original effort is needed in this module if this input voltage (electrical energy) flow is increased by 20%?* We calculated the metric following the model in Figure 21. The metric is used on the linear portion of the model. The linear portion is an approximation of smaller discrete changes. Details about the methods and case studies can be found in Publications IV and V.

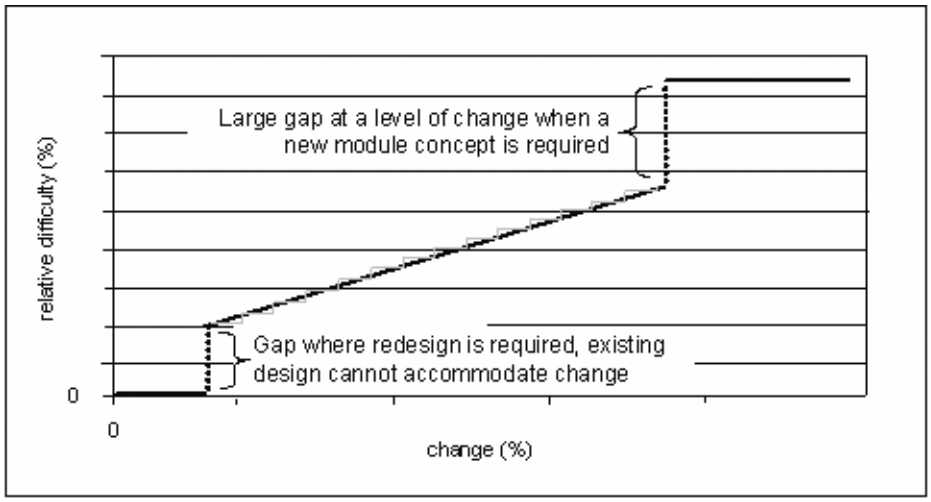


Figure 21 The general behavior of redesign effort vs. the change percentage.

We obtained a redesign effort metric for all the flow categories (Table 3). A number, e.g. 1.2, signifies that if the flow were to change by 1% the amount of effort needed to accommodate for the change is 1.2% of the original effort to design the particular component. As we can see, the values are different for each flow type as well as for each company. This was expected as we hypothesized that different interactions have different effects in terms of redesign effort. The values also depend on the company and product in question. For example, the second case study involves more software intensive products and therefore an information flow, especially information content, change is more difficult. The differences depend also on how the company sees themselves. The second case study company seems to feel more confident about their abilities to adapt to change than companies in case study 1. A closer look at the table, however, reveals that even though the values are different across the two cases, some generalizations can be made. For example, changing information flow bandwidth is considerably easier than changing information content. Also, electrical energy (typically voltage) is harder to change than information bandwidth. We believe this is due to the larger buffers typically used for bandwidth than for voltage. On the other hand, information content requires more design effort to change than an electrical flow.

The redesign effort metric values are used to determine module boundaries together with other criteria such as cost or supply chain requirements. In order to do this, the product is modularized using a modularization method. The method can be picked according to the company goals. The methods, as mentioned above, tend to give suggestions, not definitive answers as to where to draw the module boundaries. The metric is good for identifying critical interfaces in a product architecture. The larger the design effort complexity metric on a specific interface, the better it is to keep the interface within a module. And similarly, the smaller the design effort complexity metric at an interface, the better candidate the interface is to be at a module boundary. These are analogous to the tactics in software to aim to keep the high-bandwidth communication within a module and place low-bandwidth links between modules [6]. This eases the development of the modules since a team developing a module is more likely to handle the complex interfaces than if the interface was to be design by two separate teams developing separate modules. This is also supported by Sosa *et al*. [104].

The redesign effort metric can improve the flexibility, in terms of change readiness [91], of an architecture. Figure 22 shows a partial function structure of a gas sensor from the second case study with its modules defined using DSM. In addition, the inter-module redesign effort metric values are shown underlined and the intra-module redesign effort metric values in italics. It appears that this architecture could be improved by moving the module boundaries so that the most difficult interfaces are inside a module and simpler interfaces can be put to the module boundary instead. Looking at the gas sensor function structure one interface can be clearly seen as a difficult interface: the interface between the *Timing*-module and the *Processor*-module. This interface consist of five information flows (each 1.0) between the functions in each module has therefore a design effort complexity score of 5.0. If anything should change in one of these modules, it would cause major design also in the other module. From the redesign ease point of view, these two modules should be kept together as a single module. In addition we can combine the *power supply* with the *control heating* function and thereby simplifying the interfaces further. These two changes improve the total redesign effort metric sum of the architecture from 19.3 to 14.8 without making the product overly integral or violating modularization rules used and while keeping the architecture feasible. As mentioned above, this metric should be used together with other design criteria. It is worth noting that the redesign effort metric for an interface can be improved by very small changes that can still be in line with the other criteria. More examples of the benefits and use are in Publications IV and V.

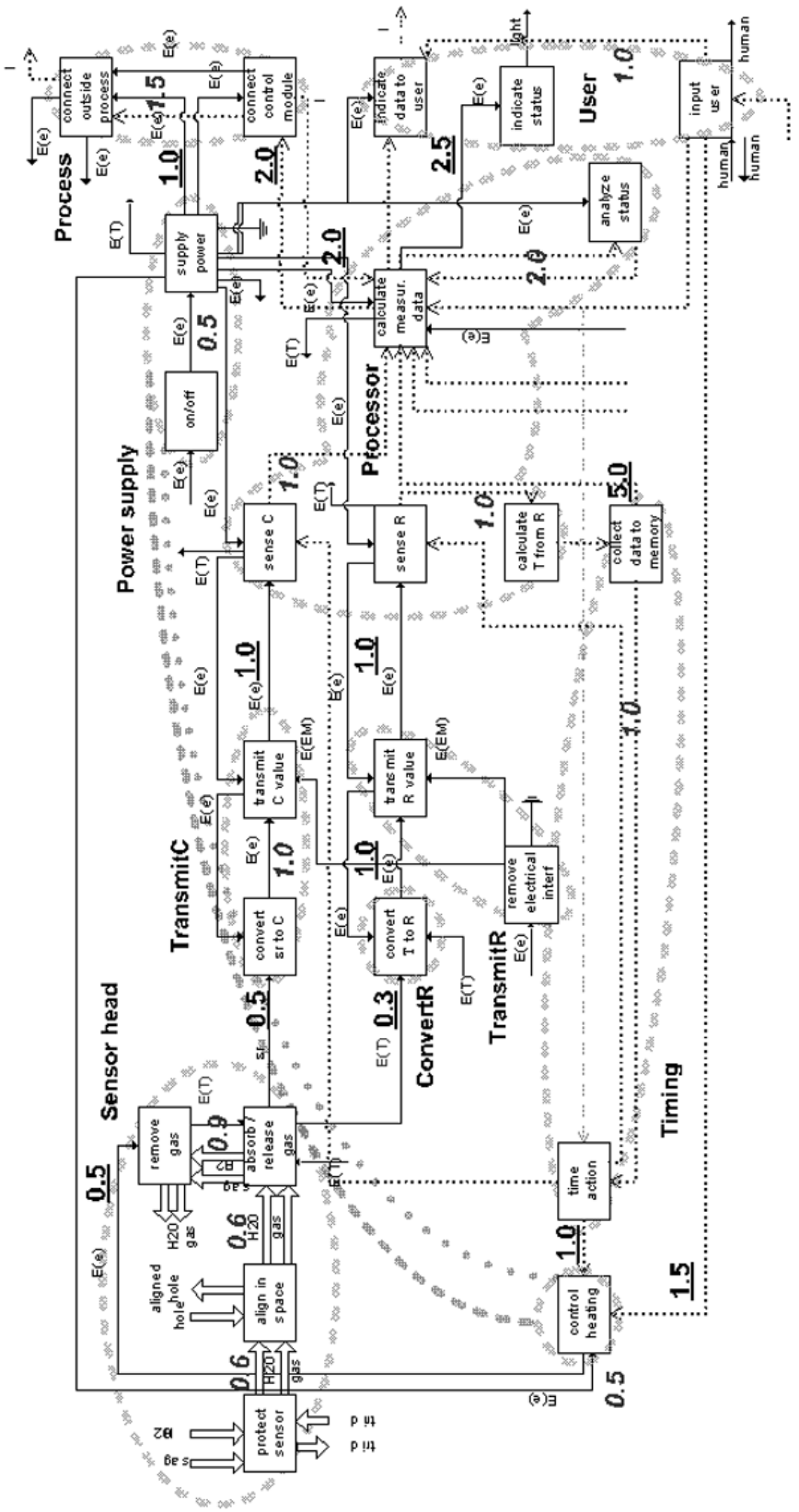


Figure 22 Partial function structure of a gas sensor used in the second case study. DSM defined modules shown in dashed lines. (Fig 5. in Publication III)

The results here were shown to be statistically from moderately to highly significant depending on the flow type. The significance can hardly be improved since the metrics are based on subjective estimates of design experts. The subjectivity adds noise to the results, but an attempt was made to overcome this deficiency by interviewing multiple experts with different backgrounds and asking several dozens of estimations per product.

5.3 Identifying Common Modules

In this section I describe a new quantitative method to evaluate module commonality. This, as all methods described in this thesis, is to be used together with other modularity and architecting methods. The methods, results, and analysis are described in more detail in Publication III and [51].

The method is based on measuring the "distance" between functions' inputs and outputs and clustering the functions into a dendrogram to visualize the possible common module candidates. Johnson *et al*. use also a Euclidian distance based dendrogram but for clustering materials based on their technical properties and aesthetics. [59] Also Pedersen [92] uses dendrograms to create product families, but his method is based on components in existing products and not applicable in the product architecture phase. Stake [107] uses a dendrogram approach to identify modules, but he identifies modules within a single product. My method identifies common modules both within and, more importantly, across products. Moreover, the method can be used at the early phases of development when only the requirements are known.

Further, this method is not restricted to comparing commonality at a single level of system hierarchy, but it can compare commonality across the hierarchies, including physical sub-system level and basic function level. We look at the similarity at different levels of hierarchy at the same time. This is different from Fellini's [29] approach, where each level is treated separately. Treating the hierarchies all at once is useful since it is often difficult to define the levels of decomposition.

Unlike many previous commonality measures [58, 115], this method sees platform and component commonality analysis not as a binary, common/not common choice, but as a more complex decision of degree of commonality. This makes the method not subject to the choice of what is common enough (Same function, but different power requirements? Same component but different color?) [115].

The following sub-sections will describe the use of the method first in the functional domain and then in the physical domain.

5.3.1 Commonality in the Functional Domain

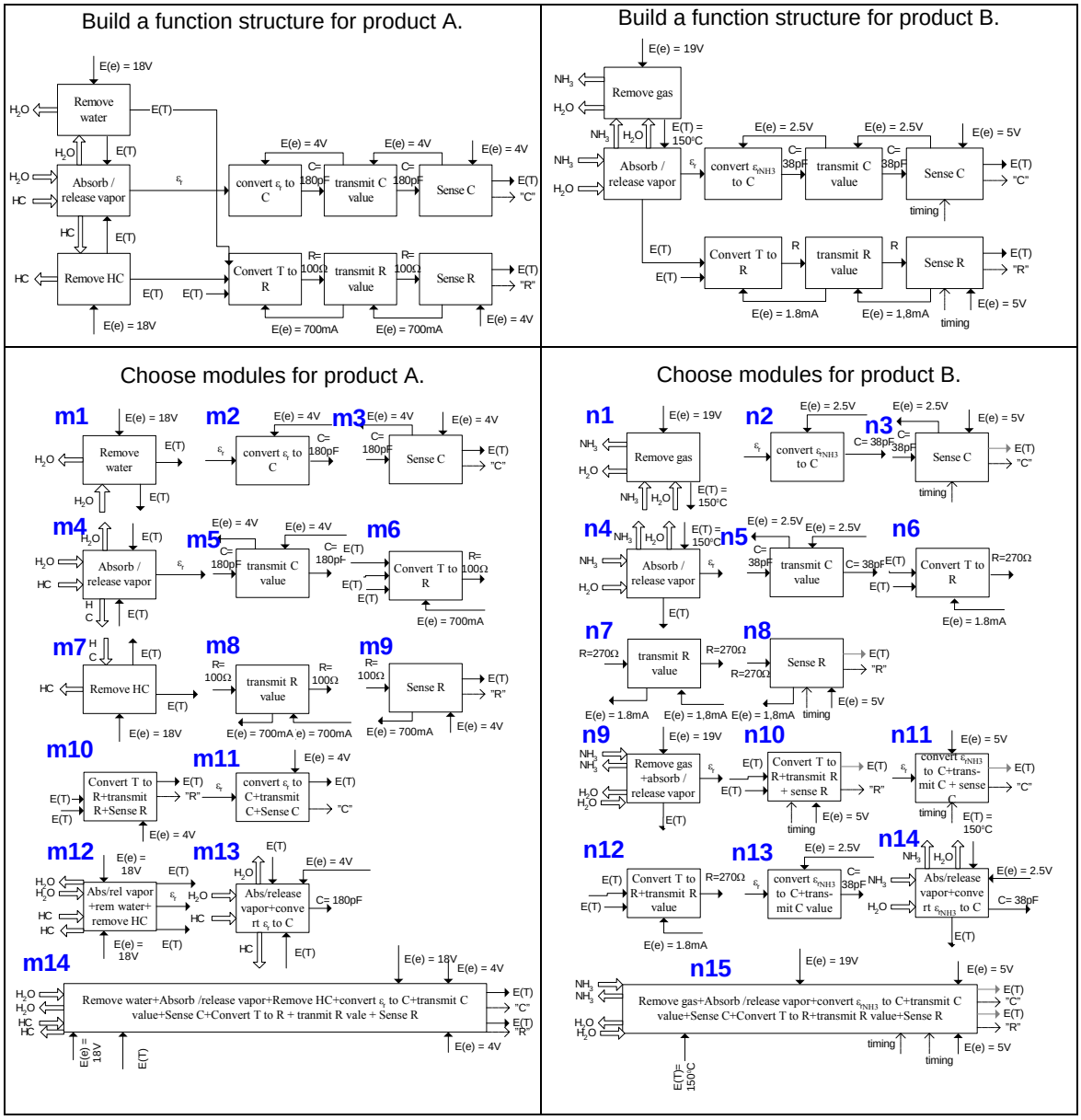
The analysis in most existing methods is often done at a component or feature level. McAdams and Wood [73] go further to the functional level of a product in their quantitative similarity metric. They base their similarity on the similarity in the vocabulary [48] used to describe the functions of various products. My method also uses the same standard vocabulary but in addition, my method measures the distance between the function inputs and outputs, using ratio scales, making the method more rigorous than the previous methods.

The distance measure is an n-dimensional Euclidian distance based on the input and output flow values of the functions. The basic steps include characterizing the input and output flows of each function and function groups with units e.g. 12W and 3W, or 1200

baud and 2400 baud, normalizing them to be between 0 and 1 by dividing by the maximum of each flow type, and comparing the functions and groups of functions pair wise to obtain the "distance" between them. The input and output values can be based either on the technical specifications derived from the customer requirements or actual flow values, if the project is a redesign of an existing product.

Each flow type is treated separately and combined at the final distance calculation phase. This approach presumes all flow types are comparable in a dimensionless space. This distance defines the commonality, or lack of it, to aid in common module selection for platforms. Table 4 illustrates the steps for measuring the distance D between functions. A family of two process sensors is used as an example. The detailed equations can be found in [51].

Table 4 Steps for measuring module commonality in the functional domain.



Characterize the inputs (x_i) and outputs (y_i) for product A.	Characterize the inputs (x_i) and outputs (y_i) for product B.
m1 x_1^{m1} = 18V	n1 x_1^{n1} = 19V
m1 y_1^{m1} = 0V	n1 y_1^{n1} = 0V
m1 x_2^{m1} = 100%	n1 x_2^{n1} = 100%
m1 y_2^{m1} = 100%	n1 y_2^{n1} = 100%
(repeat for all modules and inputs)	*(repeat for all modules and inputs)*

Calculate the distance $D(i,j) = \sqrt{\overrightarrow{\Delta x} \cdot W \cdot \overrightarrow{\Delta x}^T}$, where

$$\overrightarrow{\Delta x} = \begin{Bmatrix} x_1^i - x_1^j \\ x_2^i - x_2^j \\ x_3^i - x_3^j \\ \vdots \\ x_N^i - x_N^j \\ y_1^i - y_1^j \\ y_2^i - y_2^j \\ y_3^i - y_3^j \\ \vdots \\ y_M^i - y_M^j \end{Bmatrix}$$ and $$W = \begin{Bmatrix} W_{1_{IN}} & & & & & & & & \\ & W_{2_{IN}} & & & & & & & \\ & & W_{3_{IN}} & & & & 0 & & \\ & & & \ddots & & & & & \\ & & & & W_{N_{IN}} & & & & \\ & & & & & W_{1_{OUT}} & & & \\ & & & & & & W_{2_{OUT}} & & \\ & & 0 & & & & & W_{3_{OUT}} & \\ & & & & & & & & \ddots \\ & & & & & & & & W_{M_{OUT}} \end{Bmatrix}$$, and where

$W_{k_{IN}}$ is the weight of the input type *k*, $W_{k_{OUT}}$ is the weight of the output type *k*, *N* is the number of input types, and *M* is the number of output types. The weights include normalization terms.

We developed a total of four versions of the distance algorithm; one in Publication III and three in a continuation effort [51]. We concluded that the most effective algorithm is one where we included preference functions [3] (included in the weight in the above equations) for the flow types to handle the non-additive nature of flow difference as the value of the flow grows. E.g. the difference of 3V is far more significant between 1 and 4 Volts than between 1001 and 1004 Volts. In addition we added a weight to each flow type since some flows are more challenging than others as demonstrated in Section 5.2, Flexible Interface Design. The preference functions and weights must be defined carefully. This is where engineering judgment enters the process. So far, in this research I have found that in practice, product functions and components are either very similar or very different, and thus the method is not sensitive to the weight and preference function, at least when the preference function is a power function as described here. The results here apply for at least transformation functions $f(x)=x^{1/2}$, $f(x)=x^{1/3}$, and $f(x)=x^{1/4}$.

Once the distances between all module pairs are calculated, the modules are clustered into a dendrogram. Figure 23 shows the dendrogram for the above example. More detailed discussion of the results can be found in Publication III as well as in [51].

It is now up to the designer to decide where to set the cut-off line (yellow dashed line in Figure 23) for module commonality. The choice depends on e.g. the acceptable performance losses and the cost of over design. The purpose of the dendrogram is to ease the designer's choice on common functions by clustering similar functions together. This method is useful in the platform development prior to e.g. platform optimization methods that require predefined common modules.

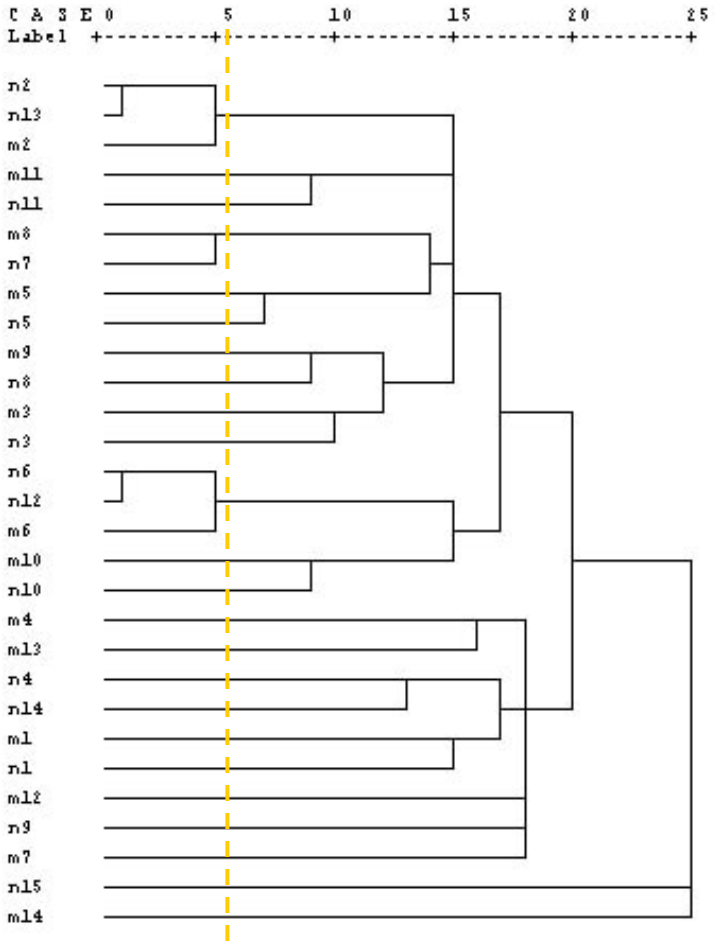


Figure 23 A dendrogram of modules for products A and B clustered according to their distance from one another.

5.3.2 Commonality in the Physical Domain

The algorithm described above can also be used in the physical domain with small modifications. In the physical domain approach the products need to be decomposed to assembly level, not to the abstract function level. The function inputs and outputs above are replaced by component, or sub-assembly, input and output requirements and other attributes such as weight or volume, when appropriate.

For example, a set of miniature drive units and their components both alone and in combinations with other components (motors, gears, and linear actuators) for a product family can be compared by using e.g. the voltage and torque specifications as well as the maximum volume of the drives (Table 5). Notice that the table includes both individual components and multiple combinations of components. This represents an example where a company has multiple products that use miniature drives and that have been designed independently and where the company has decided to reengineer the products and save costs by commonalizing some of the drives.

Table 5 Inputs for miniature drive commonality analysis.

Component or component combination	Voltage (V)	Speed (rpm)	Current (mA)	Torque (mNm)	Volume (mm^3)	Lin. force (N)
DC Motor A1	12	12950	13	0.2	2084.6	*0*
Gear a1	*0*	*0*	*0*	4.9	1442.3	*0*
Gear a2	*0*	*0*	*0*	12.1	1442.3	*0*
MotorGearA1a1	12	*0*	13	4.9	3526.9	*0*
MotorGearA1a2	12	*0*	13	12.1	3526.9	*0*
DC Motor A2	12	10500	30	0.2	2253.6	*0*
MotorGearA2a1	12	*0*	30	4.9	3695.9	*0*
MotorGearA2a2	12	*0*	30	12.1	3695.9	*0*
DC Motor B1	12	6000	6	0.4	2566.6	*0*
DC Motor B2	6	10000	8	0.1	2117.5	*0*
Gear b1	*0*	7000	*0*	20.1	4809.9	*0*
Gear b2	*0*	7000	*0*	20.1	4809.9	*0*
MotorGearB1b1	12	*0*	6	20.1	7915.5	*0*
MotorGearB1b2	12	*0*	6	20.1	7915.5	*0*
Gear b3	*0*	7000	*0*	20.1	4263.9	*0*
MotorGearB2b3	6	*0*	8	20.1	6757.1	*0*
LinMotor D1	12	*0*	67	*0*	5651.6	7.0
LinMotor D2	12	*0*	240	*0*	33934.2	220.0
LinMotor D3	12	*0*	113	*0*	13151.3	12.0
DC Motor C1	12	10000	13.6	3.3	3729.2	*0*
Screw E1	*0*	*0*	*0*	*0*	4401.6	*0*
ComboA1a1E1	12	*0*	13	*0*	9912.3	30.8
ComboA1a2E1	12	*0*	13	*0*	9912.3	75.8
ComboA2a1E1	12	*0*	30	*0*	10176.4	30.8
ComboA2a2E1	12	*0*	30	*0*	10176.4	75.8
ComboB1b1E1	12	*0*	6	*0*	17383.7	126.3
ComboB1b2E1	12	*0*	6	*0*	17383.7	126.3
ComboB2b3E1	6	*0*	8	*0*	19477.7	126.3
Screw E2	*0*	*0*	*0*	*0*	9390.0	*0*
ComboA1a1E2	12	*0*	13	*0*	19186.9	15.4
ComboA1a2E2	12	*0*	13	*0*	19186.9	37.9
ComboA2a1E2	12	*0*	30	*0*	19656.4	15.4
ComboA2a2E2	12	*0*	30	*0*	19656.4	37.9
ComboB1b1E2	12	*0*	6	*0*	19277.4	63.1
ComboB1b2E2	12	*0*	6	*0*	19277.4	63.1
ComboB2b3E2	6	*0*	8	*0*	22021.8	63.1

I apply the algorithm using equal weight (1) for all inputs and a cubic root transformation of all values. Figure 24 shows how the clustering algorithm separates the different component types into logical clusters. For example, the different Motor-gear-screw combinations are separate from the Motor-gear combinations or the single components. Further, two of the linear actuators with similar specifications to the Motor-gear-screw combinations are clustered close by indicating that the algorithm can be used

to identify similar components or sets of components. The dendrogram aids in deciding which transmissions can be replaced by new common modules. This eases the product family redesign and can be used to create a few alternative architectures to be compared against other criteria as shown in Section 6 Evaluating Platform Architectures.

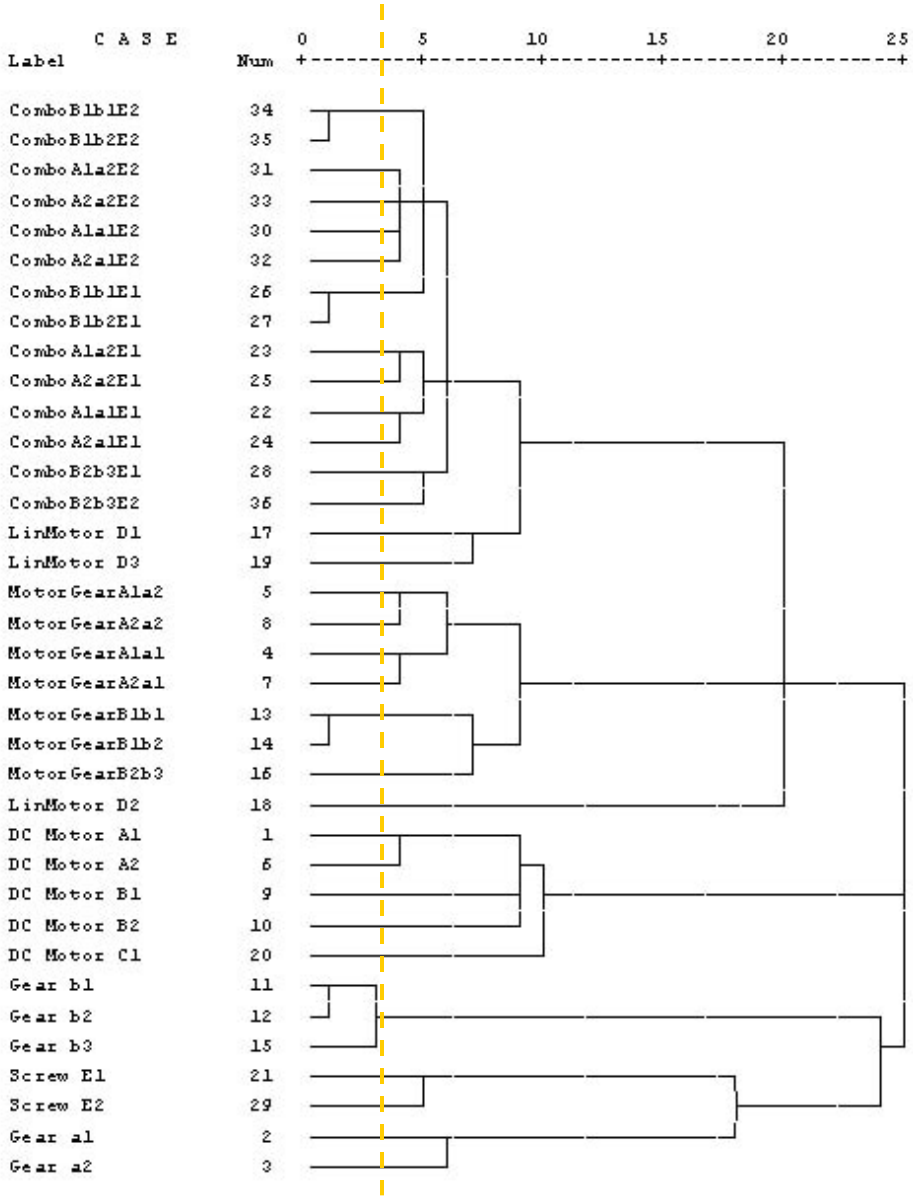


Figure 24 A dendrogram of drive components clustered according to their distance from one another.

The dendrogram clusters similar functions in a logical way in the examples here as well as in Publication III and [51], and thus seems to work as desired, but the applicability for all possible applications is yet to be tested in real PD projects.

6 EVALUATING PLATFORM ARCHITECTURES

Platform architecture evaluation is a more challenging task than evaluating a single product architecture since a platform must effectively support multiple product variants over a prolonged period of time. The platform methods introduced in section 4.3, Methods, typically define a platform based on a few criteria such as cost, commonality, and performance. In addition, there is work in platform architecture evaluation. De Weck and Chang [23] use a Pareto frontier to aid in architecture concept selection. Their method optimizes performance in respect to lifecycle costs. Kota *et al*. [63], on the other hand, present a benchmark method to compare own platform to competitors platform. Their method evaluates a platform based on how well the non value adding components are shared in a platform. Also these methods deal with only a limited set of criteria, but a platform can not be properly evaluated outside the company and business context.

Kristjansson and Hildre [65] introduce a platform assessment tool for evaluation of the strategic fit of a platform. They include multiple criteria, but the tool lacks the technical detail needed in the actual platform development.

There has also been excellent work in developing product concept evaluation methods, such as Pugh's selection process, concept screening and scoring, or trade-studies [87, 117]. However, these methods are for evaluating a single product concept. A platform concept has different requirements due to its longer lifetime and that it must enable several derivative products.

Crawley *et al*. [19] discuss how an architecture should be evaluated based on multiple *ilities*. A "good" architecture is flexible, scalable, maintainable, recyclable, etc. I will present here a method that helps assess a modular platform in a larger context based on multiple criteria including commonality, performance in terms of meeting customer requirements as well as many other ilities.

The platform architecture assessment tool that makes use of the work of many others in the field of modularity, platforming, and general product development. The evaluation metrics are from three sources: six executive-level system engineers with an average of 17 years experience, the co-author's [Publication VI] personal experiences of platform development over the last 10 years with over two-dozen platforms and the personal mistakes learned from inadequate preparation (e.g., inadequate preliminary assessment), and the literature for platform metrics used by others, such as by Ericsson and Erixon [27], Blackenfelt [8].

The tool is focused on the early platform architecture phase, before proof-of-concept prototyping. However, it can also be used subsequently for platform refinement when more data becomes available. The tool is meant to aid in modular platform architecture development, evaluation, and as a communication tool to upper management as well as between different stakeholders. Due to the approximate nature of the summed scores, the tool works as a guide and not an absolute measure of platform "goodness".

The tool consists of 19 criteria [Publication VI] that are grouped into six categories: customer satisfaction, variety, after-sale, organization, flexibility, and complexity. Each metric is evaluated using a merit scale of {0, 3, 5, 7, 10}, where 0 is the worst and 10 the best. This is analogous to an A-F grading scale. The scores are absolute scores, where the 10 is a theoretical maximum and may not always be achievable. A competitor benchmark will help establish what level to aim for.

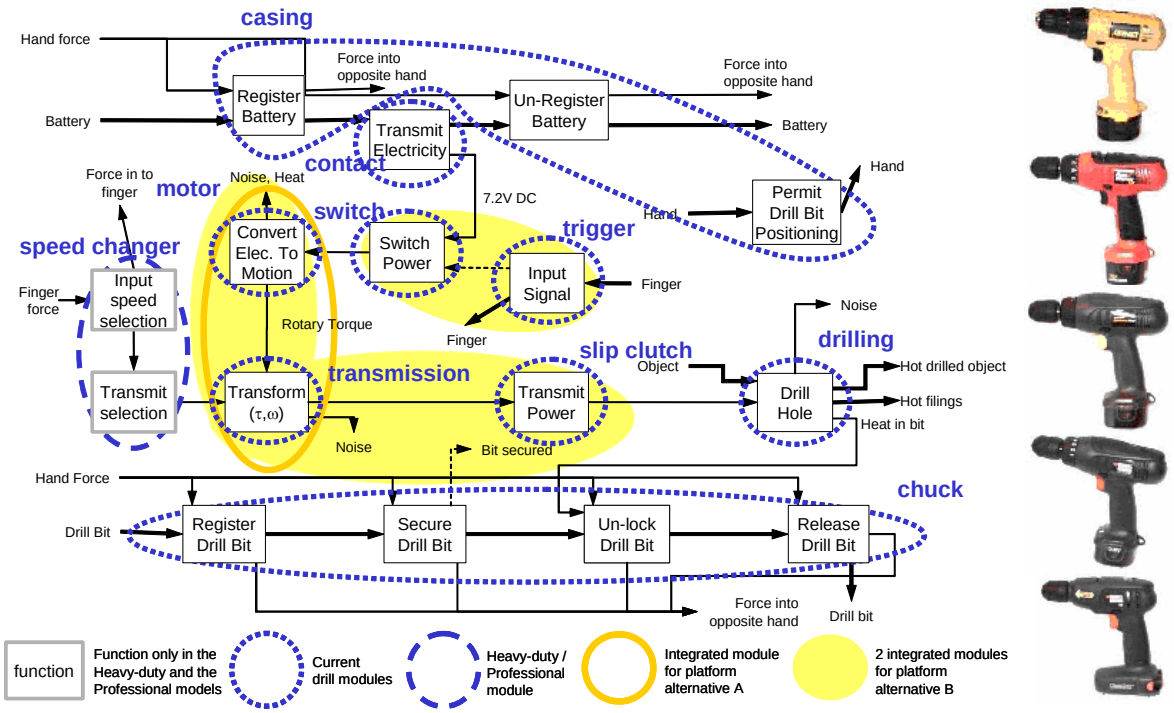


Figure 25 Case study family of drills and their family function structure, with modules shown.

We use a family of five different cordless drills: professional, heavy-duty, value brand, home-use, and a low price model to demonstrate our method (Figure 25). Table 6 shows the current cordless drill platform– in detail as an example. The individual metric calculations for the drill are in Publication VI. In addition, we will show results for platform alternatives A and B, also shown in Figure 25. We intend, that this assessment tool will be used to evaluate multiple alternative modular platforms

Table 6 Summarized scores for the three alternative platforms.

	CURRENT	A	B
Overall Multi-Criteria Platform Assessment	**Score**	**Score**	**Score**
	8.0	**7.9**	**7.5**
Platform Scorecard	Score	Score	Score
Portfolio Customer Satisfaction	**7.7**	**7.5**	**8.0**
Product Variety	**8.5**	**8.5**	**7.9**
After Sale Support	**8.9**	**8.9**	**8.7**
Organizational Alignment	**6.3**	**6.0**	**5.1**
Upgrade Flexibility	**9.6**	**9.5**	**8.2**
Development Complexity	**7.2**	**7.2**	**7.2**
	8.0	**7.9**	**7.5**
Portfolio Customer Satisfaction Scorecard	Score	Score	Score
Cost - Worth Distribution	8.0	7.6	7.2
Portfolio Customer Needs	7.3	7.3	8.8
	7.7	**7.5**	**8.0**
Product Variety Scorecard	Score	Score	Score
Planned Upgrade Carryover	9.7	9.7	8.3
Common Modules	5.9	5.9	5.4
Specification Variety	10.0	10.0	10.0
	8.5	**8.5**	**7.9**
After Sale Support Scorecard	Score	Score	Score
Partitioning for Reliability	8.6	8.5	7.9
Partitioning for Service	10.0	10.0	10.0
Environmental friendliness	8.1	8.1	8.1
	8.9	**8.9**	**8.7**
Organizational Alignment Scorecard	Score	Score	Score
Ease of Assembly	4.6	5.0	4.5
Aligned with the Organization	5.4	4.6	1.5
Make-Buy	9.6	8.8	8.6
Testability	5.4	5.5	5.7
	6.3	**6.0**	**5.1**
Upgrade Flexibility Scorecard	Score	Score	Score
Unknown Isolation	10.0	10.0	10.0
Change Flexibility	9.1	9.0	6.4
	9.6	**9.5**	**8.2**
Development Complexity Scorecard	Score	Score	Score
Function and Form Alignment	9.7	9.6	9.5
Interface Flexibility	10.0	10.0	10.0
Anti-Synergy Avoidance	0.0	0.0	0.0
1 DOF Adjustments	10.0	10.0	10.0
Limited Extremes	6.6	6.6	6.6
	7.2	**7.2**	**7.2**

Table 6 summarizes the platform assessment of all three alternative platforms. The individual metric scores can be summed (weighted sum, same approach as in [117] for product concept selection) to obtain first the sub-category scores and then the overall platform scores. The weight is based on the metric's contribution to the company profit. The current cordless drill family platform receives a score of 8.0 indicating that the platform is fairly well designed. It is better than the two other alternatives A and B that

received total platform scores 7.9 and 7.5 respectively. The overall score, however, is a rough estimate and the difference between 8.0 for the current platform and 7.9 for the alternative A is probably not significant. The true value of the analysis is in the sub-category scores.

The current platform received the highest score in flexibility. This is primarily due to the fact that the drill market is mature and no significant changes are expected. The maturity of the market is taken into account by using a low weight in the related metrics. The current platform received the lowest score in the category organization alignment. The assembly score is low. This is typical, but the score could be easily improved by adding better aligning features and eliminating a few screws.

The alternative A performed similarly to the current platform. The rank order of category scores for platform alternative A is the same as for the current platform, but the scores are inferior. The difference in scores, however, is an important indicator that our tool has enough resolution to separate two very similar architectures. The current and alternative A architecture differ by only one module.

The more integral alternative B received different scores. It performed significantly worse in flexibility and variety. This was expected, since the more integral design has larger modules that are more difficult to change if needed. The alternative B received a higher score than the other two platform alternatives in sub-category customer satisfaction. This is because many customer requirements such as weight and performance are easier to optimize with an integral design. More details are found in Publication VI.

The tool introduced above is meant for evaluating alternative modular platforms on multiple criteria (ilities). The criteria should be aligned with the company strategy. A company may choose to weigh ease of service, for example, over other criteria such as variety. The tool helps on focusing on strategic goals of the platform. Moreover, the tool can be used to benchmark one's own platform to a competitor's by reverse engineering the competitor's products to investigate the limits of the competitor's platform as well as the possibilities of one's own. The tool is also useful in differentiating from the competition. Finally, the scorecard serves as a communication tool between the different stakeholders and to upper management in pointing out the strengths and weaknesses of the platform.

7 CONCLUSIONS

I have introduced the multiple aspects of platform architecture design from the theory of product architecture and product architecture representation to the advantages and disadvantages of modular product architectures and to practical tools for platform design. In summary, this thesis has shown how to define common platform modules with easy to redesign interfaces as well as how to choose a platform alternative that is well aligned with the company strategy. In this section I will tie the tools to the platform architecture development process. Figure 26 shows the basic steps of a platform design process supplemented with the new methods developed here. This process follows the company portfolio strategy development, and the resulting modular product architecture is delivered for detail core module design (Figure 4).

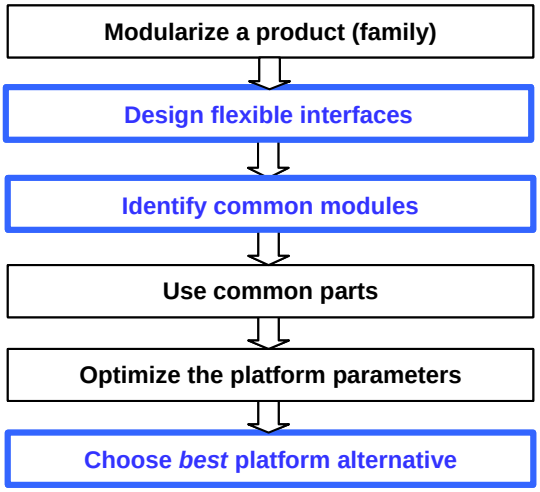


Figure 26 The steps of a modular platform architecture design process. This thesis's contribution in blue and bold boxes.

One of the research questions was to identify the biggest gaps in the modular platform development methods to date. I observed through investigation of existing methods that the platform or modular design methods are meant for single products and do not therefore properly enable product variety through a product family. Further, the current methods identify module "cores" only leaving the final module boundary definition to the designer, and use only a limited set of evaluation criteria. From these, I identified two major gaps in the current state of research: (1) lack of tools for interface design and (2) lack of design rules for how to choose the common platform components, and developed methods to fill these gaps. In addition, I recognized the need for platform architecture evaluation in the larger company context and developed a tool for that. These missing steps are added to the general modular platform architecture development process (Figure 26).

Another goal of this research was to develop a way to describe a module interface complexity quantitatively in order to fill in the first gap identified. I developed a metric to aid in designing flexible interfaces. The new approach was to look at the interface complexity as described by the material, energy, and information flows flowing through the interface. The flexibility is defined as ease of module redesign if an adjacent module

were to change. I showed how the metric has different values for different flow types at the interfaces. For example, information bandwidth is easier to change than electrical energy, which in turn is easier than information content change. I showed how the metric, used together with a modularization method, where drivers such as strategic modularity and other design criteria can be considered, can render a more flexible architecture without violating other design rules. The metric is evaluated by using it on two case studies. The research results were in agreement with the system design experts interviewed. The metric was shown to apply within a company but not across companies due to the different industries the two case study companies operate in. To date, there was no tool for estimating interface design effort complexity, and now the new metric will aid in designing products and modular platforms that are quicker to adapt to future changes than without the tool.

Once individual products are decomposed into modules according to criteria most suitable for the company, and the interfaces are properly defined, the next step in modular platform design is to identify possible commonalities in the product family in order to use common modules in more than one product and thereby saving design and manufacturing costs. The product component and function commonality analysis thus far involved simple binary decisions of common/not common, but I introduced an algorithm that takes into account possible degrees of commonality. This is an answer to the research question related to improving the common module identification and an attempt to fill the second gap in the existing methods. This new algorithm can be applied both in the physical and the functional domain and at any, and even mixed, levels of hierarchy. Furthermore, the algorithm is multidimensional and thus not limited to a single measure for commonality analysis. The algorithm is shown to provide design support through real examples in both the functional and physical domain.

As the common module candidates are identified, the interface flexibility metric can be applied again, if desired. After this, the common modules are chosen from the possible candidates by (a) calculating the estimated cost of over design and the savings from commonality, if a low functionality module is over designed in order to make it common with another module; or (b) estimating the acceptable performance loss, if a high functionality module is replaced by the low functionality module in order to make it common with another modules.

Once the common modules have been chosen, one can apply one of the optimization or other pre-existing methods described in Section 4.3.2, Module Based Platform Methods. The methods involve one or a set of platform parameters that are optimized on one or a few criteria. However, in this thesis the goal was to evaluate the “goodness” of a modular platform and its fit to the overall company strategy and not just the goodness related to a few criteria. Modularity and modular platform architectures must be evaluated in relation to the rest of the company operations and strategy. Just as with single concept selection, platform selection must also be done carefully by using multiple criteria. I showed a tool for platform architecture evaluation. The tool consists of 19 criteria and the usage of it as well as its resolution to differentiate between similar architectures was shown via an example. The tool helps in developing and evaluating a modular platform architecture. It helps a company focus on their strategy and benchmark one’s own platform to the competitors’. It also serves as a communication tool for upper management as well as between different stakeholders.

The modular platform development process is improved in this thesis, but future work is still required. The next step is to continue applying the developed methods in an industrial context and in multiple companies. Now each method was shown to apply in a few companies, but further validation of usability and effectiveness is needed to advance to the third and last stage of Blessing *et al*.'s [10] design research methodology.

A few interesting questions arose during this research but were left outside the scope of this thesis. One of them was, how much of the interface complexity metric can be generalized over companies within the same industry and across industries? This requires multiple case studies but could possible be accomplished over the course of several years. Further, the module commonality calculation algorithm is designed to have discrete numbers as inputs, but does not allow for input and output ranges such as 100-120A or max 200°C. The algorithm is already useful as such but would benefit if the inputs could be expressed in more ways. Other interesting questions are related to the platform architecture evaluation. Are there other metrics that should be considered? What is the best way to weigh the metrics? The former question was partially answered since we tried to be as exhaustive as possible and approached the issue from three directions: company expert interviews, platform design experience, and literature review. However, since industries, companies, market situations, etc. are different, it is only logical that the set of metrics and their weights are also different for specific cases. The main point, however, remains – multiple criteria involving multiple stakeholders must be used. All in all this thesis provides a suggestion for a modular product platform development process that is more advanced in ways listed above than methods so far, but the area of modular product platform design can be further explored and improved.

This thesis has shown how to define common platform modules with easy to redesign interfaces as well as how to choose a platform alternative that is well aligned with the company strategy. This modular platform process, and the set of tools developed here, will hopefully be used to make product development more effective in industrial companies. An effective platform can bring many benefits from cost savings due to module commonality to reducing time to market, but a company can only benefit from these if the platform is appropriately designed.

REFERENCES

1. Aarnio, J. *Modularization by Integration: Creating Modular Concepts for Mechatronic Products*. Doctoral Thesis. Tampere University of Technology. 2003.
2. Allen, K. R. & Carlson-Skalak, S. Defining product architecture during conceptual design. *In Proc of the ASME Design Engineering Technical Conferences.* Atlanta, GA. 1998.
3. Antonsson, E. K. & Otto, K. N. Imprecision in Engineering Design. *ASME journal of Mechanical Design*. Vol 117(B). 1995.
4. Association of National Advertisers. Prescription for New Product Success. New York. 1984.
5. Baldwin, C. Y. & Clark, K. B. *Design Rules: The Power of Modularity Design*. MIT Press. pp.471. 2000. ISBN 0262024667.
6. Bass, L., Clements, P., & Kazman, R. *Software architecture in practice*. 2nd ed. Addison-Wesley. 2003. ISBN 0-321-15495-9.
7. Benini, L. & de Micheli, G. System-level power optimization: Techniques and tools. *in Proc International Symposium on Low-Power Electronics Desig*n, pp. 288–293. San Diego, CA. 1999.
8. Blackenfelt, M. *Managing complexity by product modularization*. Doctoral Thesis. Dept. of Machine Design. Royal Institute of Technology. Stockholm. 2001.
9. Blanchard, B. S. & Fabrycky, W. J. *Systems Engineering and analysis*. 3rd ed. Prentice Hall, Upper Saddle River, NJ. 1998. ISBN 0-13-135047-1.
10. Blessing L.T.M., Chakrabarti A., & Wallace K.M., An overview of descriptive studies in relation to a general design research methodology, in: *Designers - the Key to Successful Product Development*, E. Frankenberger, et al (eds.) Springer Verlag, 1998, pp 42-56.
11. Boothroyd, G., Dewhurst, P., & Knight, W. *Product Design for Manufacture and Assembly*, 2nd ed. Marcel Dekker Inc, New York. 2002.
12. Braha, D. & Maimon, O. The measurement of a design structural and functional complexity. *IEEE Transactions on systems, man, and cybernetics*. Part A. Vol 28. No 4. pp.527-535. July 1998.
13. Braha, D. Partitioning tasks to product development teams. *In Proc of ASME Design Engineering Technical Conferences*. Montreal, Canada. 2002.
14. Browning, T. R. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management*. Vol 48. No 3. pp. 292-306. 2001.
15. Camuffo, A. Rolling Out a "World Car": Globalization, Outsourcing and Modularity in the Auto Industry. Working Paper. International Motor Vehicle Program, Massachusetts Institute of Technology. 2001.
16. Conner Seepersad, C., Hernandez G., & Allen J. K. A quantitative approach to determining product platform extent. *In Proc of ASME Design Engineering Technical Conferences*. Baltimore, MD. 2000.
17. Conner Seepersad, C., Mistree F., & Allen, J. K. A quantitative approach for designing multiple product platforms for an evolution portfolio of products. *In Proc* of *ASME Design Engineering Technical Conferences*. pp 593-602. Montreal, Canada. 2002.

18. Coulter, S. L., McIntosh, M. W., Bras, B., & Rosen, D. W. Identification of limiting factors for improving design modularity. *In Proc of ASME Design Engineering Technical Conferences*. Atlanta, GA. 1998.
19. Crawley, E., de Weck, O., Eppinger, S., Magee, C., Moses, J., Seering, W., Schindall, J., Wallace, D., & Whitney, D. The influence of architecture in engineering systems. Paper presented at *the MIT Engineering Systems Symposium*. Cambridge, MA. March 29-31, 2004. http://esd.mit.edu/symposium/monograph.
20. Cutherell D. Product architecture. In: The PDMA handbook of new product development. Rosenau M., Griffin A., Castellion G., and Anschuetz N. (eds). John Wiley & sons.1996.
21. Dahmus J. B., Gonzales-Zugasti J. P., & Otto K. N. Modular Product Architecture. *In Proc of ASME Design Engineering Technical Conferences*. Baltimore, MD. 2000.
22. Dahmus J. & Otto, K. Incorporating Lifecycle costs into Product Architecture Decisions. *In Proc of ASME Design Engineering Technical Conferences*. Pittsburgh, PA. 2001.
23. De Weck, O. L. & Chang, D. Architecture trade methodology for LEO personal communication systems. *In Proc AIAA 20th International Comm Satellite Systems Conference*. Montreal, Canada. 2002.
24. De Weck, O. L., Suh, E.S., & Chang, D. Product family and platform portfolio optimization. *In Proc of ASME Design Engineering Technical Conferences*. Chicago, IL. 2003.
25. Dori, D. *Object-Process Methodology*. Springer. 1998. ISBN 3-540-65471-2
26. Ethiraj, S. K. & Levinthal, D. Modularity and innovation in complex systems. Management science. Vol 50. No 2. pp 159-173. February 2004.
27. Ericsson, A. & Erixon, G. *Controlling design variants: Modular product platforms*. ASME press, New York, NY. pp145. 1999. ISBN 0-87263-514-7.
28. Fellini, R. Kokkolaras, M., Papalambros, P., & Perez-Duarte A. Platform selection under performance loss constraints in optimal design of product families. *In Proc of ASME Design Engineering Technical Conferences*. pp 593-602. Montreal, Canada. 2002.
29. Fellini, R. Kokkolaras, M., & Papalambros, P. Y. A rigorous framework for making commonality and modularity decisions in optimal design of product families. *In Proc of International Conference on Engineering Design*. Stockholm. 2003.
30. Fixson, S. Methodology Development: Analyzing Product Architecture Implications on Supply Chain Cost Dynamics. Presented at *the 5th Conference on Technology, Policy, and Innovation "Critical Infrastructures"*. Delft, The Netherlands. 2001.
31. Fixson, S. K. *The multiple faces of modularity – a literature analysis of product concept for assembled hardware products*. Technical report. 03-05 Industrial & Operations engineering. University of Michigan, Ann Arbor, MI. 2003.
32. Fixson S. K. & Clark J. P. On the link between modularity and cost – a methodology to assess cost implications of product architecture differences. *IEEE International Engineering Management Conference*. pp. 131-136. St John's College, Cambridge, UK. 2002.
33. Fujita, K., Takagi, H., & Nakayama, T. Assesment method of value distribution for product family deployment. *In Proc of International Conference on Engineering Design*. Stockholm. 2003.

34. Georgiopoulos P., Fellini R., Sasena M., & Papalambros P. Y. Optimal design decisions in product portfolio valuation. *In Proc of ASME Design Engineering Technical Conferences*. pp 593-602. Montreal, Canada. 2002.
35. Gershenson, J. K., Prasad, G. J., & Zhang, Y. Product modularity: measures and design methods. *Journal of Engineering Design*. Vol 15. No1. pp. 33-51. Feb 2004.
36. Gershenson, J. K., Prasad, G. J., & Zhang, Y. Product modularity: definitions and benefits. *Journal of Engineering Design*. Vol 14. No3. pp. 295-313. Sep 2003.
37. Gershenson, J. K., Prasad, G. J., & Allamneni S. Modular Product Design: a life-cycle view. *Transactions of the SDPS*. Vol 3. No 4. pp. 13-26. Dec 1999.
38. Gonzalez-Zugasti, J. P. & Otto, K. N. Platform-based spacecraft design: A formulation and implementation procedure. *IEEE Aerospace Conference Proceedings*. Vol 1. pp. 455-463. 2000.
39. Gonzalez -Zugasti, J. P., Otto, K. N., & Baker, J. D. Assessing value in platformed product value design. *Research in engineering design*. 13. pp. 30-41. 2001.
40. Gonzalez -Zugasti, J. P., Otto, K. N., & Baker, J. D. A method for architecting product platforms. *Research in engineering design.* 12. pp. 61-72. 2000.
41. Greer, J. L., Wood, K. L., & Jensen, D. D. Effort flow analysis: a methodology for directed product evolution. *Design Studies*. 25. pp. 193-214. 2004.
42. Gupta, S. & Krishnan, V. Integrated Component and Supplier Selection for a Product Family. *Production and Operations Management*, Vol. 8. No. 2. pp. 163-181. 1999.
43. Guo, F. & Gershenson, J. K. Comparison of modular measurement methods based on consistency and sensitivity analysis. *In Proc of ASME Design Engineering Technical Conferences*. Chicago, IL. 2003.
44. Guo, F. and Gershenson, J. K. A comparison of modular product design methods on improvement and iteration. *In Proc of ASME Design Engineering Technical Conferences*. Salt Lake City, UT. 2004.
45. Hauser, J.R. & Clausing, D. The House of Quality. *Harvard Business Review*. pp. 63-73. 1988.
46. Henderson, R. M. & Clark, K. B. Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly*. 35. pp. 9-30. 1990.
47. Hernandez, G., Allen, J., Woodruff, G., Simpson, T., Bascaran, E., Avila, L., & Salinas, F. Robust design of families of products with production modelling and evaluation. *ASME Journal of Mechanical Design*. Vol. 123. pp. 183-190. June 2001.
48. Hirtz, J., Stone, R., & McAdams, D. A functional basis for engineering design: Reconciling and evolving previous efforts. *Research in Engineering Design*. 12. pp. 65-82. 2002.
49. Hommes, Q. D. & Berry, P.W. Managing systems interface requirements – reconciliation using design structure matrix method. *INCOSE 2003*. 13th Annual international symposium proceedings.
50. Hölttä K. Comparative analysis of product modularization methods. *NordDesign*. Tampere, Finland. 2004.
51. Holtta, K. & Otto, K. Analyzing Module Commonality for Platform Design Using Dendrograms. (submitted)
52. Hölttä, K. & Magee C. Estimating Factors Effecting Project Task Size in Product Development – An Empirical Study. (submitted)

53. Hubka, V. & Eder, E. W. *Theory of technical systems*. 2nd ed. Springer-Verlag. 1988. ISBN 3-540-17451-6.
54. Hubka, V. & Eder, W. E. *Design Science*. Springer. 1996. ISBN 3-540-19997-7.
55. Hsuan Mikkola, J. Modularization assessment of product architecture. *DRUID winter conference 2000*. Denmark. 2000.
56. Hsuan Mikkola, J. Modularity, outsourcing, and inter-firm learning. *DRUID Summer conference 2000*. Rebild. Denmark. 2000.
57. http://www.idef.com/ (viewed 11/15/2004)
58. Jiao, J. & Tseng, M. M. Understanding product family for mass customization by developing commonality indices. *Journal of Engineering Design*. Vol 11. No 3. pp. 225-243. 2000.
59. Johnson, K., Langdon, P., & Ashby, M. Grouping materials and processes for the designer: an application of cluster analysis. *Materials Design.* 23. pp. 1-10. 2002.
60. Kaplan, C., Clark, R., & Tang, V. *Secrets of software quality*. McGraw-Hill, Inc. 1995. ISBN 0-07-911795-3.
61. Kim, K. & Chhajed, D. Commonality in product design: Cost saving, valuation change and cannibalization. *European Journal of Operations Research.* 125. pp. 602-621. 2002.
62. Koopman, P. J. A Taxonomy of decomposition strategies based on structures, behaviours, and goals. *In Proc of ASME Design Engineering Technical Conferences*. Boston. 1995.
63. Kota, S., Sethuraman, K., & Miller, R. A Metric for Evaluating Design Commonality in Product Families. *Journal of Mechanical Design*. Vol. 122. pp. 403 – 410. 2000.
64. Krishnan, V. & Gupta, S. Appropriateness and Impact of Platform-Based Product Development. *Management Science*. Vol. 47. No. 1. pp. 52-68. Jan 2001.
65. Kristjansson, A. H. & Hildre H-P. PAMatrix: A Method to Assess Platforms in Product Developing Companies. *NordDesign*. Tampere, Finland. 2004.
66. Kurfman, M., Stone, R., Van Wie, M., Wood, K., Otto, K. Theoretical Underpinnings of Functional Modeling: Preliminary Experimental Studies. *In Proc of ASME Design Engineering Technical Conferences*. Baltimore, MD. 2000.
67. Kurfman, M., Stock, M. E., Stone, R., Rajan, J., & Wood, K., 2003, Experimental studies assessing the repeatability of a functional modelling derivation method. *Journal of Mechanical Design*. Vol 125. Dec 2003.
68. Kusiak, A. Integrated product and process design: a modularity perspective. *Journal of Engineering Design*. Vol 13. No 3. pp. 223-231. 2002.
69. Maier, M. W. & Rechtin, E. *The art of systems architecting*. 2nd ed. CRC Press 2000. ISBN 0-8493-0440-7.
70. Martin, M. V. & Ishii, K. Design for variety: developing standardized and modularized product platform architectures. *Research in Engineering Design*. Vol 13. No 4. pp 213 - 235. 2002.
71. Mattson, C. A. & Magleby, S. P. The influence of product modularity during concept selection of consumer products. *In Proc of ASME Design Engineering Technical Conferences*. Pittsburgh, PA. 2001.
72. McAdams D. A., Stone, R. B., & Wood, Kristin L. Functional interdependence and product similarity based on customer needs. *Research in Engineering Design*. Vol 11. Issue 1. pp. 1-19. 1999.

73. McAdams, D. A & Wood, K. L. A Quantitative Similarity Metric for Design-by-Analogy. *ASME Journal of Mechanical Design*. Vol 124. pp 173-182. June 2002.
74. McGrath, M. E., Anthony, Michael T., & Shapiro, Amram R. *Product development: success through product and cycle time excellence*. Butteworth-Heinemann. 1992. ISBN 0-7506-9289-8.
75. McGrath M. E. *Product Strategy for High-Technology Companies*. New York: Irwin Professional Publishing. 1995.
76. Meltzer, R. J. Accelerating new product development. In: The PDMA handbook of new product development. Rosenau M., Griffin A., Castellion G., & Anschuetz N. (eds). John Wiley & sons.1996.
77. Messac, A., Martinez, M.P., & Simpson, T. W. Effective product family design using physical programming and the product platform concept exploration methods. *In Proc of ASME Design Engineering Technical Conferences*. pp. 689-699. Baltimore, MD. 2000.
78. Meyer, M. H. & Lehnerd, A. P. *The power of product platforms*. The Free Press. 2000. New York, NY. ISBN 0-648-82580-5.
79. Miller, T. D. Modular engineering. Doctoral Thesis. Department of Mechanical Engineering, Section for Engineering and Product Development, Technical Universtiy of Denmark. 2001.
80. Moore, W. L., Louvier, J. J., & Verma, R. Using conjoint analysis to help design product platforms. *Journal of product innovation management*.16. pp. 27-39. 1999.
81. Muffato, M. Introducing a platform strategy in product development. *International Journal of Production Economics*. 60-61. pp. 145-153. 1999.
82. Nayak, R. U., Chen, W., & Simpson, T. W. A variation-based methodology for product family design. *In Proc of ASME Design Engineering Technical Conferences*. pp. 701-710. Baltimore, MD. 2000.
83. Nelson, S. A. II, Parkinson, M. B., & Papalambros, P. Y. Multicriteria optimization in product platform design. *ASME Journal of Mechanical Design*. Vol 123. pp. 199-204. June 2002.
84. Newcomb, P. J., Bras, B., & Rosen, D. W. Implications of modularity on product design for the life cycle. *ASME Journal of Mechanical Design*. Vol 120. pp 483-490. Sep 1998.
85. O'Grady Peter. *The age of modularity*. Adams and Steele. 1999. ISBN 0-9670289-0-6.
86. Otto, K. A process for modularizing product families. *International Conference on Engineering Design.* Glasgow, Scotland. 2001.
87. Otto, K. & Wood K. *Product Design: techniques in reverse engineering and new product development*. Prentice Hall. Upper Saddle River, NJ. 2001. ISBN 0-13-021271-7.
88. Otto K. & Hölttä K. A multi-criteria framework for screening preliminary product platform concepts. *In Proc of ASME Design Engineering Technical Conferences*. Salt Lake City, UT. 2004.
89. Pahl G. & Beitz W. *Engineering Design*. 2nd ed. Springer-Verlag, London Ltd. 1999. ISBN 3-540-19917-9.
90. Pahl G. Fundamentals of Engineering Design. In: Handbook of mechanical engineering. Beitz W. & Kuettner K.-H. (eds). Chapter E. 1994.

91. Palani Rajan, P. K., Van Wie, M., Campbell, M., Otto, K. & Wood, K. Design for flexibility – measures and guidelines. *In Proc of International Conference on Engineering Design*. Stockholm. 2003.
92. Pedersen, K. *Designing platform families: an evolutionary approach to developing engineering systems*. Doctoral Thesis. GWW School of Mechanical Engineering, Georgia Institute of Technology. 1999.
93. Perera H. C. S. Nagarur N. & Tabucanon M. T. Component part standardization: A way to reduce the life-cycle costs of products. *International journal of production economics*. 60-61. pp. 109-116. 1999.
94. Pimmler, T. U. & Eppinger, S. D. Integration Analysis of Product Decompositions. *In Proc of ASME Design Engineering Technical Conferences*. Minneapolis, MN. 1994.
95. Rendell, J. VW top, but others are catching up fast. *Automotive world*. pp. 26-34. Sep 2001.
96. Riitahuhta, A. & Andreasen, M. M. Modularisation support of life cycle management. *in Proc of the 1st international symposium on environmentally conscious design and inverse manufacturing*. Tokyo, Japan. 1999.
97. Roemer, T. and Fixson, S. Parts commonality and communication delays in product development. *Euroma 2004*, Fontainebleau, France. June 2004.
98. Salhieh, S. M. & Kamrani, A. K. Macro level product development using design for modularity. *Robotics and Computer integrated manufacturing*. 15. pp. 319-329. 1999.
99. Sanderson, S. & Uzumeri, M. Managing product families: the case of Sony Walkman. *Research Policy*. 24. pp. 761-782. 1995.
100. Siddique, Z. & Rosen D. W. Product family configuration reasoning using discrete design spaces. *In Proc of ASME Design Engineering Technical Conferences*. Baltimore, MD. 2000.
101. Simon, H. A. The architecture of complexity. Proceedings of the American Philosophical Society. 106. pp. 467-482, Dec 1962.
102. Simpson, T., Maier, J., & Mistree, F. Product platform design: method and application. *Research in Engineering Design*. vol. 13. No. 1. pp. 2-22. 2001.
103. Simpson, T.W. Product platform design and customization: Status and promise. *AIESAM*, Special issue: Platform product development for mass customization. Vol 10. No. 1. Jan 2004.
104. Smith, J. & Duffy, A. Modularity in support of design for re-use. *International Conference on Engineering Design.* Glasgow, Scotland. 2001.
105. Sosa, M. E., Eppinger, S. D., & Rowles, G. M. *Understanding the Effects of Product Architecture on Technical Communication in Product Development Organizations*. MIT Sloan School of Management. Working paper # 4130. 2000.
106. Sosa, M. E, Eppinger, S. D, & Rowles C. M. Designing modular and integrative systems. *In Proc of ASME Design Engineering Technical Conferences*. Baltimore, MD. September 10-13, 2000.
107. Stake, R. B. *On conceptual development of modular products*, Doctoral Thesis. Division of Assembly Systems, Dept. of Production Engineering, Royal Institute of Technology. Stockholm. 2000.
108. Steuer, C. & Whitcomb C. Economical assessment of product platform concepts under uncertainty – capturing the values of flexibility with real options. *INCOSE* 13th Annual international symposium proceedings. 2003.

109. Stone, R. B., Wood, K. L. & Crawford, R. H. A heuristic method for identifying modules for product architecture. *Design Studies*. Vol 21. No 1. pp. 5-31. 2000.
110. Sudjianto, A. & Otto, K. Modularization to support multiple brand platforms. *In Proc of ASME Design Engineering Technical Conferences*. Pittsburgh, PA.September 9-12, 2001.
111. Suh, N. *Axiomatic Design: Advances and Applications*, Oxford University Press, New York, NY. 2001.
112. Sääksjärvi, M. Software application platforms: from product architecture to integrated application strategy. *In Proc of the 26th annual international computer software and application conference*. IEEE Computer Society. 2002.
113. Tatikonda, M. V. An empirical study of platform and derivative product development projects. *Journal of Product innovation management.*16. pp. 3-26. 1999.
114. Thebeau, R. E. *Knowledge Management of System Interfaces and Interactions for Product Development Processes*. Masters Thesis. System Design & Management Program, Massachusetts Institute of Technology. 2001.
115. Thevenot, H. J. & Simpson, T. W. A comparison of commonality indices for product family design. *In Proc of ASME Design Engineering Technical Conferences*. Salt Lake City, UT. September 28- October 2, 2004.
116. Thomke, S. H. The role of flexibility in the development of new products: an empirical study. *Research Policy*. 26. pp. 105-119. 1997.
117. Ulrich, K. T. & Eppinger, S. D. *Product Design and Development*. McGraw-Hill. 3rd edition. 2004. ISBN 0-07-247146-8.
118. Ulrich K. & Tung K. Fundamentals of Product Modularity. *In proc of ASME Winter Annual Meeting Symposium on Design and Manufacturing Integration*. pp. 73-79. Atlanta, GA. November 1991.
119. Ulrich, K. The role of product architecture in the manufacturing firm. *Research policy*. Vol 24. pp. 419-440. 1995.
120. Webster (www.m-w.com)
121. Wheelwright S. C. & Clark, K. B. Creating project plans to focus product development. *Harvard Business review*. 1992.
122. Whitney, D. E. Designing the Design Process. *Research in Engineering Design*. Vol 2. No 3. pp. 3-13. 1990.
123. Whitney, D. E., *Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development*. Oxford University Press. 2004.
124. Whitney D. E. *Physical limits to modularity*. Massachusetts Institute of Technology, Engineering Systems Division. Working paper. ESD-WP-2003-01.03-ESD.
125. Yin, R. K. *Case Study Research*. 3rd ed. Sage publications. 2003. In the Applied Social Research Methods Series Volume 5. ISBN 0-7619-2553-8.
126. Zamirowski, E. J. & Otto K. N. Identifying Product Family Architecture Modularity Using Function and Variety Heuristics. *In Proc of ASME Design Engineering Technical Conferences*. Las Vegas, NV. 1999.

ISBN 951-22-7766-2
ISBN 951-22-7767-0 (PDF)
ISSN 1795-2239
ISSN 1795-4584 (PDF)