



KTH Machine Design

Approaches to Modularity in Product Architecture

FREDRIK BÖRJESSON

Licentiate thesis
Department of Machine Design
Royal Institute of Technology
SE-100 44 Stockholm

TRITA – MMK 2012:11
ISSN 1400-1179
ISRN/KTH/MMK/R-12/11-SE
ISBN 978-91-7501-390-9


TRITA – MMK 2012:11
ISSN 1400-1179
ISRN/KTH/MMK/R-12/11-SE
ISBN 978-91-7501-390-9

Approaches to Modularity in Product Architecture

Fredrik Börjesson

Licentiate thesis

Academic thesis, which with the approval of Kungliga Tekniska Högskolan, will be presented for public review in fulfilment of the requirements for a Licentiate of Engineering in Machine Design. The public review is held at Kungliga Tekniska Högskolan, Brinellvägen 83, room B242 on June 11, 2012 at 10:00.

Department of Machine Design Royal Institute of Technology S-100 44 Stockholm SWEDEN			TRITA - MMK 2012:11 ISSN 1400 -1179 ISRN/KTH/MMK/R-12/11-SE ISBN 978-91-7501-390-9	
<i>Author</i> Fredrik Börjesson (fredrik.borjesson@modularmanagement.com)			<i>Document type</i> Thesis	<i>Date</i> 2012-06-11
			<i>Supervisor(s)</i> Ulf Olofsson, Ulf Sellgren	
<i>Title</i> Approaches to Modularity in Product Architecture			<i>Sponsor(s)</i>	
<i>Abstract</i> <p>Modular product architecture is characterized by the existence of standardized interfaces between the physical building blocks. A module is a collection of technical solutions that perform a function, with interfaces selected for company-specific strategic reasons. Approaches to modularity are the structured methods by which modular product architectures are derived. The approaches include Modular Function Deployment (MFD), Design Structure Matrix (DSM), Function Structure Heuristics and many other, including hybrids. The thesis includes a survey of relevant theory and a discussion of four challenges in product architecture research, detailed in the appended papers.</p> <p>One common experience from project work is structured methods such as DSM or MFD often do not yield fully conclusive results. This is usually because the algorithms used to generate modules do not have enough relevant data. Thus, we ask whether it is possible to introduce new data to make the output more conclusive. A case study is used to answer this question. The analysis indicates that with additional properties to capture product geometry, and flow of matter, energy, or information, the output is more conclusive.</p> <p>If product development projects even have an architecture definition phase, very little time is spent actually selecting the most suitable tool. Several academic models are available, but they use incompatible criteria, and do not capture experience-based or subjective criteria we may wish to include. The research question is whether we can define selection criteria objectively using academic models and experience-based criteria. The author gathers criteria from three academic models, adds experience criteria, performs a pairwise comparison of all available criteria and applies a hierarchical cluster analysis, with subsequent interpretation. The resulting evaluation model is tested on five approaches to modularity. Several conclusions are discussed. One is that of the five approaches studied, MFD and DSM have the most complementary sets of strengths and weaknesses, and that hybrids between these two fundamental approaches would be particularly interesting.</p> <p>The majority of all product development tries to improve existing products. A common criticism against all structured approaches to modularity is they work best for existing products. Is this perhaps a misconception? We ask whether MFD and DSM can be used on novel product types at an early phase of product development. MFD and DSM are applied to the hybrid drive train of a Forwarder. The output of the selected approaches is compared and reconciled, indicating that conclusions about a suitable modular architecture can be derived, even when many technical solutions are unknown. Among several conclusions, one is the electronic inverter must support several operating modes that depend on high-level properties of the drive train itself (such as whether regeneration is used). A modular structure for the electronic inverter is proposed.</p> <p>Module generation in MFD is usually done with Hierarchical Cluster Analysis (HCA), where the results are presented in the form of a Dendrogram. Statistical software can generate a Dendrogram in a matter of seconds. For DSM, the situation is different. Most available algorithms require a fair amount of processing time. One popular algorithm, the Idicula-Gutierrez-Thebeau Algorithm (IGTA), requires a total time of a few hours for a problem of medium complexity (about 60 components). The research question is whether IGTA can be improved to execute faster, while maintaining or improving quality of output. Two algorithmic changes together reduce execution time required by a factor of seven to eight in the trials, and improve quality of output by about 15 percent.</p>				
<i>Keywords</i> Clustering Algorithm, Design Structure Matrix, Modular Function Deployment, Product architecture, Product family, Product platform			<i>Language</i> English	

Acknowledgements

Thanks to Doctor Ulf Sellgren and Professor Ulf Olofsson, both of KTH Royal Institute of Technology in Stockholm, Sweden, for allowing me to pursue this licentiate degree while living and working in the US.

I have learned a tremendous amount from my faithful co-authors, Doctor Sellgren of KTH and Assistant Professor Katja Hölttä-Otto of University of Massachusetts Dartmouth. You have helped me improve my academic writing style. Hopefully I am better now than when I started.

Thanks also to my colleague Dr. Gunnar Erixon for patiently offering feedback on several revisions of most of my papers.

By helping me out every day, my wife Teresa has made it possible for me to pursue a licentiate degree while keeping a full time job.

Stockholm, May 2012

Fredrik Börjesson

List of appended publications

This thesis consists of a summary and the following appended papers:

Paper A

Borjesson, F., Improved Output in Modular Function Deployment Using Heuristics, Proceedings of the 17th International Conference on Engineering Design (ICED'09), Vol. 4, ISBN 9-781904-670087, pp. 1-12.

The author wrote the paper in its entirety.

Paper B

Borjesson, F., A Systematic Qualitative Comparison of Five Approaches to Modularity, International Design Conference – Design 2010, Dubrovnik – Croatia, May 17-20, 2010.

The author wrote the paper in its entirety.

Paper C

Borjesson, F., Sellgren, U., Modularization of novel machines: motives, means and opportunities, Proceedings of NordDesign 2010, Chalmers University of Technology, Gothenburg, Sweden, August 25-27, 2010.

The author performed most of the writing, analysis, and preparation of graphics. Doctor Sellgren wrote the section on research context, and provided ideas on the structure of the paper.

Paper D

Borjesson, F., Hölttä-Otto, K., Improved Clustering Algorithm for Design Structure Matrix (accepted for publication), Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2012, DETC2012-70076. Chicago, IL, USA, August 12-15, 2012.

The author developed the algorithm, coded it in Matlab, ran the trials, documented the result, developed the graphics, and wrote most of the paper. Assistant Professor Hölttä-Otto provided feedback on the structure of the paper, proposed several references, and suggested improvements to the text for improved understanding.

ABBREVIATIONS

Abbreviation	Term
CR	Customer Requirement
DP	Design Parameter
DPM	Design Property Matrix
DSM	Design Structure Matrix
FR	Functional Requirement
GA	Genetic Algorithm
IGTA	Idicula-Gutierrez-Thebeau Algorithm
ITC	Improved Termination Criterion
MD	Module Driver
MIM	Module Indication Matrix
PMM	Product Management Map
PP	Product Property
QFD	Quality Function Deployment
SMA	Suppressing Multicluster Allocation
TS	Technical Solution

NOMENCLATURE

Term	Definition
Approach to Modularity	A structured approach where data is collected, analyzed, and transformed to predict the best Modular Product Architecture
Cluster (<i>noun</i>)	Collection of one or more Elements
cluster (<i>verb</i>)	generate a set of Clusters by means of an algorithm
ClusterBid	Degree of fit between a selected Element and each of the existing Clusters; calculation includes a punishment for ClusterSize
ClusterSize	Number of Elements in Cluster
Component	Simple physical entity which has Interaction with other simple physical entities
Component-DSM	Matrix of Interactions between pairs of Components
Core	Part of IGTA/IGTA-plus/R-IGTA responsible for moving randomly selected Elements from one Cluster to another, and keeping track of best solution so far
Customer Requirement	statement of the usage experience the customer desires in their use of the product
Dendrogram	Hierarchical representation of the degree of Product Property or Module Driver similarity between Technical Solutions
Design Parameter	term used by Nam Suh, corresponds to Product Property or Technical Solution

Design Property Matrix	matrix used to describe the relative impact of design changes to Technical Solutions on the performance of the product, as captured by Product Properties
Design Structure Matrix	matrix representation of a system or project in which all constituent components or activities are listed together with their corresponding dependency pattern
Element	Component or Technical Solution
Extra-cluster interaction	Interactions between Elements that belong to different Clusters
Function	Transformation of energy, information, or material
Functional Requirement	term used by Nam Suh, corresponds to Customer Requirement or Product Property
Function-structure diagram	Flowchart showing functions and the exchange of energy, information, or material between them
Function-structure heuristics	Three rules of thumb (Stone, Wood & Crawford 2000) applied to a function-structure diagram to yield Modules
Genetic Algorithm	search heuristic that mimics the process of natural evolution, by generating solutions using mechanisms such as inheritance, mutation, selection, and crossover
Heuristics	Rule of thumb that usually yields good results
Hierarchical Clustering Algorithm	algorithm that operates on a matrix to generate a hierarchy of clusters with similar elements, the output of which is usually presented as a dendrogram (tree-graph)
Idicula-Gutierrez-Thebeau Algorithm	algorithm for clustering Component-DSM
IGTA-plus	Modification of IGTA that includes two algorithmic changes, SMA and ITC
Improved Termination Criterion	selecting candidate Elements from a list, and subsequently deleting the Element from that list
Interaction	Exchange of energy, information, material or an association of physical space and alignment
Interface	Surface or volume between two or more Clusters, through which Interaction may take place; if no Interaction takes place, there is no Interface
Interface Matrix	Matrix of Interactions between pairs of Modules
Intra-cluster interaction	Interactions between Elements that belong to the same Cluster
Modular Function Deployment	modularity method that involves populating and analyzing three interlinked matrices used to describe the relation between Customer Requirements, Product Properties, Technical Solutions, and Module Drivers

Modular Function Deployment (MFD)	Modularity method that involves populating and analyzing three interlinked matrices used to describe the relation between Customer Requirements, Product Properties, Technical Solutions, and Module Drivers
Modular Product Architecture	Representation of a product or family of products as a collection of Modules, which allows for efficient development, production, and marketing
Module	Cluster that forms a functional building block with specified interfaces, selected for company-specific reasons
Module Driver	one of 12 pre-defined strategic reasons for creating interfaces, used for describing the business intent of the product structure
Module Indication Matrix	matrix used to describe the strategic intent of individual Technical Solutions, using Module Drivers
Multicluster allocation	Feature of IGTA where an element may be assigned to more than one cluster if the Multicluster condition is true
Multicluster condition	More than one Cluster returns the highest ClusterBid in IGTA
Product Management Map	visualization of the interlinked matrices QFD, DPM, and MIM used in MFD
Product Property	Precise quantifiable statement of what the product has to do
Quality Function Deployment	matrix used to describe the relation between Customer Requirements and Product Properties
R-IGTA	Modification of IGTA-plus to cluster simultaneously with regard to Component-DSM and DPM/MIM, using ratio of TotalCost and Reangularity as an optimization criterion
Reangularity	A metric between zero and one that measures the degree to which a design is uncoupled, extended here to cover Modules
Suppressing Multicluster Allocation	allowing an Element to be assigned to one and only one Cluster
Technical Solution	Physical entity designed to embody Product Properties and carry a required function in the product
Thebeau's algorithm	<i>(same as) IGTA</i>
TotalCost	Sum of all Intra and Extra-cluster interactions, with an additional punishment for the latter

TABLE OF CONTENTS

ABBREVIATIONS	3
NOMENCLATURE	3
1 INTRODUCTION	9
1.1 Product architecture.....	9
1.2 Modularity	10
1.3 Approaches to modularity	10
1.4 Modularity versus Standardization.....	10
1.5 All approaches have to model reality	11
1.6 The author's interest in modularity	12
1.7 Research questions	13
1.8 Delimitations	13
1.9 Interrelation of topics	13
1.10 Relation to other concepts and researchers' work.....	14
1.11 Papers in the context of product development process	14
1.12 Thesis outline	14
2 FRAME OF REFERENCE.....	15
2.1 Introduction	15
2.2 Fundamental concepts	15
2.2.1 Theory of Technical Systems.....	15
2.2.2 Quality Function Deployment (QFD).....	17
2.2.3 Hierarchical Clustering	19
2.2.4 Design Structure Matrix.....	20
2.2.5 Function structure heuristics	21
2.2.6 Clustering algorithms for DSM	22
2.2.7 Modular Function Deployment (MFD)	23
2.3 Concepts summary	25
3 RESEARCH METHODOLOGY.....	26
3.1 Scientific method.....	26
3.1.1 Question	27
3.1.2 Literature survey	28
3.1.3 Analysis / object of study.....	28
3.1.4 Problem / question	28

3.1.5	Observations	28
3.1.6	Hypothesis.....	29
3.1.7	Analysis / interpretation.....	29
3.1.8	Report.....	29
4	SUMMARY OF RESULTS AND APPENDED PAPERS	30
4.1	Introduction	30
4.2	Paper A – Improved output	30
4.2.1	Background	30
4.2.2	Findings.....	31
4.3	Paper B – Qualitative comparison.....	31
4.3.1	Background	31
4.3.2	Findings.....	32
4.4	Paper C – Modularization of novel machines	33
4.4.1	Background	33
4.4.2	Findings.....	33
4.5	Paper D – DSM Clustering.....	33
4.5.1	Background	33
4.5.2	Findings.....	34
5	DISCUSSION.....	35
5.1	Introduction	35
5.2	Challenges in architecture research.....	35
5.3	Paper A – Convergence properties.....	35
5.4	Paper B – Qualitative comparison.....	37
5.5	Paper C – Hybrid drive.....	38
5.6	Paper D – Improved clustering algorithm	39
5.7	Impact of research on consulting	40
6	CONCLUSION AND FUTURE RESEARCH.....	41
6.1	Introduction	41
6.2	Conclusions	41
6.2	Future research	42
6.2.1	Observe development of novel product types.....	42
6.2.2	Survey-based evaluation	43

6.2.3 Automatic clustering of FS-DSM	43
6.2.4 New Convergence Properties.....	43
6.2.5 Modularization of the electronic inverter	44
6.2.6 Further computational improvements to IGTA-plus	44
6.2.7 Heuristics to improve IGTA	44
6.2.8 GA-core for IGTA	44
6.2.9 Expand the problem domain for IGTA.....	44
7. REFERENCES	46

APPENDED PAPERS A-D

1 INTRODUCTION

This chapter presents the background information to modularization and clustering, the terminology used, the objective and research questions, as well as briefly describes the used research methodology and outlines the structure of this thesis.

1.1 Product architecture

This thesis deals with product architecture. Architecture is a familiar term, and we typically think of buildings or floor plans when we hear it. The term “product architecture” is much less known among a general audience. Wikipedia does not offer a definition, as shown in Figure 1 (Wikipedia 2012).

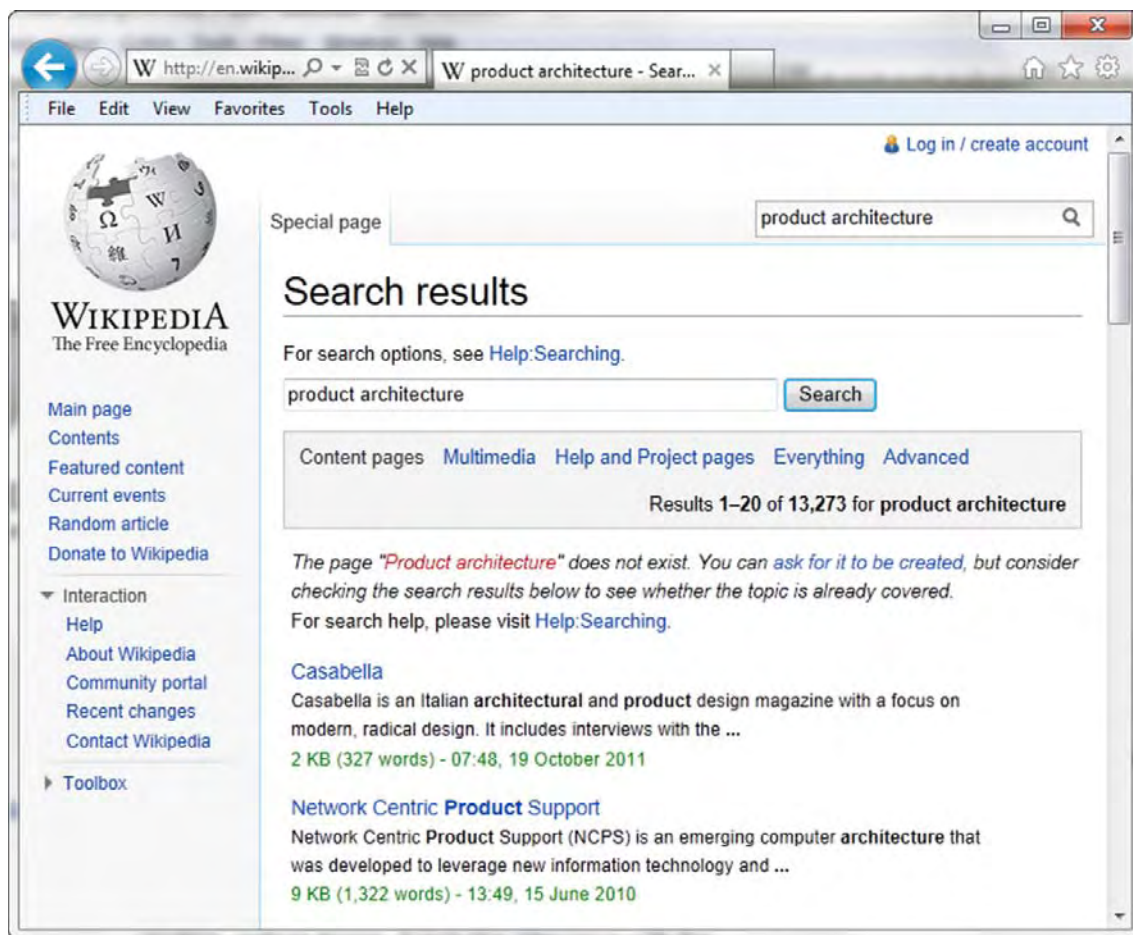


Figure 1. Product Architecture is not defined by Wikipedia

Among design engineers, the term is well known, but still defined differently. The following definition of Product Architecture is taken from (Wyatt, Wynn, Jarrett & Clarkson 2012):

“Product architectures are the abstract conceptual structures underlying the functioning of engineering artefacts, and their design is an important but difficult task (Ulrich 1995).” The original definition (Ulrich 1995) reads “Product architecture is the scheme by which the function of a product is allocated to physical components.” The terms “product family” and “product platform” are preferred by some researchers, as in the following segment from (Simpson et al. 2011): “A product family is a group of related products that are derived from a common set of components, modules, and/or subsystems to satisfy a variety of market applications where the common ‘elements’ constitute the product platform (Meyer & Lehnerd 1997)”. The desire to achieve commonality is one of the reasons for creating product platforms, but not the only one. Common unit (Erixon 1998) is one twelve Module Drivers used to define the strategic intent of a proposed Product Architecture. (Erixon 1998) states that “product architecture is mostly used in the US and is used here synonymous with product structure”. Product structure is defined (Erixon 1998) as “the elements of a product and their relations (Tichem & Storm 1995)”.

1.2 Modularity

The terms “module” and “modularity” are often used in the context of product architecture, and there is often some confusion with regard to the meaning of these terms. This is confirmed by (Yu, Yassine & Goldberg 2007), who state simply that the term modularity is an ambiguous and elusive notion that has been loosely used in different ways by different people at different times. This clearly is not a good situation. In the context of the present thesis, we will impose restrictions on the term “module”, to make it more well-defined and useful. Modules shall be defined as groups of technical solutions that carry out one or several functions, and which have a standardized interface to the world around it. This is consistent with (Erixon 1998). Modularity entails standardizing the interfaces, which implies one module may be interchanged for another, allowing for a different performance levels or styling, for example. Modular product architecture may be viewed as a subset of product architecture.

1.3 Approaches to modularity

Modular product architectures are generated through the application of a pre-defined method. An approach to modularity includes the method by which the architecture is derived – but it covers a bit more than just the method itself. In actual projects, the author has found that a cross-functional team is a very important success factor, as is solid management commitment. Very often, the work happens in a workshop format. The method itself is the way the data is captured and processed, which is a slightly more narrow concept.

1.4 Modularity versus Standardization

Product architecture is often approached with component standardization. Modularity and standardization are *not* the same thing. The graphic in Figure 2 highlights the main differences in these two views.

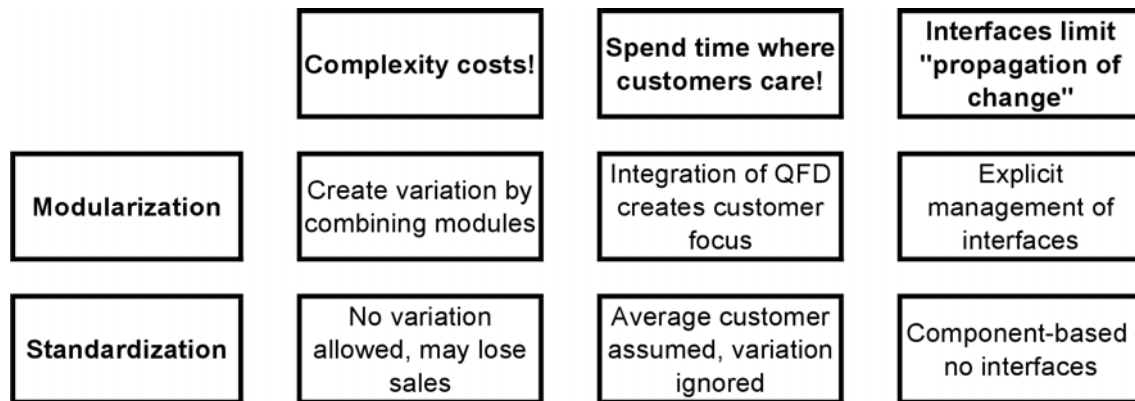


Figure 2. Modularization and Standardization are not the same thing

We might say that modularization embraces variation and deals with it through the active management of standardized interfaces. Standardization tries to find an average performance level, which ultimately may generate dissatisfied customers and reduced sales.

1.5 All approaches have to model reality

All approaches to modularity have to build a model of reality, that captures the aspects of the product that have implications for the architecture (where the interfaces are required, for example). Although some of the details differ between different approaches, there are similarities. The graphic in Figure 3 tries to show, on a very high level, what product architecture approaches have in common.

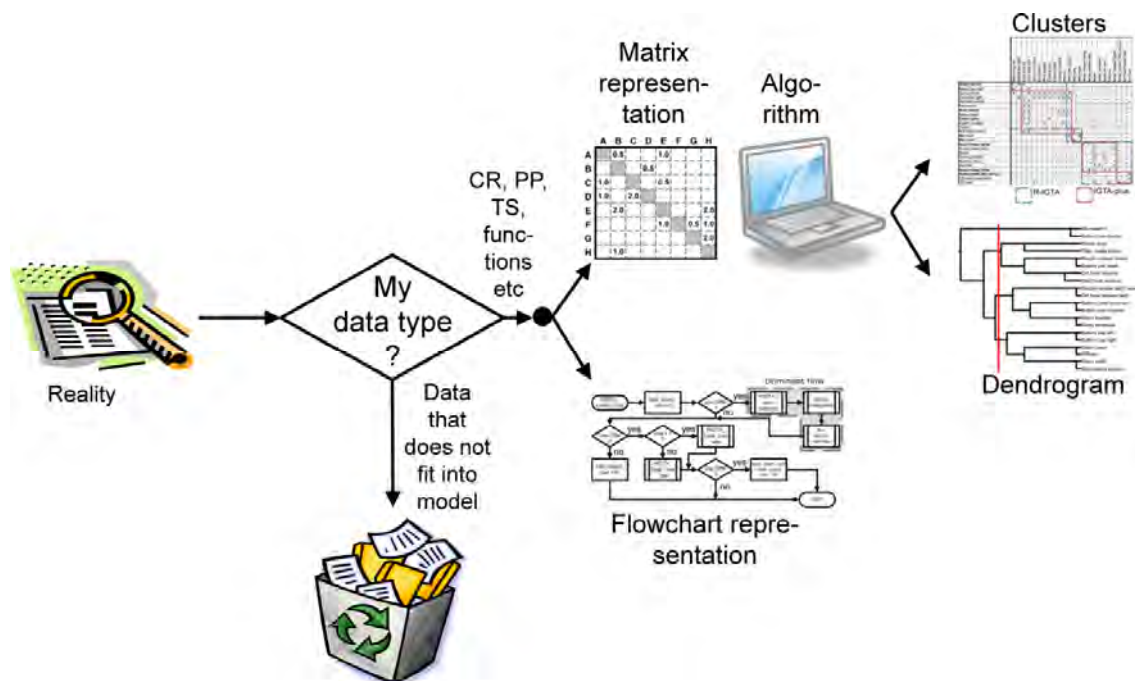


Figure 3. Architecture work

On the far left, we see an icon that tries to represent the view of reality: this may be a previous generation of products with characteristics similar to the new one, or a competitor's product. Whatever the source of data, team representatives from Engineering or Marketing functions in the company gather data about the product, its usage, the customers etc, and then sift through the data to determine what is important in the project. Some data will be

discarded at this point, because it is out of scope or does not fit into the representation of the product, be it a matrix, a drawing, a flowchart etc. What is left may be a list of Customer Requirements, Product Properties, desired Functions or Features, cost data for concept selections etc. Some choice is typically made about the way this data should be represented. One or several representations may be available, including matrices, flowcharts, product sketches etc. If the objective is to make predictions about the best possible modules, it usually becomes necessary to select some type of pre-defined representation. Figure 3 shows two such options. The upper is a matrix representation, which has computer algorithm support for generating modules. The lower is a function structure diagram (a type of flowchart). When diagrams are used, the work may be conducted on paper, and module generation may be manual, using a set of pre-defined heuristic rules for what constitutes good modules. Computer algorithms operating on matrix representations include such methods as Design Structure Matrix, DSM, and Module Function Deployment, MFD. Depending on the algorithm, the output may be a sorted matrix or a Dendrogram, as shown in the graphic. The computer-generated output is analyzed by the team members, and decisions are made about modules. Typically, there are many iterations of changing data and resorting before the output is satisfactory.

Once the output is deemed useful, it is documented in some form, and goes to detailed design, where three-dimensional representations using Computer Aided Design are often used.

Although this description of reality is a simplification, the purpose is to position the present thesis, and to define the domain of problems we are addressing. This will be detailed in the next section.

1.6 The author's interest in modularity

The author's interest in the topic is not only academic. Since December of 2002, the author has worked as a product architecture consultant (at Sweden-based consulting firm Modular Management) and has been involved in 15-20 client projects. Almost all of the research topics were inspired by real problems encountered in the consulting work. There are some obvious similarities between academic research and the "research" that happens in real projects through the application of new ideas that get conceived and tried out. One similarity is that project "research" and most academic research both try to improve existing methods. The main difference may be in the emphasis placed on scientific rigor. In project application, the primary objective is to generate a useful output, to solve the problem immediately at hand. Academic research builds on results by other researchers and aims to generate output that improves the methods by which products are conceived or designed.

One very important assumption has been that all the research topics in the present thesis have some practical application. There is a heavy slant toward the issue of practical usage of all the methods presented.

There have been three types of influence on the research topics in the present thesis.

The first and foremost is the experience gained in actual project work with real clients. Working with a client has many advantages and very few disadvantages. The advantages include a strong focus on output and access to detailed subject matter knowledge. A possible disadvantage may be the time pressure.

The second is product architecture training experience. The author was involved in a long engagement with a global client over the course of about five years. As a part of this

engagement, the author devised training material and conducted training with hundreds of engineers in Europe, USA, Mexico, and Brazil.

The third is contact with the academic world. The author has attended conferences and authored papers in conjunction with other researchers.

1.7 Research questions

The unifying theme of the four research papers included in the present thesis is whether we can improve the methods used for generating modular product architecture. Each of the papers addresses some facet of this overall theme.

- Can we supply additional information to improve the output of the methods (MFD, in particular)?
- Can we define selection criteria objectively, yet incorporate experience-based criteria?
- How well do the methods (MFD and DSM, in particular) work for new types of products at an early phase in the product development process?
- Can the current computerized algorithms for module generation be made to run faster and generate better results (DSM, in particular)?

1.8 Delimitations

The present thesis deals with structuring of products at an early phase of the development. This is applicable both to existing products and novel product types.

Only physical products are within scope. Although some aspects of modularity may be applicable to abstract products such as bundles of services, that is not covered here. The interfaces between modules are physical, and involve spatial relations or the transfer of energy, matter, or information through a physical contact surface or a defined volume, which implies we are not concerned with the structure of software.

We assume modularity is applied to products where the existence of standardized interfaces does not present a possible detriment to the performance, as may be the case in the design of anthropomorphic robots, for example, where the distribution of weight is extremely critical. It may be possible to argue that highly integrated products with tough requirements on reliability fall into that category too, as may be the case with pacemakers, for example.

Finally, the theories of modularity typically work best above a certain level of complexity. Product design for extremely simple products probably do not require modularity. This may be the case for a coffee filter, for example, where a solid understanding of filtration is more useful.

1.9 Interrelation of topics

The overall theme of the four papers appended in the present thesis is *improved methods*. We consider MFD, DSM, and Function-Structure Heuristics to be *fundamental methods*. Each fundamental method has a set of advantages and disadvantages, which is explored in Paper B. One path to improved methods is to combine two or more fundamental methods into a *hybrid method*. As discussed in Paper B, hybrid methods usually have a new set of disadvantages that are absent in the methods upon which they build. Most methods *represent data* in one of two forms, a matrix or a graphical format such as a function-structure diagram. Paper A explores Product Property types used in one particular matrix-based method, MFD, and proposes a scheme whereby features of Function-Structure Heuristics may be integrated. Paper C applies two matrix-based methods, MFD and DSM, to a novel product in an early

phase of product development, and makes a qualitative comparison of the outputs. Paper D, finally, focuses on an important clustering algorithm for DSM and presents improvements that increase the quality of output while making the computations significantly faster.

1.10 Relation to other concepts and researchers' work

Paper A integrates features of Function-Structure Heuristics (Stone, Wood & Crawford 2000) into MFD (Erixon 1998). Paper B builds on the works by (Keller & Binz 2009), (Huang 1996), and (Hölttä 2005), but instead of simply dictating a set of evaluation criteria, a method is shown whereby external criteria may be integrated with experience-based criteria. Paper C evaluates the usefulness of MFD and DSM when applied to a novel product in an early phase of development, when relatively little is known about the constituent Technical Solutions. The most common type of case application used in academic studies involves fairly well-known products. Paper D improves the work by (Thebeau 2001B) by proposing computational improvements that radically improve speed and quality.

1.11 Papers in the context of product development process

Figure 4 shows the five papers laid out in a product development process.

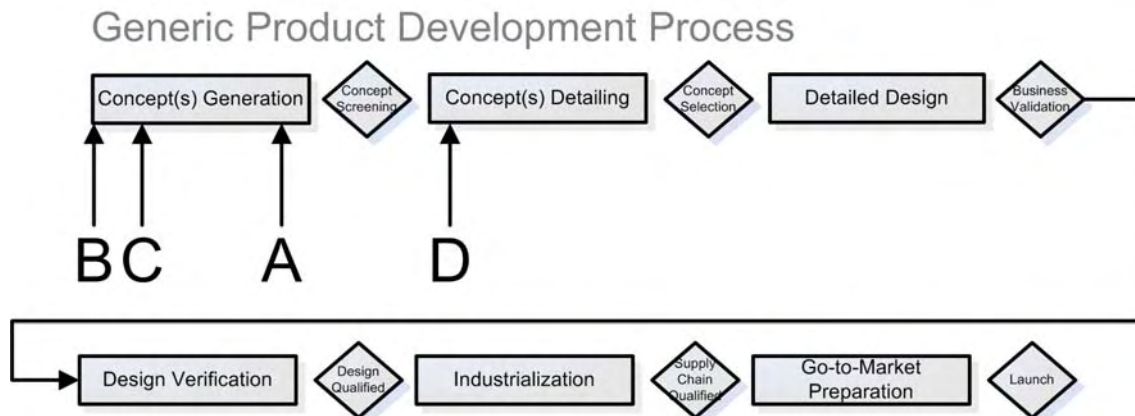


Figure 4. Papers A-E in the context of product development

Paper B involves selecting the right approach, e.g., MFD, DSM, Function-structure heuristics, or some hybrid approach. Papers A, C, and D all deal with the generation of a modular concept: Paper A looks at the role of properties, paper C examines the value of a qualitative comparison, and paper D proposes specific algorithmic improvements that apply to DSM. Once the concepts are generated, the team would attempt to determine the required performance levels for all the modules ("module variants" in MFD terminology).

1.12 Thesis outline

Chapter 1 is the justification of the research questions, as well as the context, both to the work of other researchers and the interrelation of the topics themselves. Chapter 2 goes into some definitions we use. Chapter 3 describes the methodology. Chapter 4 is a summary of the papers. Chapter 5 is a discussion. The author attempts to assess the value of the scientific contributions in each paper. Chapter 6, finally, outlines some possible future work.

2 FRAME OF REFERENCE

This chapter provides some fundamental theory of modular product architecture.

2.1 Introduction

In this chapter, we will look more deeply into the theory on which this work relies.

2.2 Fundamental concepts

2.2.1 Theory of Technical Systems

The following graphic from (Hubka & Eder 1996) shows how Product Properties – in a very broad sense – fall into larger categories that capture the Purpose of the Technical System, the Life phases, and finally the relation between the product and its environment, Humans and Society. Note especially that several properties in Life phases are used as Module Drivers in MFD.

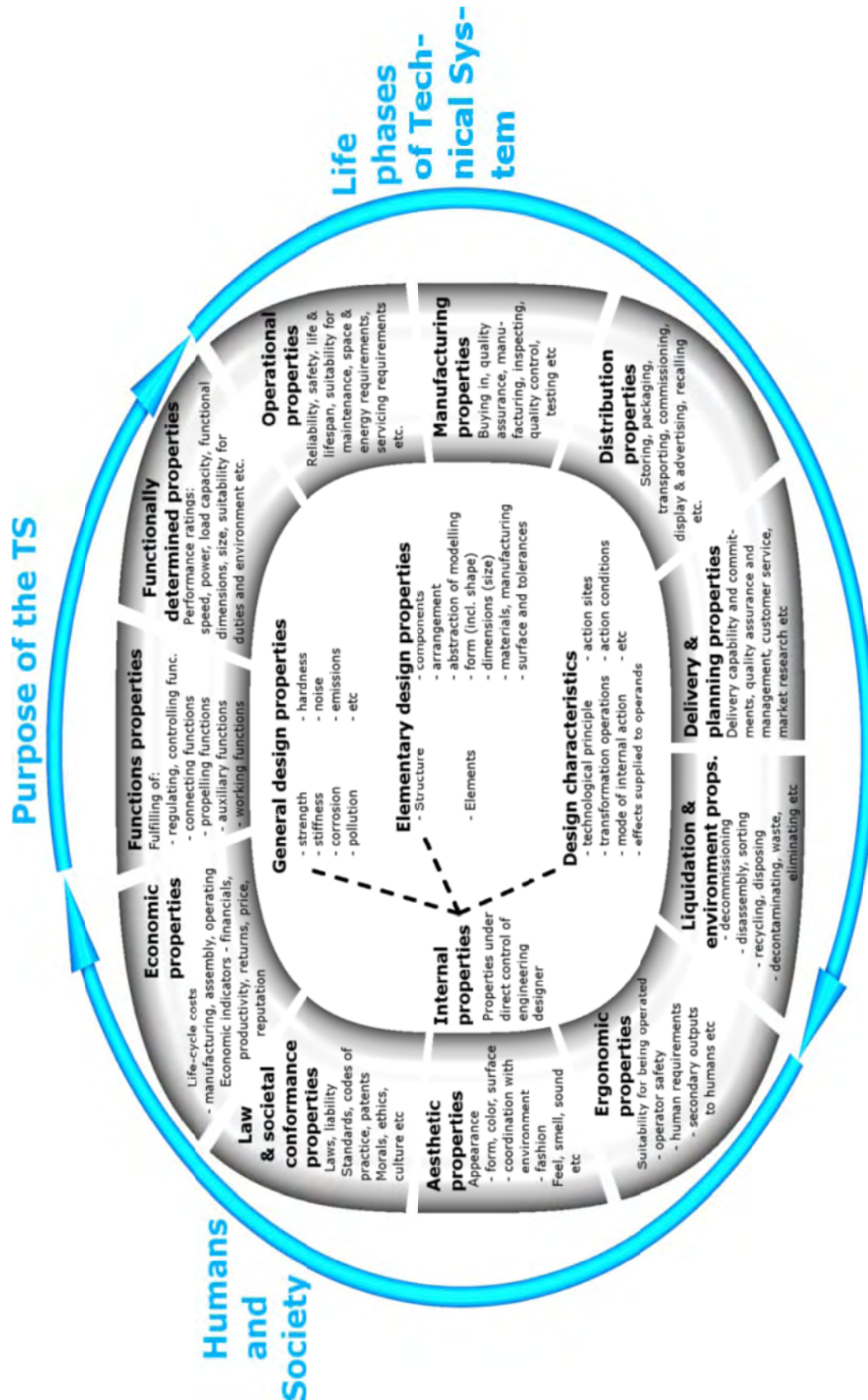


Figure 5. Property types according to (Hubka & Eder 1996), please rotate page to read

2.2.2 Quality Function Deployment (QFD)

According to (Hauser & Clausing 1988), QFD originated in 1972 at Mitsubishi's Kobe Shipyard and was perfected over time by Toyota and others. The QFD Institute (QFD Institute 2012) lists Dr. Yoji Akao as "one of the founders of QFD". One of Dr. Akao's publications is (Akao & Mizuno 1994).

The following graphic shows a comparison of a QFD as it normally appears in a full House of Quality (top), image from (Hauser & Clausing 1988), and in MFD (bottom). Note the QFD as used in MFD does not feature the mandatory "roof" used in House of Quality. Conflicting requirements are dealt with in MFD by Technical Solution decomposition, which happens when the DPM is populated. There is no guaranteed solution to built-in conflicts, though; MFD just offers another way of looking at these conflicts.

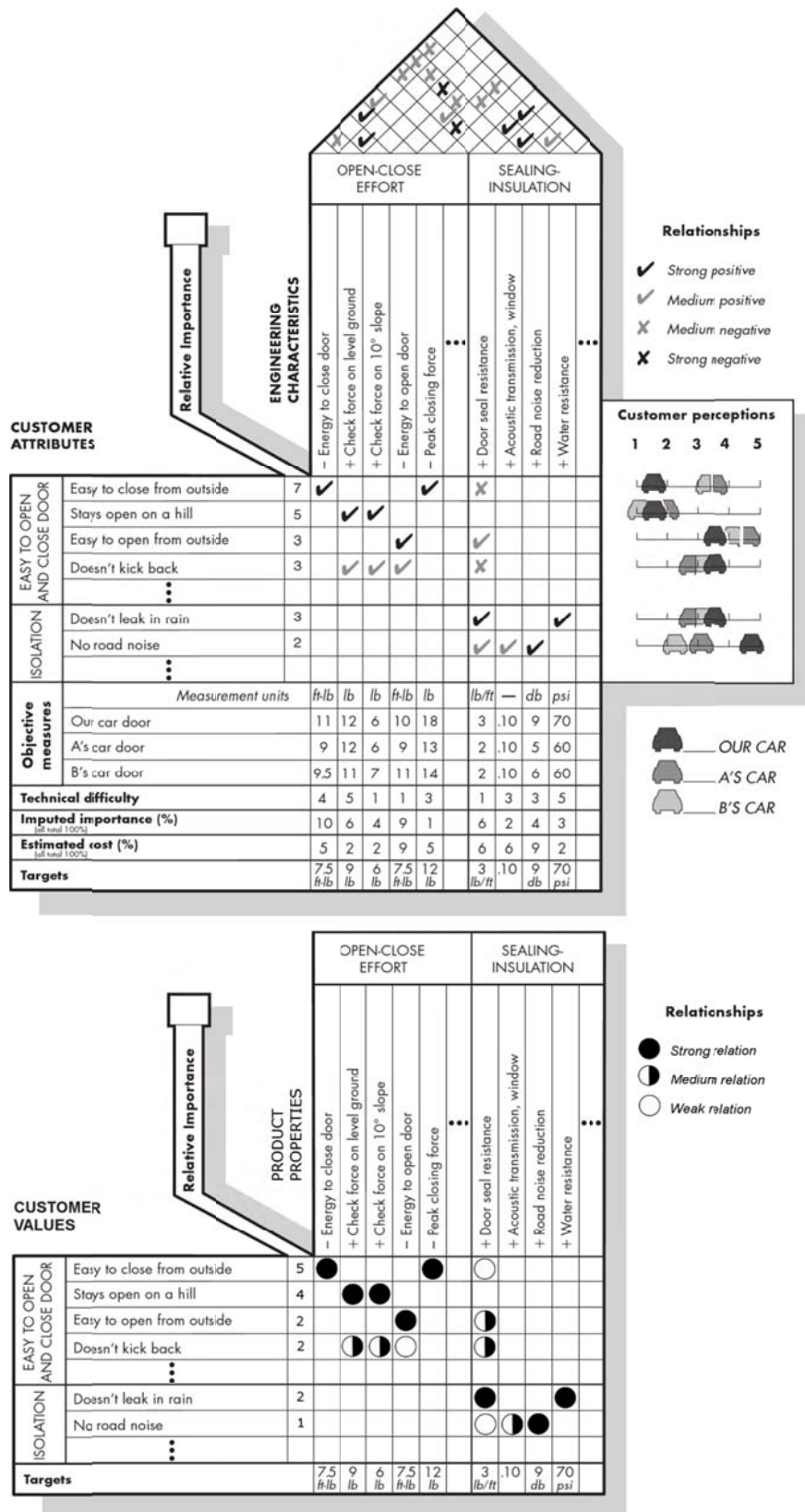


Figure 6. QFD as used in House of Quality (top) and MFD (bottom)

There has been academic criticism against QFD (Short et al. 2009), as well as a commonly encountered skepticism based on the opinions that (a) there is a massive work effort to populate the matrices and (b) it seldom leads to anything. The implementation of QFD in MFD addresses both of these points. First, it reduces the work load by cutting out the roof, at least as a mandatory component. Second, it uses the Product Properties (referred to as Engineering Characteristics in the graphic in Figure 6) to the Technical Solutions through the use of the Design Property Matrix, thereby “closing the loop” and making the QFD more conclusive. (Bylund, Wolf & Mazur 2009) propose a variation of QFD they call Blitz QFD, which is faster.

2.2.3 Hierarchical Clustering

Hierarchical Clustering (see, for example, Romesburg 2004) is used to bring structure into large two-dimensional arrays of data, where there is an underlying pattern waiting to emerge. A number of objects are described on a pre-defined number of dimensions, meaning each object gets a score on several pre-defined “questions”. The values can be continuous or discrete. The values are seen as coordinates in a multidimensional space. Points are considered close to one another if the distance between them is low. Distance can be calculated using the Pythagorean theorem (square root of the sum of the squares of the differences of each coordinate-pair) or some other metric. In the end, the distance relations are shown in a Dendrogram (tree-graph), which allows the person interpreting the data to view the points as individual points, clusters of points, clusters of clusters etc.

In MFD, hierarchical clustering is used to generate Modules. Modules are clusters of Technical Solutions that seem similar in their Product Property and/or Module Driver scoring patterns. The use of dendrograms in MFD to do clustering was pioneered by (Stake 2000) and has since been studied by others (Hölttä-Otto et al 2008). Dendrograms do not prescribe the number of modules – that is left up to the person interpreting the dendrograms. Dendrograms can be used for Quality Assurance work during MFD, not just for module generation. Any of the the three key matrices in MFD can be analyzed using dendrograms.

The following graphic shows how a DPM may be transformed into a Dendrogram using hierarchical clustering. The example uses a simplified cordless hand vacuum cleaner.

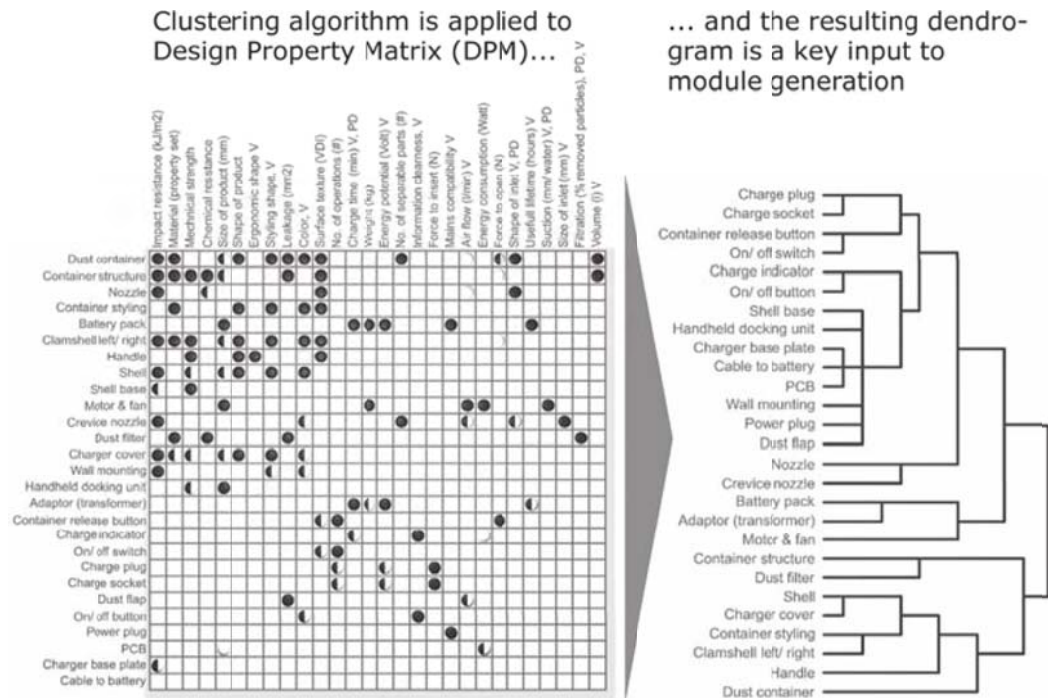


Figure 7. How a DPM is transformed into a dendrogram

2.2.4 Design Structure Matrix

Although this thesis is somewhat tilted toward MFD, two of the papers (C and D) deal with DSM (Eppinger, Whitney & Smith 1994). Clustering works on a completely different principle in DSM, and Hierarchical Clustering cannot be applied. The reason is DSM clusters are defined in a way that is fundamentally different. While MFD is concerned with similarity (e.g. scoring patterns are similar), DSM is concerned with *coupling*. The degree of coupling between two Components (this is the DSM terminology corresponding to Technical Solution) is defined as the amount of Interaction between them. A good module is one where the Components have strong Interactions between them, within the module, but weak or no interactions with Components in other modules. That is the only clustering rule in DSM, no other consideration is made, which implies that unless some explicit mechanism is added to deal with Customer Requirements or Company Strategy, those aspects of module generation are absent in DSM. Therefore, DSM may be thought of as an engineering driven approach. Attempts have been made (Blackenfelt 2000) to extend DSM with company strategy, and although it is possible to do so (two matrices are required), module clustering becomes more complex, and no algorithm is available.

A DSM matrix is square, whereas a DPM in general is rectangular. As pointed out by (Li 2011), the techniques for DSM clustering cannot be directly adapted for the clustering or decomposition of a rectangular matrix due to different matrix formats.

The graphic in Figure 8 shows a network of interconnected nodes (Components) and the DSM representation.

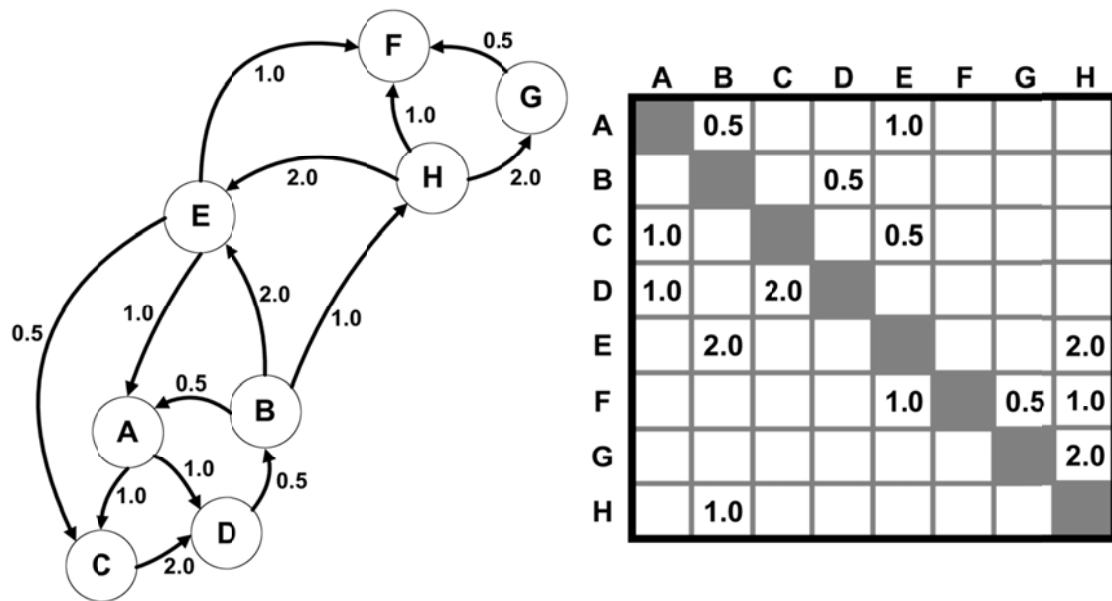


Figure 8. DSM representation of connected Components

Algorithms for clustering DSM have been proposed by (Thebeau 2001A, Thebeau 2001B) and others. You may read more about that in Paper D.

2.2.5 Function structure heuristics

Function structure heuristics were proposed by (Stone, Wood & Crawford 2000). To use these heuristics, you would first represent the product or the system as a Function Structure diagram. Figure 9 shows a very simplified function structure diagram of a cordless hand vacuum cleaner.

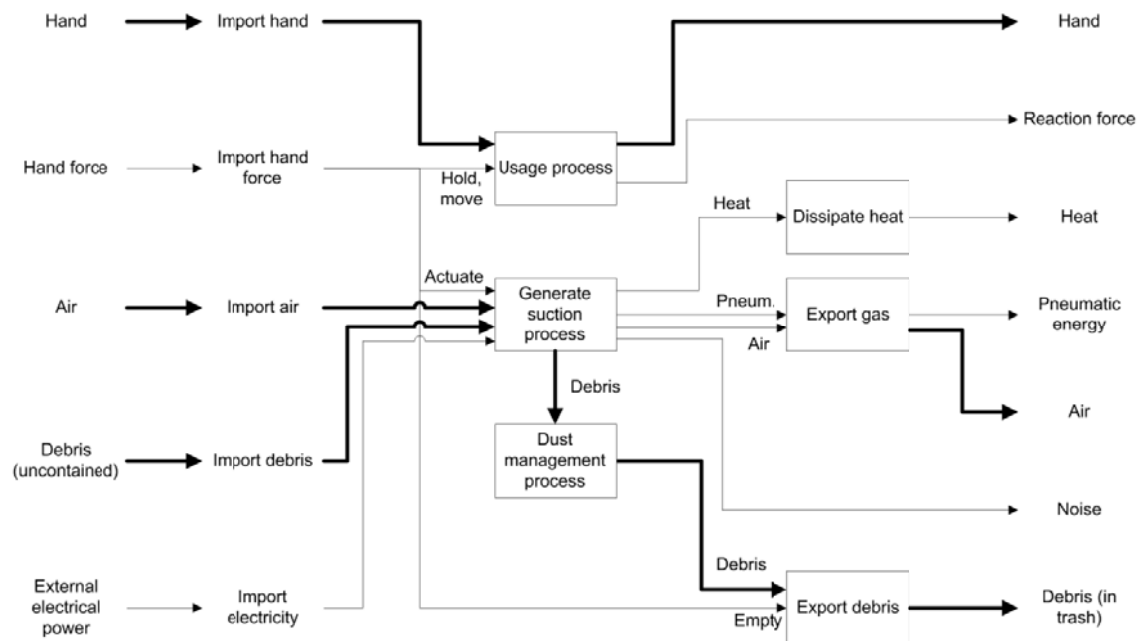


Figure 9. Function structure diagram of a cordless hand vacuum cleaner (simplified)

Traditionally, thick lines are used for matter, thin lines for energy in different forms, and dotted lines for information or signals. Ideally, each function should be described using an action verb that details the type of transformation taking place, and a noun that defines the object of that action. A good example might be “generate suction”. The input might be energy in the form of rotational torque, and the output might be the pressure differential between inlet and outlet on the rotating impeller.

The heuristics proposed by (Stone, Wood & Crawford 2000) are shown in Figure 10. The Dominant flow heuristic predicts that functions involved in the same flow of matter, energy, or information should form a module. In a handheld vacuum cleaner, there is a flow of air from nozzle through a duct to the vortex generator: this forms a module, as predicted by Dominant flow. The Conversion-transmission heuristics predicts that when a flow is transformed from one type to another, and subsequently transmitted, those functions should form a module. Mechanical torque is generated in an electrical motor and then transmitted to through a shaft: this forms a natural module by that heuristic. Branching-combining, finally, dictates an interface where a flow branches or combines. A good example may be the bus in a computer, where boards can be added for increased memory, improved graphics etc.

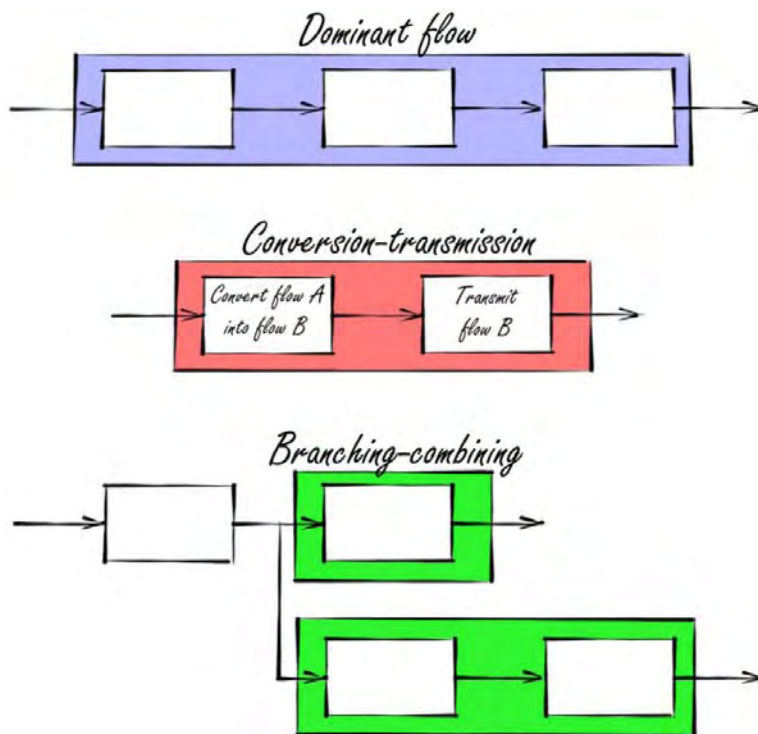


Figure 10. Function structure heuristics (adapted from Stone, Wood & Crawford 2000)

2.2.6 Clustering algorithms for DSM

IGTA (Idicula 1995; Gutierrez Fernandez 1998; Thebeau 2001A) was translated from C into Matlab by (Thebeau 2001B). The algorithm attempts to minimize the value of an objective function, TotalCost, by moving one element at a time. The value of TotalCost is a measurement of the “goodness” of the configuration: the lower the value, the better. The algorithm is stochastic, meaning elements are picked at random. An approach similar to (Thebeau 2001B) but with a different objective function was used by (Whitfield, Smith & Duffy 2002).

Genetic Algorithms (GAs) were explored by (Yu, Yassine & Goldberg 2007) who proposed a set of metrics for DSM optimization, building on information theoretical metrics, combined with GA. An improved metric was introduced by (Helmer, Yassine & Meier 2010), also with GA.

An algorithm which may be adapted for the purposes of clustering was presented by (Li 2011). The associated Matlab source code is publicly available (Li 2010).

Architecture generation is explored by (Wyatt, Wynn & Jarrett 2012) using a method that could be applied before DSM. Their algorithm generates possible solutions by adding and deleting components or relations, in accordance with certain predefined rules. Their software environment uses the Cambridge Advanced Modeller software framework (Wynn et al. 2009).

2.2.7 Modular Function Deployment (MFD)

MFD uses three interlinked matrices to integrate the Voice of Customer, the Voice of Engineering, and the Voice of the Company to predict a modular product architecture. Building on research conducted in the 1990s, this approach to modularity was described by (Erixon 1998) and subsequently improved (Nilsson & Erixon 1998) with the addition of the Design Property Matrix (DPM). Paper A offers a brief description of MFD and the three key matrices, the QFD, the DPM, and the Module Indication Matrix (MIM), which interrelates the Technical Solutions with the company strategy, using Module Drivers. MFD is compared qualitatively with four other approaches in paper B.

The graphic in Figure 11 shows an example simplified Product Management Map (PMM) for a cordless handheld vacuum cleaner. The first matrix, the QFD, interrelates Customer Requirements and Product Properties. Product Properties should be measurable, controllable, and solution-free. The QFD in this example uses shaded circles to signify strong, medium, and weak relations, in addition to no relation (no circle). A dark circle, such as the relation between “Can pick up all the dirt” and “Power (V)” signifies that there is a strong relation. A change in the battery voltage – which determines the available power – has a strong impact on the ability to pick up dirt.

The second matrix, the DPM, relates Product Properties and Technical Solutions. Technical Solutions embody functions required in the product. Battery voltage is provided by a battery pack, for example. To change the battery voltage, we would expect to make modifications to the battery pack, or possibly select a new battery technology with a different cell voltage.

The third matrix, the MIM, relates Technical Solutions to Module Drivers, the MFD-specific term for the company strategy. This example uses five of the twelve Module Drivers. The significance of the scoring in the column for Common Unit, for example, is those Technical Solutions come in one single version only, e.g., all the cordless handheld vacuum cleaners we plan to build using our modular product architecture use the same Clamshell, Exhaust grate, Microswitch, Release spring, and Impeller.

To generate viable modules, MFD looks at the scoring of the DPM and MIM, to find Technical Solutions that are similar in their scoring-patterns. For small matrices, this can be done visually, but for larger matrices, statistical software is typically used. By visual inspection of Figure 11, we can see the scoring for Power button, Styling handle, Escutcheon, and Dust bin are virtually the same: they all have color-variation and the Module Driver is Styling, e.g., we want to use these Technical Solutions to create visual variation. Could they be the same module? To determine whether that is a viable module, something needs to be

known about the product geometry. The Power button moves in relation to the handle, so that needs to be a separate module. The Dust bin needs to be removed and emptied, so that has to be a separate module. The Styling handle and Escutcheon are physically close, and could easily be designed as one single piece, and would make a useful module. The decision to integrate them might also save time in assembly.

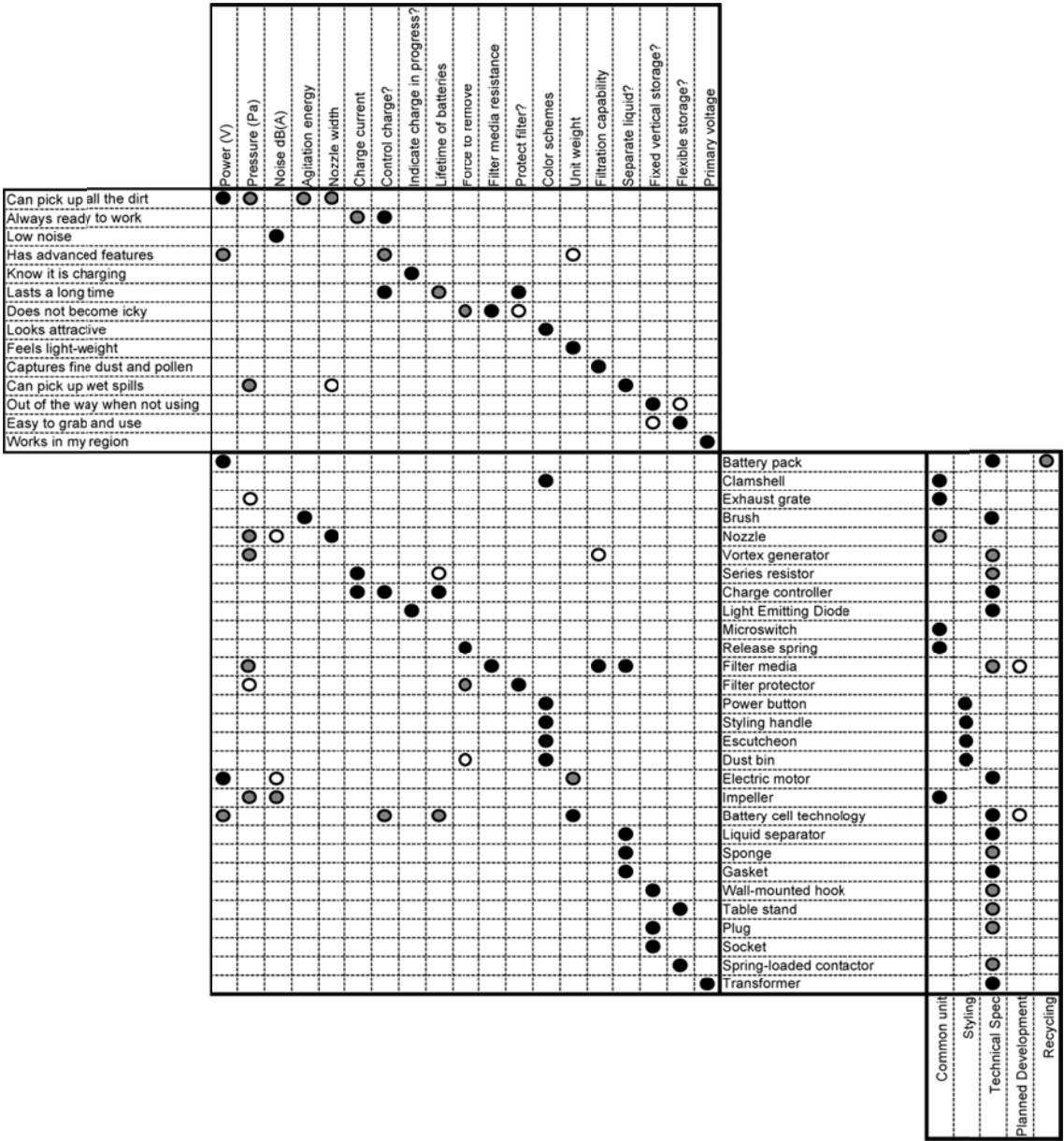


Figure 11. Example Product Management Map (PMM) for cordless handheld vacuum cleaner

2.3 Concepts summary

Figure 12 summarizes the theory used in each of the papers.

Theory of	Paper			
	A	B	C	D
Technical Systems	✓			
QFD	✓	✓		
Hierarch. Clust.	✓	✓	✓	
DSM		✓	✓	✓
Func. Struct. Heur.	✓	✓	✓	
Clustering for DSM				✓
MFD	✓			

Figure 12. Summary of theory used in each of the papers

3 RESEARCH METHODOLOGY

The methodology behind the present research is described and classified using a framework for qualitative research.

3.1 Scientific method

The basic steps of the scientific method are shown in Figure 13.

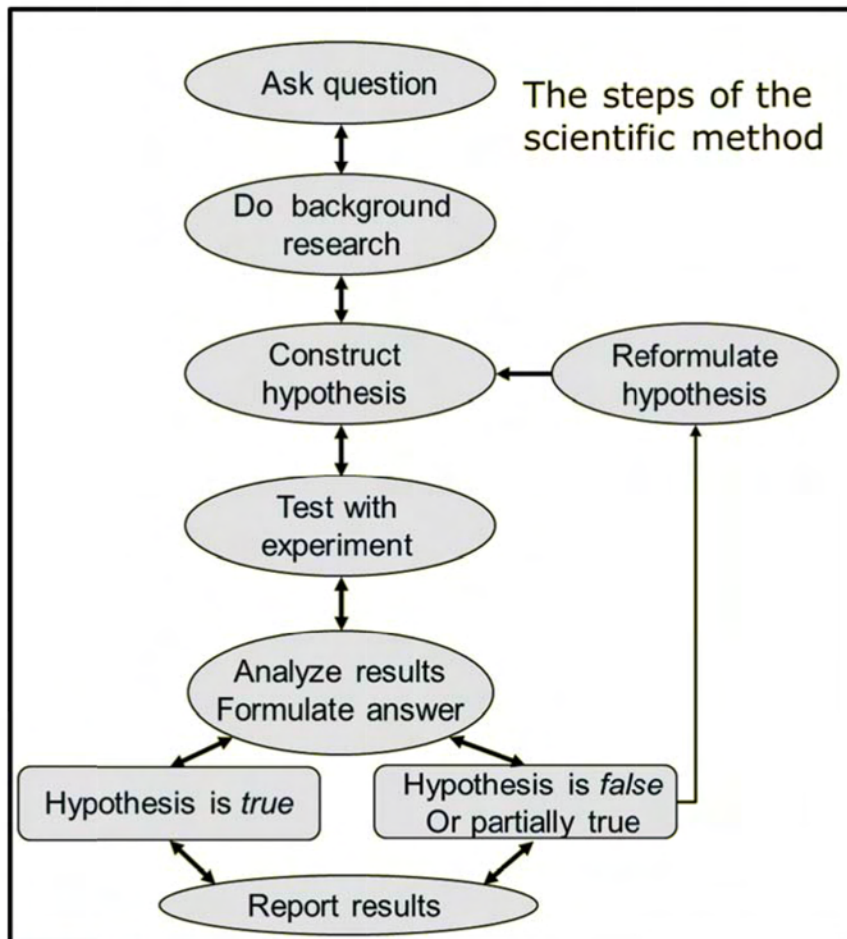


Figure 13. Scientific method

In the case of “desk research”, this model may look a little different. For example, we have no scientific measuring device for collecting data from experiments. An experience from a project is a type of experience, but the learnings are of a qualitative nature, and much less quantitative than in the natural sciences, for example.

A diagram outlining the steps of the qualitative research process can be found in (Backman 1998). The graphic in Figure 14 has been translated into English from the original Swedish, by the author.

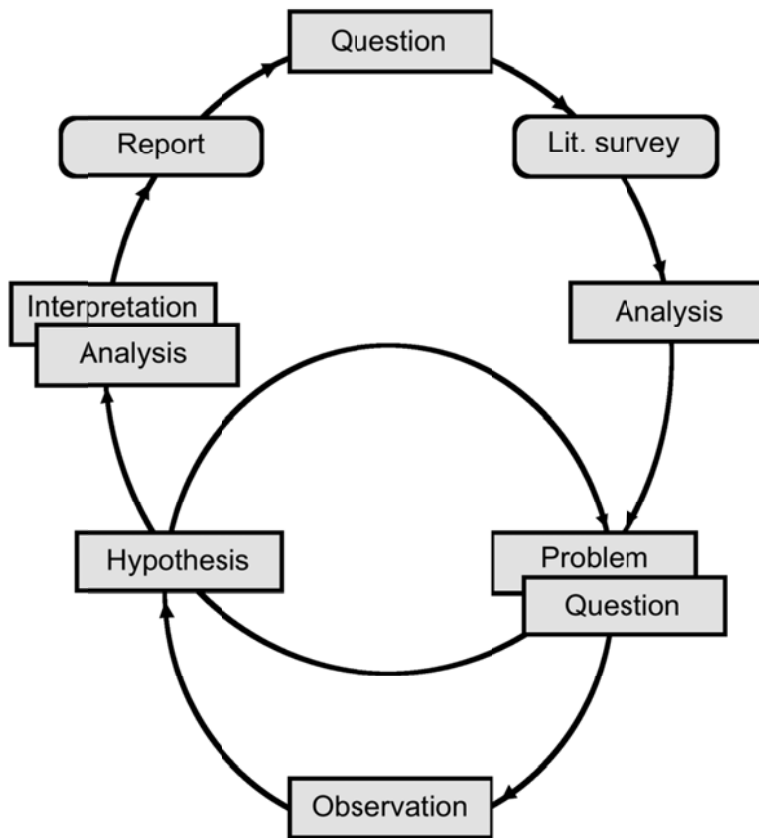


Figure 14. Scientific research process, qualitative, after (Backman 1998)

Using the structure from the diagram above, let us step through the research process used in this thesis.

3.1.1 Question

In qualitative research – as pointed out by (Backman 1998) – the question is often of a practical or applied nature, and the question may be stated in terms of “how” something happens or “why” something happens or is the way it is. In the present work, we ask ourselves how something we know how to do can be done better, more efficiently, or be applied to a new domain of problems:

- *How* can we improve the selection of properties in MFD, to obtain results that obey physics and the required product geometry?
- *How* can we devise selection criteria for the purpose of method evaluation, without relying exclusively on subjective criteria?

- *How* can we apply existing tools like MFD and DSM to novel product types in an early phase of product development?
- *How* can we make DSM clustering algorithms faster?

3.1.2 Literature survey

The literature survey does not happen once, typically. The papers in the present thesis have been influenced by impressions from papers delivered by other researchers at conferences, books, exchange of ideas with other researchers and project team members, professional colleagues, own ideas accumulated from previous projects etc. Section 2.2 lists the main influences on the works in the present thesis.

3.1.3 Analysis / object of study

The term “analysis” is in reference to the choice of object of analysis. In a case study, the object of analysis would be data from the project. Thus, in papers A, C, and D, there was a clear object of analysis. Paper A was the modular structure of a cordless hand vacuum cleaner, paper C the Forwarder with hybrid drive train, and D the new proposed algorithm operating on a cordless hand vacuum cleaner again, compared to the old algorithm. In paper B – the paper that aims to create selection criteria, the object of study was the set of modularity methods itself.

3.1.4 Problem / question

In papers A, C, and D, the formulation of the research question was relatively straight forward:

Paper A – problem “statistical methods for generating modules generate poor output” – question “how can we introduce new data to make the output more useful?”

Paper C – problem “existing methods are usually applied to products that are well understood and have been around for some time” – question “how can we apply them to new products at an early phase in development?”

Paper D – problem “DSM clustering algorithms are slow, and for practical use in real projects they would have to be much faster” – question “how can we make modifications to increase speed substantially?”

In paper B, the research question shifted somewhat as the research was conducted. The original research question was to attempt to assess, objectively, which of the existing fundamental or hybrid approaches to modularity is best. This research question does not seem to have a clear-cut answer, for at least two obvious reasons: first, it depends on the situation and second, it depends on the criteria used for the evaluation, and selection of criteria is mostly subjective. To determine the criteria, a number of academic sources were used, but the author had a desire to (a) integrate experience-based criteria and (b) condense the list to a new set of exhaustive but orthogonal (e.g., independent) criteria. To create such a new list of criteria, a method based on pairwise comparison was used, followed by statistical processing. That method became the real focus of the research paper.

3.1.5 Observations

With the exception of paper D, which was conducted in a client setting (e.g., a real project), the observations are made “at the desk”. In paper A, the output of a standard MFD was compared with the output from an enhanced MFD, using the new properties proposed in the paper. In paper B, the data was compiled and analyzed by the author. In paper C, the

predicted modular structure of the hybrid Forwarder was analyzed “at the desk”. Finally, in paper D, the comparison of the execution times and output of the two algorithms was done by the author using his own computer equipment and software setup.

3.1.6 Hypothesis

The hypothesis phase mostly preceded the observation phase. The only exception was paper B, see below.

The hypothesis in paper A was that geometrical data, dominant flow, options, and module driver compatibility could all be added to MFD to make increase the likelihood that output would be useful. The hypothesis of the usefulness of the geometrical data was based on the 2003 Operator Seat client project (unpublished material). The usefulness of the dominant flow heuristic was based on work with function structure diagrams for the purpose of developing modularity training material with a major client during 2004-2006.

The hypothesis in paper B was formulated after the author went through the academic material on the topic and discovered that each author had a unique set of criteria. The hypothesis was that it should still be possible to ascertain how similar any two criteria are, and that this could be done with a mental process since the statistical process would even out any individual fluctuations or inconsistencies.

The hypothesis in paper C was MFD and DSM could both be used, but some qualitative interpretation of the output would be required.

Finally, in paper D, the hypothesis was the Matlab code of IGTA could be restructured to take advantage of the matrix operations more efficiently, and that memory could be introduced to make the algorithm converge more rapidly.

3.1.7 Analysis / interpretation

In the model presented by (Backman 1998), the analysis and interpretation phase is emphasized as potentially being the most demanding in qualitative research. Papers B and C are highly qualitative, whereas papers A and D are more quantitative.

Paper A – analysis is based on the number of “flat subtrees”, as described in the paper. The interpretation is with more data available to the hierarchical clustering algorithm, the output thus generated is more conclusive.

Paper B – qualitative assessment of data derived as numbers, but really based on pairwise subjective comparisons.

Paper C – qualitative comparison with a discussion of pros and cons of the proposed architectures generated by MFD and DSM.

Paper D – quantitative assessment based on the timed execution time of each version of the algorithm. Quality of solution obtained was based on a plot of the calculated “cost” of 10 000 runs.

3.1.8 Report

All the papers were submitted to conferences. At the time of writing (May 2012), paper D has been accepted for publication in August of 2012. Conference papers go through peer-review.

4 SUMMARY OF RESULTS AND APPENDED PAPERS

This chapter summarizes the results from the appended papers and division of works for each paper is presented

4.1 Introduction

Figure 15 illustrates how the papers are interrelated, and the sources that influenced or inspired each paper.

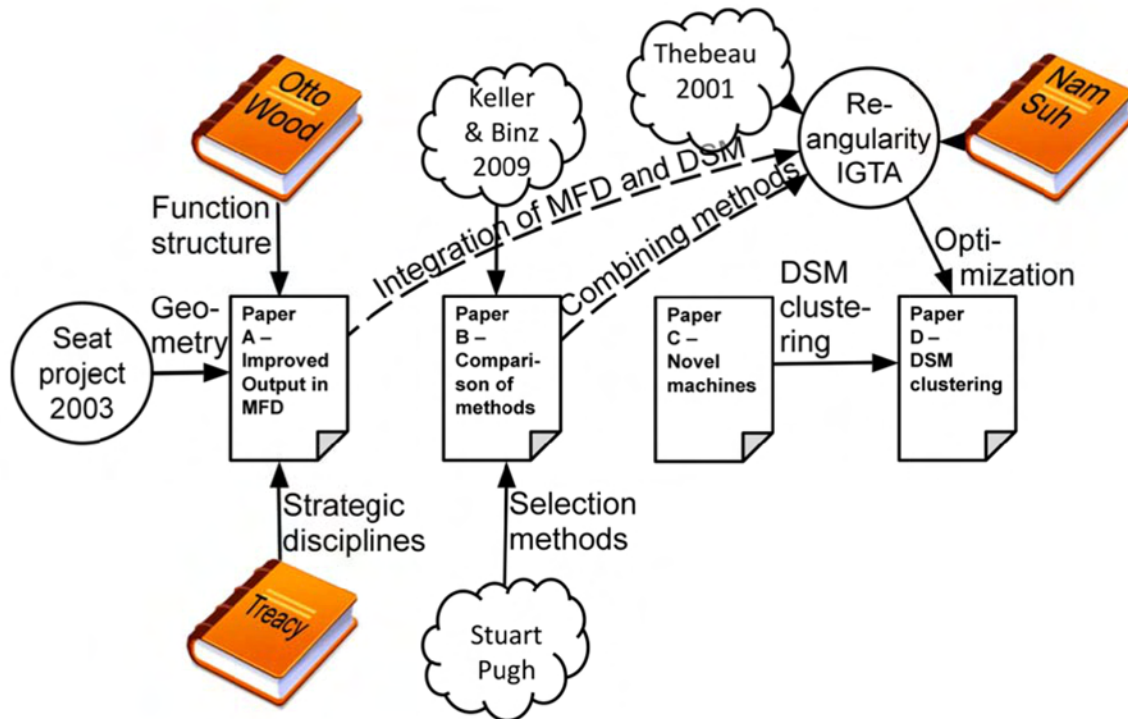


Figure 15. Interrelation of papers

4.2 Paper A – Improved output

4.2.1 Background

There were two sources of inspiration for this paper. The first was a client project conducted in 2003 together with Dr. Gunnar Erixon, a professional colleague at Modular Management and an important architecture mentor. The objective was to create a modular structure for the operator seat in a construction vehicle. The client was a large European manufacturer of

Construction Vehicles. This client did not manufacture seats. Seats were purchased from seven different suppliers, and there was a total of more than fifty seat types available. Not all seat types fit in all vehicles. In addition, there were quality issues with many of the models. The client was interested in creating a modular operator seat, and to select two strategic suppliers, and get rid of the other suppliers. Modular Function Deployment, MFD, was used to create the key matrices which then get plugged into a computer algorithm which then generates proposed modules.

The issue was that an operator seat, in order to be a seat, must obey certain geometrical necessities. The seat follows the human body, so we would expect a seat to have a seat cushion, back rest, head rest, and arm rests in certain locations. The modules that came out of the software did not seem to respect the required geometry of a seat. The algorithm was not at fault: it was faithfully producing a Dendrogram of the matrix data provided. The problem was no geometrical information had been supplied, so it was unreasonable to expect the algorithm to “know” the things a human knows about a seat. Thus, the idea of geometrical properties was born. The seat was defined into a number of regions. The boundaries between these regions were called region interfaces. Each Technical Solution in the seat system could be tagged by how close it needs to be to each of these region interfaces. The seat recliner, for example, must sit between the seat cushion and the back rest, it cannot sit in the headrest. When this information was included, the algorithm generated output that made much more sense, and the general feeling in the team was the new property type thus introduced had been highly useful.

The second inspiration was an observation from modular product architecture training. The training material used a cordless handheld vacuum cleaner, the Dustbuster® from Black & Decker as the example product. Like a seat, this product also has to obey certain geometrical necessities (for example, the suction and the exhaust cannot be in the same location).

4.2.2 Findings

Paper A presents three results. First, it shows how four proposed new property types, the Convergence properties, can be used to generate modular product architecture that respects – among other things – product geometry and the necessary exchange of matter, energy, or information between Technical Solutions. Second, the paper demonstrates how the proposed Convergence properties can be represented in a matrix format, including the Dominant flow, which is normally shown in a Function structure diagram. Third, the paper proposes the use of “large flat subtrees” as a measure of missing data. Large flat subtrees indicate lack of information, which generally diminishes the practical usefulness of the Dendrogram output for purposes of architecture definition.

In addition to product geometry and the exchange of matter, energy, or information, Paper A also explores how technical options can be integrated, and how module driver compatibility can be described. Paper A uses a cordless handheld vacuum cleaner as a study object.

4.3 Paper B – Qualitative comparison

4.3.1 Background

Paper A was presented at ICED 2009 in Stanford. At that conference, the author attended a presentation by a German Ph.D. student, Alexander Keller, whose research topic was quite abstract indeed: to construct a formal approach by which methods can be evaluated for efficiency and effectiveness (Keller & Binz 2009). The idea of comparing modularity

methods seemed like it would have practical application. Real projects consist of people with experience of different methods. Positive experience leads team members to want to apply the method again in their next project. A negative experience might be a deterrent. Very little time may be used in the actual selection of the method itself. Although project experience has shown Modular Function Deployment to be useful in a range of projects, we must recognize that each method has its own set of limitations. The relative strength of MFD may be that it integrated Customer Requirements and Company Strategy. This is powerful in many projects, but what if the project scope is pure re-engineering for the purpose of reducing product assembly cost or material cost? In those cases, Customer Requirements may be out of scope (e.g., the product must do exactly the same thing), and Company Strategy might be irrelevant (e.g., do the same thing but at lower cost). In such a scenario, a method focused on the way components actually interact, such as DSM, might be more relevant.

The problem in evaluating methods, of course, is that the very process of choosing the selection criteria is tainted by our opinion of what's important, and that is dictated largely by the experience we have. A seemingly objective evaluation may not be objective at all, because the criteria have been selected in favor of one particular method, perhaps with the objective of showing that particular method to be best! This happens in industry, too, when engineers using Pugh as a concept selection tool (Pugh 1991) go into the evaluation with a favorite concept, and select the criteria and weights to favor that particular outcome. (Stuart Pugh understood this risk and presented his famous method as a concept *generation* tool. To discourage use as a concept *selection* tool, he did not recommend the use of weights.)

Thus, the question became whether we can take evaluation criteria from academic studies, and integrate those with criteria that we know to be important from experience, to come up with a comprehensive list of criteria that is a little more objective than what we would get if we just sat down with a blank piece of paper and started writing. Why not rely exclusively on academic studies, to avoid any trace of subjectivity? Because projects generate important learnings. We do not wish to completely discard our experience, but we also do not want the evaluation to be driven exclusively by experience. Several sources were compared, and using an approach based on pairwise comparisons followed by hierarchical clustering, a Dendrogram of evaluation criteria could be generated.

4.3.2 Findings

The paper takes the evaluation criteria from three academic sources and integrates them with the author's experience-based criteria, makes a pairwise comparison of the degree of similarity between each pair of criteria in the combined list, and using a Dendrogram representation, finds a set of criteria (a) on an appropriate level of detail, (b) that do not overlap and (c) allow for a qualitative comparison of the methods.

In the second half of the paper, three fundamental methods (DSM, MFD, and Function-structure heuristics) and two hybrid methods (FS-DSM and eISM) are evaluated by the author, using the derived criteria. This represents the author's opinion.

In its conclusion, the paper states that all methods have their unique set of strengths and weaknesses, and that no single method has only strengths. It is possible to construct hybrid approaches to modularity, such as the one proposed by (Blackenfelt 2000), but typically these approaches have a new set of disadvantages. Very often, the hybrid approaches have some new difficulty when it comes to actual module generation or clustering. The method proposed by Blackenfelt, for example, assumes manual module generation, and no automatic algorithm

is proposed, only a three-stage heuristic approach which is shown in Chapter 5. The approach by (Sellgren & Andersson 2005) uses three matrices in a format similar to that used in MFD, with two key differences. First, instead of Product Properties, the authors use Functions. Second, instead of the Module Indication Matrix (MIM), the authors use a DSM to interrelate the Components. The purpose of the paper is to define this new format and discuss how it may be used. No suggestion is made with regard to the actual clustering.

Finally, paper C makes the observation or comment that non-matrix based methods may be inherently more difficult to use in large projects with many interrelated Technical Solutions or components. This is discussed in Chapter 5.

4.4 Paper C – Modularization of novel machines

4.4.1 Background

The inspiration here came from collaboration with Doctor Ulf Sellgren. There was a major research project at KTH Royal Institute of Technology involving a type of machine called a Forwarder. Forwarders are used in the forestry industry. Among many other topics, the possibility of using a hybrid drive system was being explored, mainly for environmental reasons. Doctor Sellgren proposed that MFD and DSM may both be applicable for this type of system, and that it may be interesting to see in an “artificial case” how well the output of these two methods works in practice (and possibly support each other), even if very little is detail is known about the hybrid drive system.

4.4.2 Findings

In this paper, MFD and DSM are applied to a new type of machine, a Forwarder with a hybrid drive train. The paper uses MFD with Convergence Properties proposed in Paper A, as well as DSM clustering, and compares the outputs. The paper compares the output of MFD and DSM and makes some preliminary conclusions about interfaces on a subsystem-level, in particular with regard to a modular structure of the electronic inverter using plug in converter modules that connect to a power bus and receive control signals from a control-unit that supports several different system configurations. The paper identifies the inverter as the single most challenging subsystem in terms of its complexity and overall impact on the performance of the product.

4.5 Paper D – DSM Clustering

4.5.1 Background

The way this paper came about is probably a good example of the “nonlinear” and sometimes unpredictable way in which research happens. During the work on another paper, the author came across different algorithms for clustering a Design Structure Matrix. One, which was published in 2001, is the inaccurately named “Thebeau’s algorithm”, which builds heavily on the work by two previous researchers, John Idicula (Idicula 1995) and Carlos Iñaki Gutierrez Fernandez (Gutierrez Fernandez 1998). We shall refer to the algorithm as the Idicula-Gutierrez-Thebeau Algorithm or IGTA for short. Upon reading the thesis (Thebeau 2001A) the idea was born to extend the formulas to encompass MFD. Although the work of detailing the algorithm extended over more than a year, the first formulas and simulations conducted in June of 2010 indicated it could work. The final algorithm was coded in Matlab, and named R-IGTA, with the R signifying Reangularity (Suh 1990). The actual algorithm runs were

quite time consuming, and it became apparent the core of the algorithm had to be modified as to execute more quickly. Two such algorithmic changes were made, resulting in a good speed improvement. The way these algorithmic changes were made, they could also be applied to pure DSM clustering, quite regardless of MFD. Thus, the term IGTA-plus was coined to signify the original algorithm, IGTA, but with the algorithmic improvements that made it almost eight times faster. This resulted in a paper that only deals with DSM and the algorithm itself.

4.5.2 Findings

Paper E uses an existing clustering algorithm for DSM based on (Thebeau 2001A) and adds two algorithmic improvements, increasing the execution speed by a factor of eight, and improving the quality of the output in the process. The paper shows the improved clustering algorithm as a flowchart.

5 DISCUSSION

This chapter discusses the results

5.1 Introduction

The previous section summarized the results of the papers. In this section, we will discuss some challenges in architecture research and in each of the papers.

5.2 Challenges in architecture research

One of the main challenges in Architecture research is there is no “best” modular architecture with which we can compare the results. In MFD, the architecture is a balance between strategic needs captured in the MIM and the “functional” needs captured in the DPM. Since every company has a unique strategy, the architectures will not come out the same. DPM represents a pure engineering view, but as pointed out by many researchers, there is not a single “correct” decomposition of a product, so even the list of Components may end up being different, leading to different module clusters in the end.

When we gauge the quality or usefulness of the output, we have to apply our judgement, and that is subjective. As researchers, we may even fall into the trap of viewing our own particular approach as uniquely well suited. Very rarely do you read a scientific paper that says “we devised a new algorithm, we tried it out, and it was a massive fiasco”. There might be ways of mitigating this. One is the approach based on the idea that you let several independent teams apply two or several algorithms, and then compare the outputs of the teams, to see if one algorithm consistently seems to outperform the other. That is at least a scientific approach. The problem is usually, the only people who are willing to participate in such trials are students, and they are not experienced designers of products, so they do not represent the learnings you would get with an experienced team of product designers. What scientists can do is to look at isolated case studies, where a new approach is tried out in a real project with real designers, but then of course it is very hard to try out several methods. In a real project, there is often very little time for “playing around”, so once the results are in, the team will move on. The third approach is “desk studies” where the scientist himself or herself tries out the algorithm. The disadvantage, of course, is we might view our own proposed algorithms as fantastic, and tend to overrate them.

So what is the undeniable scientific value in each of the Papers included in this thesis? Let us take another look, and boil it down to its core.

5.3 Paper A – Convergence properties

The idea of Convergence Properties touches something very important in Architecture research: what types of data should influence architecture decisions? How might the location of a proposed interface change, as new information is supplied to the algorithm, or irrelevant information gets deleted?

Most of the time, the real issue is not too much information, but too little information. Upon reflection, we might want to word that statement differently, because we are constantly

flooded with information – so how can there be too little? There is a flood of irrelevant information, of course, and only a minute fraction is useful in product architecture decisions. In a real project, we may be presented with dozens of documents and spreadsheets, all with product or market information, but typically in formats that make the data hard to compare, and very little is actually useful in predicting interfaces.

When the number of Geometrical properties goes up, the clustering algorithm will tend to reproduce the existing product structure. The graphic in Figure 16 shows a progressive use of geometrical properties from no Geometrical Properties to the lower right hand pane where four Geometrical Properties are used.

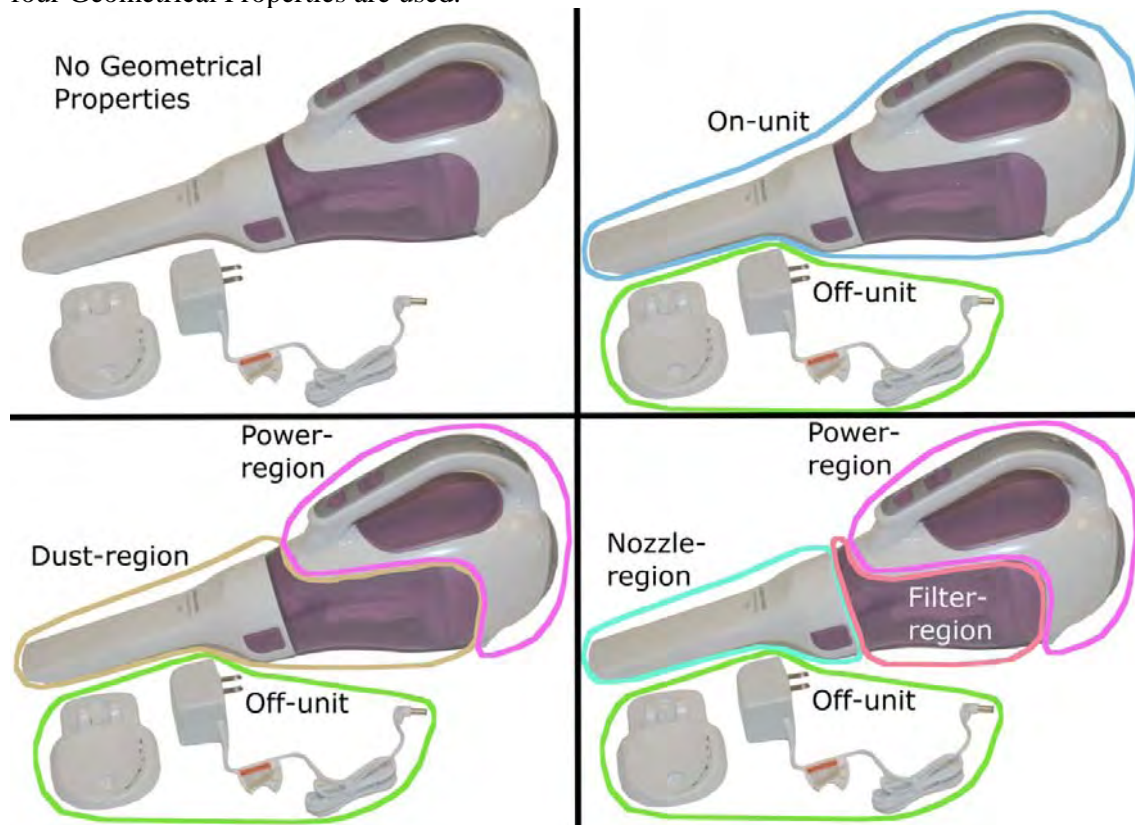


Figure 16. Four levels of geometrical information for a cordless handheld vacuum cleaner

As more Geometrical Properties get added, the relative influence of the Product Properties and Module Drivers decreases, and the clustering output becomes increasingly dictated by the geometrical structure of the existing product. This is undesired: why would we even perform clustering if we knew exactly what kind of output we wanted? On the other hand, with no Geometrical Properties, many proposed clusters may correspond to a bag of disconnected parts or violate the required shape of the product.

Clearly, there is some kind of “sweet spot” for Geometrical properties. We cannot know a priori how much geometrical information is required and that can only be determined through trials. In practical terms, several runs of module clustering can be performed, with different levels of product geometry, which allows for a comparison and determination.

5.4 Paper B – Qualitative comparison

The scientific value of this paper is the method used to merge criteria from several sources, including the author's own experience, to generate a set of consistent and non-overlapping criteria for evaluating methods.

Paper B also makes the claim that approaches based on matrices are easier in large projects, with many Technical Solutions. One approach, which does not use matrices, is based on Function Structure diagrams. An argument could be made that there is good software support even for flowcharts, and that this is not a real limitation, even for complex systems. A section from Tiimo Lehtonen's PhD dissertation (Lehtonen 2007) is reproduced below (pages 103-104 of Lehtonen's thesis). His block-diagram of a tunnel boring machine, reproduced in Figure 17, is included to give the reader some idea of the practical challenges of creating this representation, and updating with further detail as required. Not an easy task!

“In the research, we started with applying functional modularity. For this reason, a functional structure was drawn of the tunnel drilling rig that described the implementation used at the time. The structure was created in a highly pragmatic manner. The figure below features power input on the left and the machine control on the right. The presentation was large in size, and it is not possible to present it in the page format of a dissertation. The figure below will, however, provide an idea of the size of the modeling.

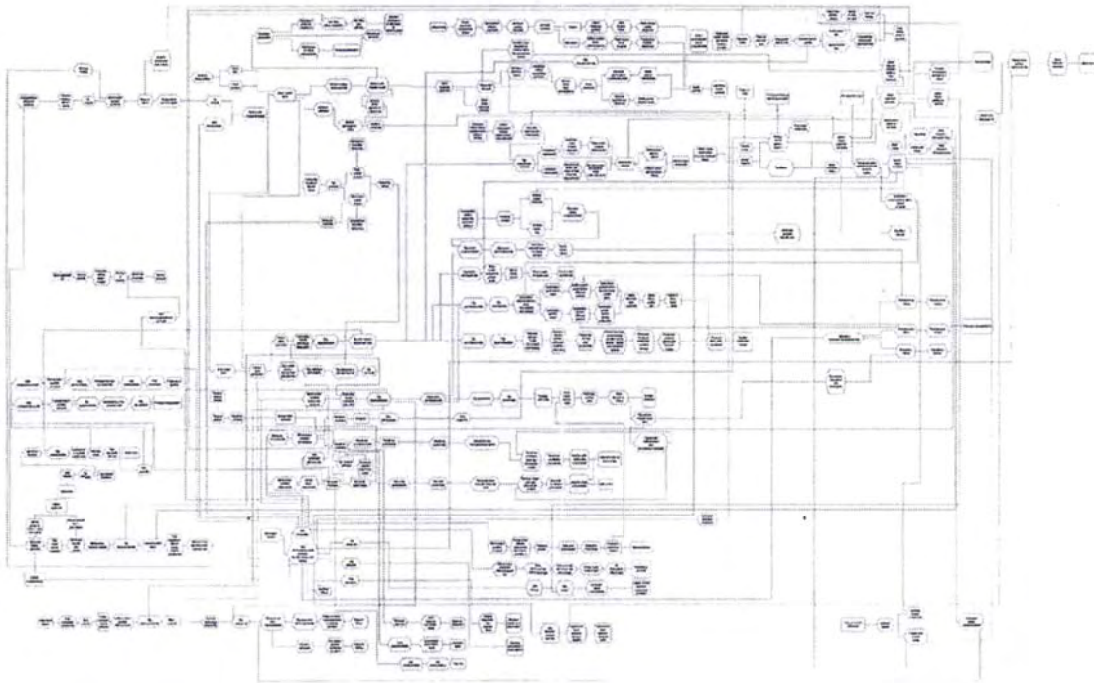


Figure 17. Lehtonen's block diagram of a tunnel boring machine has about 320 function blocks (Lehtonen 2007)

“The functional diagram of the Jumbo as an ‘as implemented’ model. Power input comes from the left and the machine control from the right. The figure does not indicate the mode of operations, but all functions in the moving and the drilling mode are drawn in the same diagram.

“The diagram was used to recognize functional modules. However, the work did not yield results. The modules to be outlined in the diagram could not be implemented in practice in the assembly structure, and they did not yield a necessary level of variation. **This empirical observation is the same that we deduced with the theoretical example in Chapter 5** [of Lehtonen’s thesis; bold font by Lehtonen]. After this, we proceeded with the work by moving on to matrix methods.”

Although flowchart software has functions that rearrange blocks to minimize crossing lines (“routing”), some human intervention is normally necessary. For products with the complexity of the drilling machine shown in Lehtonen’s graphic above, flowchart just do not seem to be a feasible approach. Better then to approach it with a matrix and tag the functions or technical solutions with the interaction types of interest.

5.5 Paper C – Hybrid drive

This paper tests the hypothesis that MFD and DSM can be used for novel machine types, like a hybrid Forwarder. There are several sources of complexity in a hybrid drive powered vehicle. One is the electronics, and one is the drive train itself. A series hybrid is simpler than a parallel, but a series hybrid is never as efficient as a parallel. A parallel hybrid drive system has something like the planetary gear in the Prius to add torques from two sources, the internal combustion engine and (in the Prius case) the MG1/MG2 motor-generator pair. In the Forwarder project, it was determined that supercapacitors would be used to yield high peak torques for digging out of the mud. From a theoretical perspective, they behave like batteries, but from an electronics perspective, they provide much higher peak currents, and may require other semiconductors, so there are some practical complications there. The paper also identifies that depending on the architecture of the hybrid drive system (series or parallel, regeneration or no regeneration), the electronics need to support many operating modes.

In a real project, we would have worked with skilled and experienced power electronics engineers to devise an architecture for the power electronics, which allows for all these operating modes. In this academic paper, the author had no such team on which to rely, so certain assumptions had to be made about the structure of the electronics. In that sense, the results may not be transferable to reality. The paper predicts that all the power conversion devices should be, essentially, plug-in units. Is this doable in reality? Hard to say. The author is aware of no academic papers that detail the structure of the Prius power electronics. Some documentation is available from Toyota, but it is quite high-level.

If the hypothesis is we can use MFD and DSM, it seems fair to say: yes it can be used in this context, but there will be more iterations. In a real project, the first conclusion might have been “try to devise all the power conversion units so they have the same interface to the power bus and the control bus”. The engineers might come back and say this is not doable. That would have to be fed back into MFD and DSM, the data rescored and the clusterings rerun. Would MFD and DSM add value, in this context? Probably, but the real “proof of the pudding” would be to actually do it with a real team, and that was not possible. Thus, we have to take the conclusions as indicative. Representations such as MFD and DSM may be used to predict where new or modified requirements will have an impact on the product.

5.6 Paper D – Improved clustering algorithm

The improved algorithm for DSM Clustering outlined in Paper D came about in a slightly convoluted way. As the author was investigating the inner workings of IGTA, for the purpose of expanding the problem domain to encompass MFD-clustering by including Reangularity (Suh 1990) in the objective function, it became clear that computational efficiency needed to be addressed. This new algorithm was given the name R-IGTA, where R signified Reangularity. A paper discussing R-IGTA is forthcoming.

The upper half of Figure 18 illustrates the logical relation between IGTA, IGTA-plus and R-IGTA. The lower half, however, shows the sequence in which these algorithms were devised.

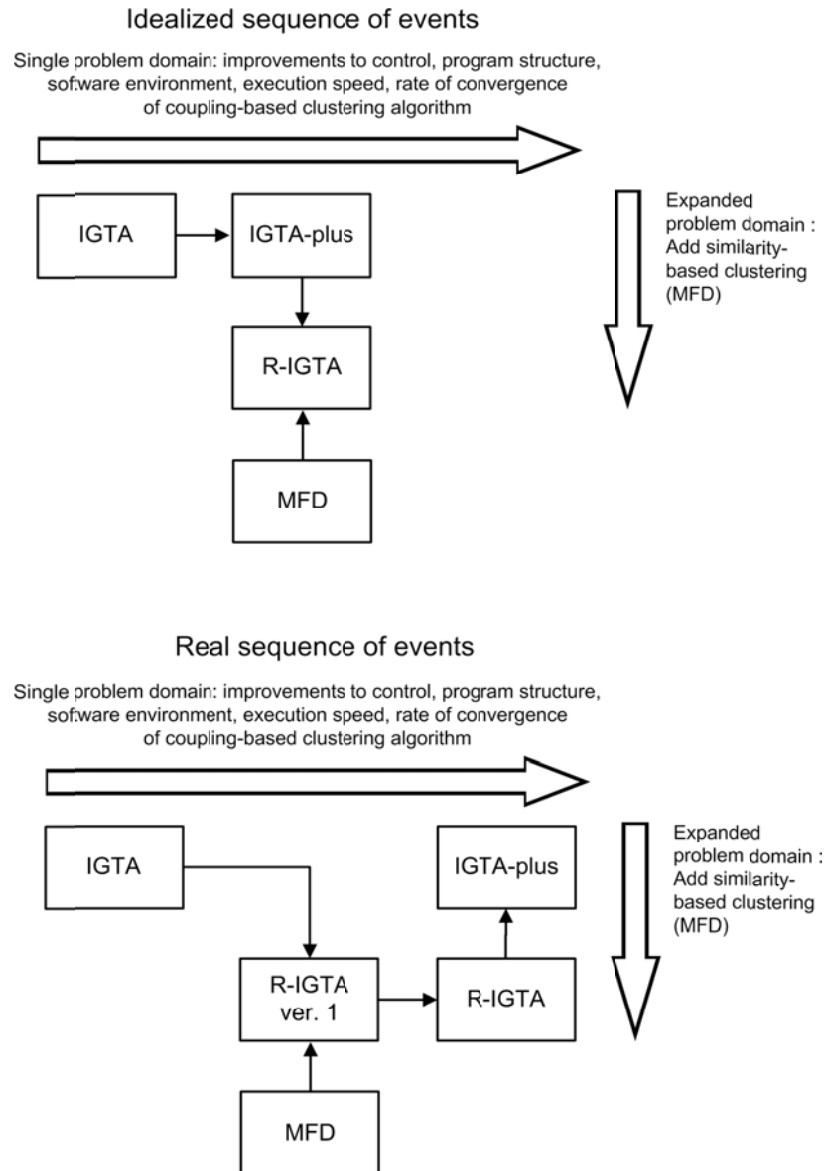


Figure 18. Evolution of IGTA-plus

IGTA-plus was derived from the computationally enhanced version of R-IGTA, essentially by stripping it of anything related to Reangularity.

The literature survey conducted as part of writing the paper on IGTA-plus and some of the research from the R-IGTA work indicated Genetic Algorithms are becoming a more and more dominant approach for DSM clustering algorithms. One paper (Yu et al 2007) describes an algorithm written in C++ for clustering a DSM using the principles of GA. An example problem involving 60 Components is used, and the execution time is listed as “about a day”. With 57 Components, IGTA-plus is able to perform one complete clustering in 0.34 seconds, meaning 10 000 complete runs can be concluded in about an hour. As described in paper E, a large number of runs are usually required to guarantee the best possible solution has been found. In real projects, the computational efficiency of the clustering algorithm is an issue. Waiting a full day for the output is inconvenient, especially if we imagine that several iterations may take place, where data is modified and the algorithm is re-run. GA may be powerful in many ways, but not nearly as fast.

5.7 Impact of research on consulting

Project work with real clients clearly impacts research: in fact, the topics of research are often born out of real problems faced in client engagements. To what extent does research have an impact on consulting? Specifically, to what extent are the research topics described in this thesis applicable to the context of consulting?

In the experience of the author, new consultant “tools” are the result of either technology “push” or “pull”. Technology push is when useful ideas are generated through research, own or external. Pull is when new ideas applied in project work seem applicable elsewhere. Some assessment is made as to what seems most promising, with a view to the balance of urgency versus time required to develop or document the idea.

When ideas are generated from technology push, an attempt is made to evaluate the usefulness in a real project, which may include internal projects in some cases. If deemed useful, the idea is documented and rolled out in the organization, usually following a pre-defined procedure similar to “stage gate” processes used in product development.

With regard to the papers in this thesis, some of the ideas in Paper A have been used in real project work. Specifically, geometrical properties have been used in some projects, and Option properties are used in almost every project. Paper B is something that could conceivably be used in the prioritization of research topics – e.g., a formalized way of prioritizing potential topics – but this is often done in an organic fashion. Paper C is related to novel products at a very early point in the development, and this has been a very rare type of client engagement, as the author’s experience goes, so MFD has never been used with DSM in the conceptual manner outlined in the paper. Paper D is related to a larger, ongoing research topic, that of clustering with respect to MFD and DSM simultaneously. The author hopes to test the new algorithm in one or two real client projects during 2012.

6 CONCLUSION AND FUTURE RESEARCH

This chapter presents conclusions and suggests possible future work

6.1 Introduction

As stated in section 1.7, the research question is whether we can improve the methods used for generating modular product architecture. In this section, we will offer an overall conclusion, as well as discuss possible future research.

6.2 Conclusions

This thesis deals with four specific research topics that all support the question whether existing methods can be improved in terms of their usefulness.

In general, the answer has to be stated as yes. In chapter 4 we looked at the answers to the research questions, and in chapter 5 we discussed the contribution to research in engineering design. Andersson (Andersson 2003) claims the complete rationale for a product cannot be elicited or represented, as it is not a finite task. If that is the case, it seems we cannot expect there to be a “perfect solution”, meaning neither the end goal (the design) or the process of getting there (the approach we take to design or in a narrower sense, the approach to modularity). Figure 4 positions the research topics in the present thesis in an early phase of product development, so we might ask to what extent we have improved the tools in this phase.

By focusing on method selection, application of existing methods to novel product types, selection of product properties, and finally supporting algorithms for clustering, we have shown how different stages of the Concept Generation phase can be improved through the use of new techniques or application of existing techniques in a new way.

Method selection is often completely overlooked. It is not easy to give clear-cut guidelines. To reengineer an existing product, in a situation with no or very few new customer requirements and no change of company strategy, DSM might be sufficient. It does a good job of describing component interactions, and takes an “engineering centric” view of the product. If the product is new, or there are many new customer requirements and/or a new company strategy, then MFD is probably a better choice. Because it uses three interlinked matrices, capable of handling at least four types of information (requirements, properties, technical solutions, and strategy) MFD is more flexible than DSM. The task of selecting a method must also include considerations such as time required to learn the tool, whether software is available to support data analysis, whether the task of writing specifications is supported etc. These are examples of subjective criteria we may wish to include in the selection itself, and we have seen that integrating such criteria is possible.

Whether approaches to modularity, such as DSM or MFD, really get used in development of novel product types is hard to say, and the present thesis does not answer that question. To answer such a question would mean interviewing James Dyson, for example, to find out

whether any particular structured method was used in the development of the famous cyclonic vacuum cleaner (Dyson 1986). The answer is probably no. In the case of a cyclonic separation system, Dyson's innovation essentially replaces the filter, e.g., essentially only one subsystem in the product. The situation may be different in the case of the Toyota Prius, where the entire drive train was replaced with a parallel hybrid. This is a much more complex system and a much bigger change, and it would be more reasonable to assume some structured process was used. However, we do not have easy access to the core development team at Toyota, and they do not openly talk about their work, which is part of Toyota's core Intellectual Property.

In the author's experience, both MFD and DSM suffer from similar problems in the usefulness of the output typically obtained on the first iteration. In many cases, the clustering algorithms will yield useful output for some of the technical solutions or components, but not all. This is caused by lack of relevant information, not by a deficiency in the algorithms themselves (which are generic, and can operate on very simple data or very complex data). In project work, whether MFD or DSM is used, teams typically create new data to address deficiencies in the previous iteration of output, and then re-run the algorithms to see if the new output is better than the previous output. After a few iterations, conclusions are drawn from the data and the output may be summarized in specifications, tables, matrices etc, as required. Through a more conscious use of properties that describe product geometry and required component interactions, the number of iterations required can be reduced, and a more useful output can be obtained on the first attempt. For existing products, both geometry and interaction are quite well understood, and this presents no significant obstacle during the data gathering phase. For new products, it will be more challenging, as there are fewer knowns and more unknowns, both geometrical and functional.

As presented in Figure 18, the author's real interest in DSM clustering was triggered by a research topic related to the "marriage" of MFD and DSM, which in turn builds on the structured comparison that identified these two approaches as being most complementary. In project work, the team often sits around and watches while the "clustering expert" runs the software that generates output, based on the matrices of data collectively populated by the team. It would be completely unrealistic to wait a full day for the output! Recent research in the field of product architecture identifies Genetic Algorithms as being the most promising path for flexible clustering algorithms (Simpson et al. 2011). One aspect which seems completely ignored is many such algorithms built on GA take on the order of a full day to complete, given a problem with roughly 60 components. Execution time definitely must be an evaluation criterion, and although algorithms that use a full day to complete may be interesting research prototypes, researchers should spend time trying to improve execution speeds to make them useful.

6.2 Future research

6.2.1 Observe development of novel product types

To the extent that we may gain access to projects that develop new product types, it would be interesting to evaluate the usage of MFD, DSM, or some other structured approach to modularity to determine their usefulness in this context. Such projects are often more or less secret, which is the real obstacle to such research.

6.2.2 Survey-based evaluation

The evaluation of five approaches to modularity could be repeated with a team, instead of as a one-person effort. Both the generation of experience-based criteria and the pairwise comparison would benefit from such a setup. It may also be possible to analyze the “Criteria-DSM” using IGTA-plus instead of HCA. The advantage would be the former dictates the number of modules, where the latter does not.

6.2.3 Automatic clustering of FS-DSM

Paper B discusses a hybrid method called FS-DSM (Blackenfelt 2000). The clustering algorithm presented is expressed as a three-step heuristic which builds on work by (Newcomb, Bras & Rosen 1996; Coulter, McIntosh, Bras & Rosen 1998; Gu & Sosale 1999; Kusiak & Chow 1987). It is conceivable that someone could transform the proposed three-step heuristic into a computerized algorithm, but doing so would be a significant programming task. During a brief phone interview in 2011 about the algorithm, the author asked Dr. Blackenfelt whether he ever devised a program to perform the three-step clustering. Dr. Blackenfelt stated that to the best of his memory, all the clustering was performed by hand, by rearranging rows and columns in an Excel spreadsheet. In the end, we have to conclude that an algorithm is surely possible, but that devising it seems far from trivial. Genetic Algorithms (GAs) allow for more flexibility in terms of their objective functions (Simpson et al. 2011). Therefore, a GA may be more suited than IGTA as a basis. Blackenfelt’s heuristics may be expressed as objective functions. In each round of clustering, the input is taken to be the output of the previous stage. The input to the first stage is the “raw” FS-DSM. In each stage, the objective function is changed to reflect the heuristics in **Error! Reference source not found..**

6.2.4 New Convergence Properties

It is interesting to ask whether Convergence Properties fit into some theoretical framework, like the one by (Hubka & Eder 1996). Three related research topics are discussed below.

First, it may be interesting to look at Functions as the intermediary between Customer Requirements and Technical Solutions. This has been explored by many researchers. (Sellgren and Andersson 2005) use functions in their eISM model. (Suh 1990) uses Functional Requirements as the starting point and then relates those directly to Design Parameters. It may be possible to use Functions and Properties in the same matrix. Clustering is, after all, a statistical process.

Second, we may use the QFD scoring to aid in the clustering. The way to do this would be to create a new matrix calculated as QFD times transpose of DPM, possibly normalized to have values between zero and nine, and then include that, together with DPM/MIM, in the clustering. This would be equivalent to simply adding the Customer Requirements, as some kind of super-functions, and allowing that to influence the Clustering. This matrix would have the same number of columns as there are Customer Requirements in the QFD, and the same number of rows as there are Technical Solutions in the DPM. The most attractive feature of this idea is this scoring is “free” – the scoring has already been done once, in the QFD, and the matrix product can be machine-generated, essentially providing more information at no extra cost (in terms of labor), and potentially improving the output in the process.

Third, the overall nature of the target segments may be included in the QFD and DPM. This was done in a recent project, in 2011, where the concept became known as “Red/Green/Blue” properties. The product was an upright corded vacuum cleaner for the US market. It was

determined that users fell into three main segments. They were assigned names and a color code. The team made an attempt to determine which of the segments were mainly supported by each of the Customer Requirements. This information was then added as a “super-property” and allowed to influence the clustering. Here is how that worked, using an example. One of the segments was called “Time Challenged”. The color code assigned in this case was Green. A Customer Requirement that was ranked as very important in this segment was called “Easy to wind cord”. There was an option property called “Presence of automatic cord winder”, and of course a Technical Solution called “Cord winder”. What the team ended up doing was to introduce “Green” as a super-property, and then put a strong scores on “Easy to wind cord” in the QFD and “Cord winder” in the DPM. The same was done for the other two “colors”. The way this influenced the clustering was to drive the modules toward three large “super groups” of features that each respectively supports a group of features. Thus, to the extent that it is possible, Cord winder would get clustered with other features that support “Time Challenged” customers’ needs.

6.2.5 Modularization of the electronic inverter

The evaluation of MFD and DSM applied to a novel machine type, the hybrid Forwarder, identified the electronic inverter as being in need of a more accurate Technical Solution breakdown, which potentially has to support several operating modes. The forthcoming algorithm called R-IGTA could possibly be applied to the system, using the same DPM and DSM, but applying a new algorithm which tries to integrate both matrices into one single output, thereby addressing the second issue described in the previous section. To be useful, however, this work would require access to someone with a detailed knowledge of inverter technology, which is hard to find.

6.2.6 Further computational improvements to IGTA-plus

IGTA-plus could be improved even further in at least two ways. First, the iterative loops used to calculate ClusterBid could probably be written as matrix operations. Second, a much better initial configuration could be calculated using a non-stochastic algorithm such as the one by (Li 2011), which is extremely fast – and available freely as Matlab code (Li 2010). One way of obtaining a better initial configuration would be to multiply the DPM by its own transpose, normalize the values thus obtained, add it to the DSM, and then apply the algorithm by Simon Li (Li 2011; Li 2010) to the resulting matrix.

6.2.7 Heuristics to improve IGTA

It may be possible to devise heuristics that make IGTA-plus more efficient, too. Automatic generation of heuristics is explored in (Prieditis 1993). However, the type of heuristics thus generated seems mainly applicable to *planning problems*, e.g., the objective is to find a sequence of steps that transform the initial configuration to a desired and known end-state. In MFD or DSM type clustering, the end state is not known, but it may still be possible to devise heuristics that make the search faster, such as moving components that have a large impact on TotalCost more frequently than components with a low impact.

6.2.8 GA-core for IGTA

The two objective functions used in IGTA-plus, TotalCost and ClusterBid, could be used in a Genetic Algorithm. It would be an additional challenge to write a GA that executes as quickly as IGTA-plus.

6.2.9 Expand the problem domain for IGTA

The scope of IGTA-plus could be extended to encompass MFD, to strike a balance between the module requirements in the DSM and the DPM/MIM. This is explored in a forthcoming

paper by the author, co-authored with Professor Katja Hölttä-Otto (Assistant Professor of Mechanical Engineering, University of Massachusetts Dartmouth). The algorithm is tentatively named R-IGTA, where the R represents Reangularity (Suh 1990).

7. REFERENCES

- Akao, Y., Mizuno, S., 1994, The Customer-Driven Approach to Quality Planning and Development, Asian Productivity Organization, Tokyo.
- Alexander, C., 1964, Notes on the synthesis of form, Harvard Press, Boston, ISBN 978-0674627512 .
- Andersson, F., 2003, The Dynamics of Requirements and Product Concept Management, Dissertation, Chalmers University of Technology, Gothenburg, Sweden. ISBN 91-7291-383-5.
- Blackenfelt, M., 2000, Modularisation by relational matrices - a method for the consideration of strategic and functional aspects, Proceedings of the 5th WDK Workshop on Product Structuring, Tampere, Finland, February 7-8.
- Borjesson, F., 2009, Improved Output in Modular Function Deployment Using Heuristics, Proc. of the 17th International Conference on Engineering Design (ICED '09), Vol. 4, ISBN 9-781904-670087, Stanford, 2009, pp 1-12.
- Borjesson, F., 2010, A Systematic Qualitative Comparison of Five Approaches to Modularity, 11th International Design Conference, Design 2010, Dubrovnik, Croatia, May 17-20.
- Browning, T. R., 2001, Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions, IEEE Transactions on Engineering Management, 48(3):292-306, DOI 10.1109/17.946528, ISSN 0018-9391.
- Chiriac, N., Hölttä-Otto, K., Suh, E.S., Lysy, D., 2011, Level of Modularity and Different Levels of System Granularity, ASME Journal of Mechanical Design (accepted for publication).
- Clarkson, J. P., Simons, C., Eckert, C., 2004, Predicting Change Propagation in Complex Design, Journal of Mechanical Design, 126(5):788-798, DOI 10.1115/1.1765117, ISSN 1050-0472 (printed), 1528-9001 (online).
- Dyson, J., 1986, Vacuum cleaning appliance, Patent number 4593429, United States Patent Office.
- Eppinger, S. D., Whitney, D. E., Smith, R. P., Gebala, D. A., 1994, A model-based method for organizing tasks in product development, Res. Eng. Design 6:1-13, DOI 10.1007/BF01588087, ISSN 0934-9839.
- Ericsson, A., Erixon, G., 1999, Controlling Design Variants: Modular Product Platforms, ASME Press, The American Society of Mechanical Engineers, New York, NY. ISBN 0-87263-514-7.
- Erixon, G., 1998, Modular Function Deployment - A method for Product Modularisation, PhD thesis, The Royal Institute of Technology, Stockholm.
- Everitt, B. S., Landau, S., Leese, M., Stahl, D., 2011, Cluster Analysis, 5th edition, Wiley, West Sussex, ISBN 978-0470749913.
- Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C., Clarkson, P. J., 2009, Change propagation analysis in complex technical systems, Journal of Mechanical Design 131(8), 081001, DOI 10.1115/1.3149847, ISSN 1050-0472 (print), 1528-9001 (online).

- Gutierrez Fernandez, C. I., 1998, Integration Analysis of Product Architecture to Support Effective Team Co-Location, Master's Thesis at Massachusetts Institute of Technology, Department of Mechanical Engineering.
- Hauser, J. R., Clausing, D., 1988, The House of Quality, The Harvard Business Review, May-June, No. 3, pp. 63-73.
- Helmer, R, Yassine, A., Meier, C., 2010, Systematic module and interface definition using component design structure matrix, J. Eng. Des. 21(6):647-675, DOI 10.1080/09544820802563226, ISSN 0954-4828 print / 1466-1837 online.
- Hölttä, K, Tang, V., Seering, W. P., 2003, Modularizing product architectures using dendrograms, Proceedings of International Conference on Engineering Design (ICED'03), August 19-21, Stockholm, Sweden.
- Hölttä, K., 2005, Modular Product Platform Design, Doctoral Dissertation, Helsinki University of Technology, Department of Mechanical Engineering, Machine Design, Helsinki, Finland. ISBN 951-22-7766-2.
- Hölttä, K., Otto, K. N., 2005, Incorporating design effort complexity measures in product architectural design and assessment, Design Studies 26(5):463-485, DOI 10.1016/j.destud.2004.10.001, ISSN 0142-694X.
- Hölttä-Otto, K., Tang, V., Kevin, O., 2008, Analyzing module commonality for platform design using dendrograms, Res. Eng. Design 19:127-141, DOI 10.1007/s00163-008-0044-3, ISSN 0934-9839.
- Idicula, J., 1995, Planning for Concurrent Engineering, Gintic Institute Research Report, Singapore.
- Lehtonen, T., 2007, Designing Modular Product Architecture in the New Product Development, Doctoral Dissertation, Tampere University of Technology, Tampere, Finland, Publication 713. ISBN 978-952-15-1898-0, ISSN 1459-2045.
- Li, S., 2010, Matlab program code for two-mode clustering, downloaded Feb 24, 2012, available at <http://users.encs.concordia.ca/~lisimon/software.html>.
- Li, S., 2011, A matrix-based clustering approach for the decomposition of design problems, Res. Eng. Design 22:263-278, DOI 10.1007/s00163-011-0111-z, ISSN 0934-9839.
- Martin, M. V., Ishii, K., 2002, Design for variety: developing standardized and modularized product platform architectures, Res. Eng. Design 13:213-235, DOI 10.1007/s00163-002-0020-2, ISSN 0934-9839.
- Mathworks, The, Inc., 2010, Matlab R2010a, <http://www.mathworks.com>, .
- Nilsson, P., Erixon, G., 1998, The Chart of Modular Function Deployment, Proceedings of 4th Workshop on Product Structuring, Delft University of Technology, Delft, The Netherlands.
- Pimmler, T., Eppinger, S., 1994, Integration Analysis of Product Decomposition, Minneapolis: ASME Design Engineering Technical Conferences-6th International Conference on Design .
- Russell, S., Norvig, P., 2002, Artificial Intelligence: A Modern Approach (2nd Edition), Prentice Hall, Englewood Cliffs, ISBN 978-0137903955.

- Sellgren, U., Andersson, S., 2005, The Concept of Functional Surfaces as Carriers of Interactive Properties, Proceedings of International Conference on Engineering Design (ICED'05), August 15-18, Melbourne, Australia.
- Sharman, D. M., 2002, Valuing Architecture For Strategic Purposes, Master's Thesis at Massachusetts Institute of Technology, System Design & Management Program.
- Sharman, D. M., Yassine, A. A., 2007, Architectural Valuation using the Design Structure Matrix and Real Options Theory, *Concurrent Engineering* 15(2):157-173, DOI 10.1177/1063293X07079320, ISSN 1063-293X (print), 1531-2003 (online).
- Simpson, T. W., Maier, J. R. A., Mistree, F., 2001, Product Platform Design: Method and Application, *Res. Eng. Design* 13:2-22, DOI 10.1007/s001630100002, ISSN 0934-9839.
- Simpson, T. W., Bobuk, A., Slingerland, L. A., Brennan, S., Logan, D., Reichard, K., 2011, From user requirements to commonality specifications: an integrated approach to product family design, *Res. Eng. Design* (online first, 19 August 2011), DOI 10.1007/s00163-011-0119-4, ISSN 0934-9839.
- Stake, R. B., 2000A, Using cluster analysis to support the generation of modular concepts in the MFD-method, International CIRP Manufacturing Systems Seminar, June 5-7, Stockholm, Sweden.
- Steward, D. T., 1981, The Design Structure System: A Method for Managing the Design of Complex Systems, *IEEE Transactions on Engineering Management*, 28(3):71-74, ISSN 0018-9391.
- Stone, R. B., Wood, K. L., Crawford, R. H., 2000, A heuristic method for identifying modules in product architectures, *Des. Studies*, 21(1):5-31, DOI 10.1016/S0142-694X(99)00003-4, ISSN 0142-694X.
- Suh, N. P., 1990, *The Principles of Design*, Oxford University Press, Inc., New York, NY 10016.
- Suh, E. S., Kott, G., 2010, Reconfigurable Parallel Printing System Design for Field Performance and Service Improvement, *Journal of Mechanical Design*, 132(3), 034505, DOI 10.1115/1.4000961, ISSN 1050-0472 (print), 1528-9001 (online).
- Thebeau, R. E., 2001A, Knowledge Management of System Interfaces and Interactions for Product Development Process, Master's Thesis at Massachusetts Institute of Technology, System Design & Management Program.
- Thebeau, R. E., 2001B, Matlab program code, available at <http://www.dsmweb.org/>.
- Ulrich, K., 1995, The role of product architecture in the manufacturing firm, *Research Policy* 24(3):419-440, DOI 10.1016/0048-7333(94)00775-3, ISSN 0048-7333.
- Whitfield, R. I., Smith, J. S., Duffy, A. H. B., 2002, Identifying Component Modules, Proceedings of the 7th international conference on artificial intelligence in design AID'02, pp. 571-592.
- Wyatt, D., Jarrett, J. P., Clarkson, P. J., 2012, Supporting product architecture design using computational design synthesis with network structure constraints, *Res. Eng. Design* 23:17-52, DOI 10.1007/s00163-011-0112-y, ISSN 0934-9839.
- Wynn, D. C., Nair, S. M., Clarkson, P. J., 2009, The P3 platform: an approach and software system for developing diagrammatic model-based methods in design research. In: Norell Bergendahl, M., Grimheden, M., Leifer, L., Skogstad, P., Lindemann, U. (eds) *Proceedings*

of the international conference on engineering design (ICED '09), Palo Alto, California, USA, pp. 559-570.

Yu, T.-L., Yassine, A. A, Goldberg, D. E., 2007, An information theoretic method for developing modular architectures using genetic algorithms, Res. Eng. Design 18:91-109, DOI 10.1007/s00163-007-0030-1, ISSN 0934-9839.

Zamirowski, E. J., Otto, K. N., 1999, Identifying Product Portfolio Architecture Modularity Using Function and Variety Heuristics, Proceedings of the 1999 ASME Design Engineering Technical Conferences (DETC99/DTM-8760), Las Vegas.

IMPROVED OUTPUT IN MODULAR FUNCTION DEPLOYMENT USING HEURISTICS

Fredrik Borjesson

Royal Institute of Technology (KTH), Department of Machine Design,
Stockholm, Sweden

ABSTRACT

In Modular Function Deployment, technical solutions are grouped into modules according to the product properties and the strategic intentions of the company. Statistical methods such as hierarchical clustering are useful in the formation of potential modules, but a significant amount of manual adjustment and application of engineering common sense is generally necessary. We propose a method for promoting better output from the clustering algorithm used in the conceptual module generation phase by adding Convergence Properties, a collective reference to data identified as option properties, geometrical information, flow heuristics, and module driver compatibility. The method was tested in a case study based on a cordless handheld vacuum cleaner.

Keywords: Conceptual product development, modular products, Modular Function Deployment, module drivers, clustering algorithm, hierarchical clustering, statistical approach, heuristic methods

0 ABBREVIATIONS

Table 1 is a summary of the abbreviations used, with a brief description.

Table 1. Abbreviations, existing and proposed

Existing accepted terminology			
Data		Matrices	
Abbreviation	Meaning	Abbreviation	Meaning
CR	Customer Requirement. Expression of customer need.	QFD	Quality Function Deployment. Defines the relation between CR and PP.
PP	Product Property. Measurable and controllable translation of CR.	DPM	Design Property Matrix. Defines the relation between PP and TS.
TS	Technical Solution. Physical carrier of a required function.	MIM	Module Indication Matrix. Defines the relation between TS and MD.
MD	Module Driver. Describes the company-specific strategy.	PMM	Product Management Map. Interlinked visualization of QFD, DPM, and MIM.

Proposed new terminology			
Data		Matrices	
Abbreviation	Meaning	Abbreviation	Meaning
OP	Option Property. Presence of specific technical features.	QED	Extended QFD. Defines the relation between CR and OP.
GP	Geometrical Property. Representation of key regions in product.	CPM	Convergence Property Matrix. Defines relation between CP and TS.
HP	Heuristic Property. Application of branching-combining, conversion-transmission, and dominant flow heuristics.	ePMM	Extended PMM. Standard PMM plus QED and CPM.
DC	Driver Compatibility. MD compatibility to guarantee strategically compatible clusters.		
CP	Convergence Properties. Collective reference to OP, GP, HP, and DC.		

1 INTRODUCTION

Modularity methods are concerned with the translation of customer requirements and company strategy into a product architecture that offers reduced time-to-market (by allowing flexible configurations), lower unique part number count and often material cost reduction (through reduction of suppliers and improved purchasing leverage). According to Hölttä-Otto [1], there are three main approaches to modularity.

Heuristics refer to rules of thumb that very often give good results. In [1], two main categories are investigated: modules dictated by the patterns of flow (matter, energy, and information) between the functional blocks, and patterns of commonality/variety in a family of products. These methods are highly repeatable [1] but do not consider strategic objectives [2].

DSM [3] may be used to determine the ideal sequence of development tasks in a project, but it can also be used to define modules in a product architecture [1]. The best sequence is one

that minimizes the number of coupled tasks. DSM does not consider strategic objectives or even functional requirements of the product.

MFD [4] is a five-step method for translating customer requirements into a modular architecture, while considering the strategic objectives (described using twelve predefined Module Drivers). Cross-functional teams are used. Project data is captured in three core matrices. MFD allows for a high level of concurrency in the conceptual phase, before modules are defined. In this paper, a module is defined as a functional block with standardized interfaces, selected for company-specific reasons [4].

Module generation is based on grouping Technical Solutions into modules with related functions and similar strategic intent. In real projects, this involves sorting a large amount of data, and for practical reasons this must be done using statistical methods. The output is often shown as a Dendrogram, a hierarchical representation of the level of similarity between the Technical Solutions. Very often, however, the first Dendrogram just does not seem fully to make sense.

This paper is part of a larger research topic, with the goal of determining how we can improve MFD to yield better output. A better understanding of the product properties that drive product architecture decisions is believed to be at the heart of this question. This paper deals with a more narrowly defined topic: the introduction of so called *Convergence Properties* to yield more useful conceptual module output from the statistical algorithms.

2 BRIEF MFD THEORY

Without a solid understanding of Customer Requirements, any product architecture effort risks becoming an engineering-driven exercise without useful market application. QFD [5] is a powerful tool for describing Customer Requirements in terms of Product Properties.

Shortly after MFD [4] was introduced, it was improved by the addition of the Design Property Matrix (DPM) [6], linking the Product Properties in the QFD with the Technical Solutions of the Module Indication Matrix (MIM). This version of MFD is the reference for comparisons with proposed improvements.

Module Drivers are used to describe the strategic intent of an architecture, and is a key feature of the MFD approach. The Drivers are summarized in Table 2 below.

Table 2. Module Drivers used in MFD [4]

Driver	Technical consideration
Common Unit	Allow solutions to be used in several variants
Carry Over	Allow solutions to be used in future product generations
Technical Specification	Create a range of modules with regard to specification level
Styling	Create a range of modules with regard to styling variation
Planned Design Changes	Allow new design to be incorporated
Technology Push	Allow new technology to be incorporated
Process / Organization	Protect scarce resources in development or production
Strategic Supplier Available	Outsource development and manufacturing to external partner
Separate Testability	Allow module level testing before final assembly
Service / Maintenance	Allow easy replacement or service of parts
Upgrading	Allow customer to upgrade performance after purchase
Recycling	Extract dangerous or valuable materials at time of scrapping

Module Drivers support basic company strategy. Drivers that support the same strategy are said to be compatible [4]. A module should consist of technical solutions with compatible Module Drivers only. Empirical research [7] indicates the interpretation of driver compatibility varies somewhat from one company to another. However, Figure 1 provides a useful guideline. Conflicting driver combinations are indicated by a minus sign while mutually supporting drivers get a plus sign. A question mark means potentially there is interaction, but the nature must be determined on a case-by-case basis.

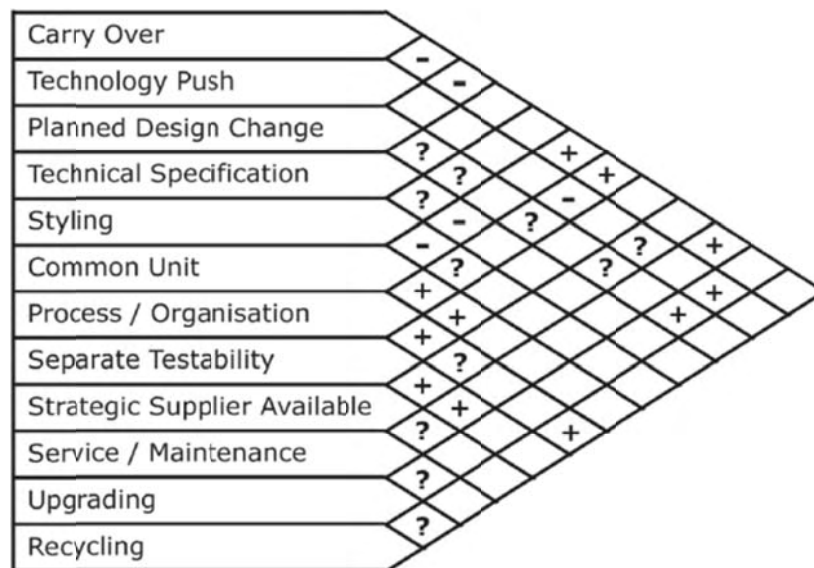


Figure 1. Module Driver compatibility according to Erixon [4]

Some of the strengths of MFD are:

- Allows high degree of parallelism in the conceptual phase of the work
- Use of matrices for all project data implies automated statistical approaches to module generation may be used, which is particularly useful in large projects
- Incorporates customer driven, engineering driven, and strategic considerations

Some of the weaknesses of MFD are:

- Results depend on the experience and technical expertise of the team conducting the work [1]
- Quality is highly dependent on good property definitions and consistent scoring, which typically requires experience
- For certain product types, there is not always a sufficient number of customer-driven product properties to generate modules on the right level of resolution

3 CONVERGENCE PROPERTIES

The reason why hierarchical clustering often does not converge on a useful first output is problems with the data, not the algorithm. When engineers apply their common sense to determine what constitutes a useful output, they rely on additional information which typically is not part of the data supplied to the algorithm.

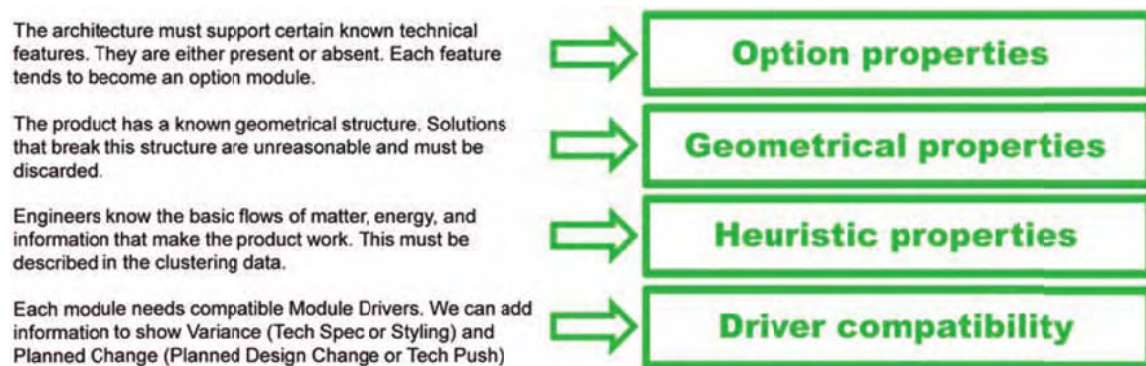


Figure 2. *Convergence Properties are a response to common practical issues in MFD*

We aim to promote better output in the module-clustering phase of MFD by considering several practical concerns and including data pertaining to the four areas named in Figure 2. The concept of *Convergence Properties* is introduced to formalize some of the common sense applied in practical applications of MFD. In this paper, we will look at four types of Convergence Properties.

Option properties determine whether a certain product option is enabled in the final product configuration. These are special Boolean properties, e.g., take yes/no values only. Traditional Product Properties used in MFD are solution-free. Option properties are radically different in that they normally stipulate a specific solution.

Geometrical properties reflect additional knowledge about the probable physical configuration of the final product. We may imagine dividing the product into pre-defined

regions, and then labeling each technical solution by whether it would likely be located in a certain region.

Heuristic Properties are based on function structure. Two sets of heuristics are presented in [8] and [9]. The former are based on flow of matter, energy, and information, whereas the latter are based on function and variety (Causally-linked functions, Similarity/repetition, and Commonality/variety). We will be using the flow heuristics in this work, but not the function and variety heuristics, since the latter overlap more strongly with what may be achieved using the normal Product Properties and Module Drivers of traditional MFD. Flow heuristics describe how functions interact in the product, which complements the performance-based properties of MFD. Each of the three flow heuristics are presented below. The word Flow refers to a flow of matter, energy, or information.

Dominant flow : If the same Flow goes through a sequence of functions, they should form a module.



Figure 3. Dominant flow heuristic

In product where there is a strong dominant Flow, this Flow typically determines the system performance. Almost always, the technical solutions have to be in a certain sequence. If the performance level has to be changed, all the technical solutions involved in that flow typically have to be re-engineered in harmony. A jet engine might be an intuitive example: stepping up the thrust involves changing inlet, fan blades, compressor, exhaust geometry etc.

Conversion-transmission : Functions that convert one type of Flow into another should form modules. If the conversion is followed by transmission, that should be part of the same module.

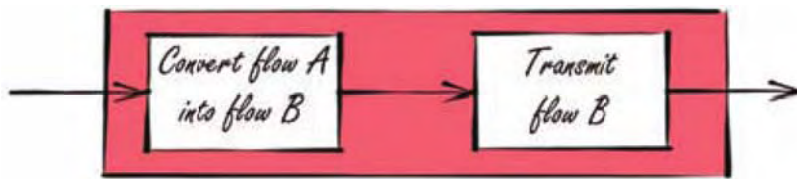


Figure 4. Conversion-transmission heuristic

This heuristic is based on the idea that Flow transformation typically happens in a technical solution that has some critical property related to the conversion itself, and this property tends to be very local to that technical solution. Transmission is included because it is convenient to deliver the output to where it is needed. A good example would be a DC-motor with outgoing shaft.

Branching-combining : If a Flow splits up into parallel function chains, the subfunctions that make up those chains should form modules. This also applies to combining flows.

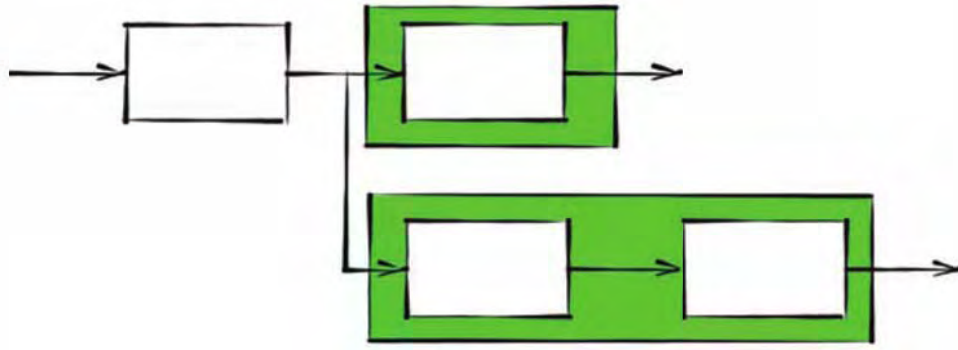


Figure 5. Branching-combining heuristic

Branches dealing with the same Flow often perform independent tasks and often have to be flexibly configurable. To enable that, it is useful to use the same interface in each branch. This results in a bus architecture, as in a PC where boards can be added flexibly.

Driver Compatibility may be used to define groups of Module Drivers in such a way that drivers within a group support the same strategy. Very often in MFD project work, the DPM is much larger than the MIM. Since the clustering algorithm is essentially statistical, the strategic intent reflected in the MIM is drowned out by the performance requirements described in the DPM. By supplying additional information about compatibility, we can decrease the risk that Technical Solutions with incompatible drivers get allocated to the same module.

Extended PMM accommodates Convergence Properties

One of the key outputs of MFD is the Product Management Map (PMM), which in turn consists of three matrices, Quality Function Deployment, Design Property Matrix, and Module Indication Matrix. A traditional PMM is shown on the left in Figure 6. An Extended PMM (ePMM) is shown on the right, and it represents a potential solution to the problem of *incorporating Convergence Properties into MFD in a matrix format to allow usage of statistical tools for module generation*. The ePMM has two additional matrices: the Convergence Property Matrix (CPM) and the Extended QFD (abbreviated QED to underscore the analogy to QFD). The QED uses Option Properties that take yes/no values only. The remaining Convergence Properties are not derived from customer needs, so that part of QED is left blank.

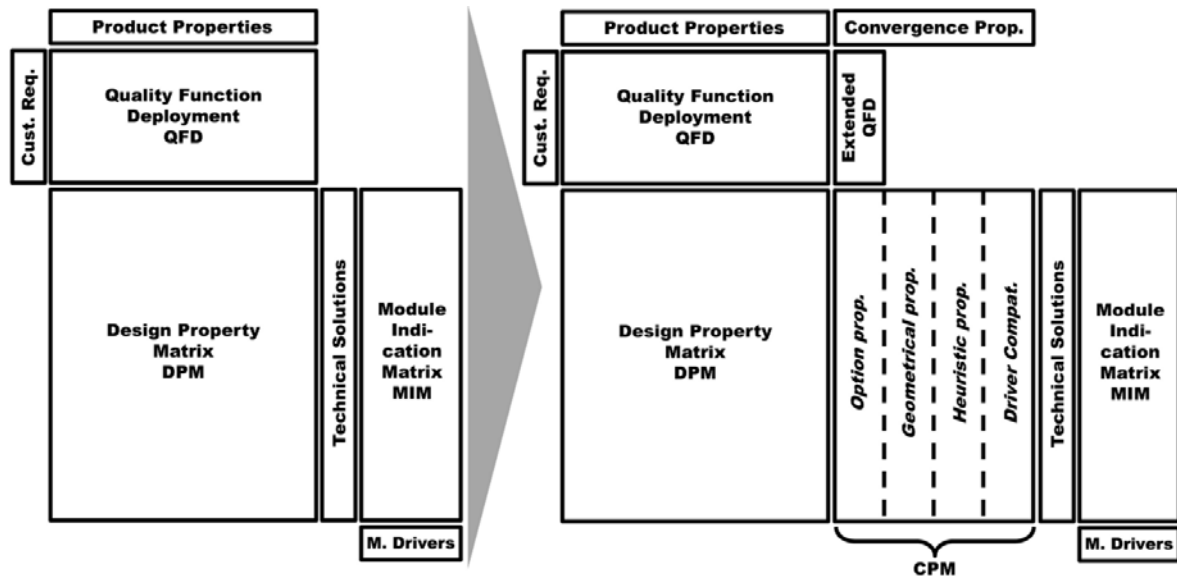


Figure 6. PMM and the Extended PMM (ePMM)

4 CASE STUDY – CORDLESS HANDHELD VACUUM CLEANER

Background

The case study presented here is based on a cordless handheld vacuum cleaner, which has been used successfully in a basic five day MFD-training for hundreds of students. This case study is based on learnings from that training as well as desk research. One key element of the training is the scoring of the DPM and the subsequent application of clustering on that data, which is shown schematically in Figure 7.

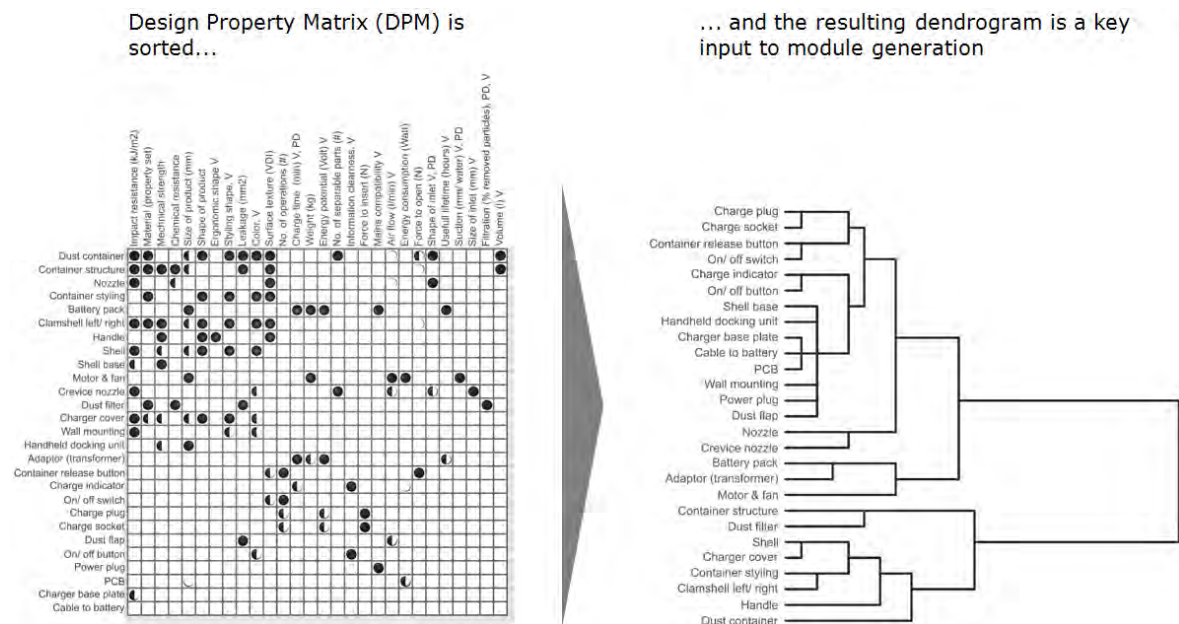


Figure 7. Schematic of the transformation of DPM into a Dendrogram

Figure 8 shows how a team might interpret the Dendrogram in Figure 7. Potential modules are indicated with boxes.

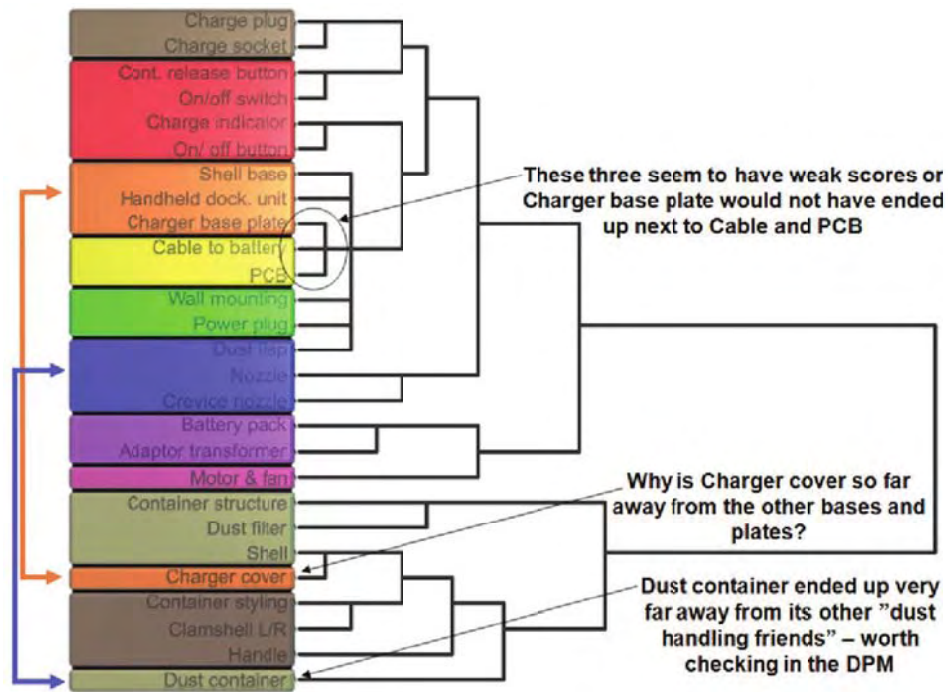


Figure 8. Possible interpretation of the dendrogram in Figure 7

Project experience has shown that dendrograms rarely can be used as is. In the example above, there are several irregularities. This raises the suspicion that our example DPM contains incorrect scoring or even poorly defined properties.

Why dendrograms must be interpreted

Table 3 shows why dendrograms often have to be interpreted, and not used as is.

Table 3. Examples of reasons why dendrograms need to be interpreted

Reason	Example
Geometrical information about the product is not considered	All product styling elements, regardless of physical location, become one single module
Critical properties are overlooked	Two different fuel sources, electricity and gas, are treated the same, resulting in impossible dual-fuel modules
Engineering properties cloud the DPM matrix	Too much weight is placed on surface and material properties, resulting in a skewed QFD and not enough focus on interface-driving properties
Required sequence of certain technical solutions is ignored	Nozzle and exhaust create a module, ignoring the need for an impeller to drive the flow
Modules are not strategy clean	Size of DPM overshadows MIM completely, resulting in a clustering that is almost completely DPM-based
Important requirements are not translated into useful properties	“Easy to clean” is translated into a statement such as “cleanability”, which on the surface may look like a property, but in reality is nothing but a reworded requirement
Key concept selections are treated as any other property	The method used to release a dust container from the handheld unit is captured by a property-like statements such as “Number of steps to remove container”
Tremendous energy is expended on engineering properties	Color, finish, and material are described in tremendous detail, when normally surface properties do not drive an interface in the architecture
System properties are never disaggregated	“Suction power” is introduced, resulting in modules that are simply much too large, vaguely corresponding to subsystems

To show how the proposed four types of Convergence Properties might help module creation in the handheld vacuum cleaner example, we will now look at an example of each one.

Option Properties

When we devise a new architecture for a family of cordless vacuum cleaners, we may decide to support an optional charge in progress indicator, as shown in Figure 9. We may capture this by creating an Option Property called “Presence of Charge Feedback”.

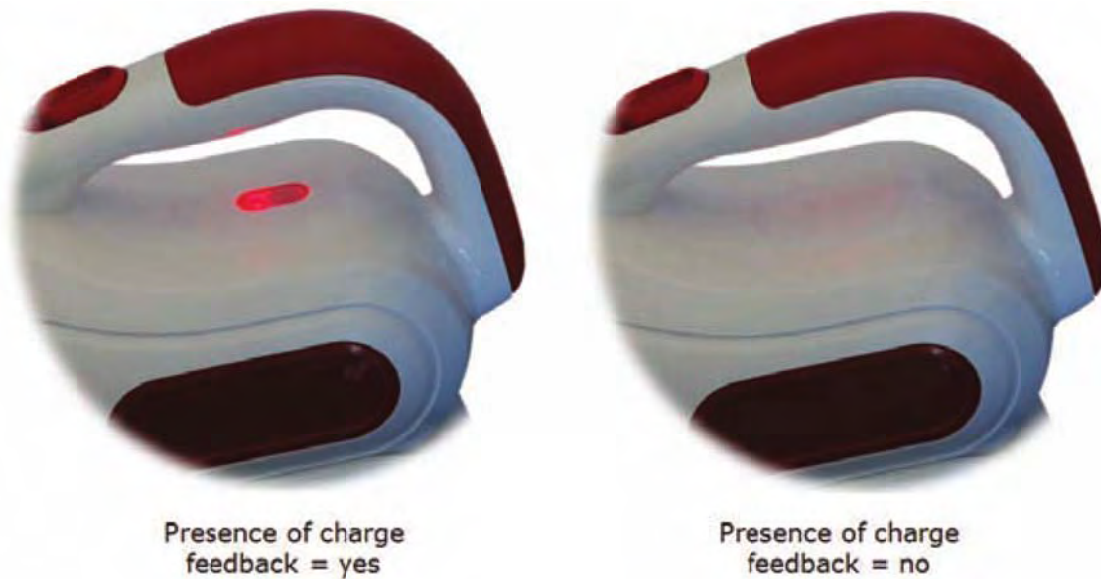


Figure 9. Charge feedback is either present or it is not. Photo manipulated by author.

Geometrical Properties

Without knowledge of product geometry, hierarchical clustering could propose a module composed of suction nozzle and air exhaust, for example. They both relate to pressure drop and are both characterized by an important cross section. Such a module would ignore the normal physical configuration of the product. In terms of geometry, we know the nozzle is typically in the front, and the exhaust typically in the back. Figure 10 shows three regions: Front, Rear, and Off unit.

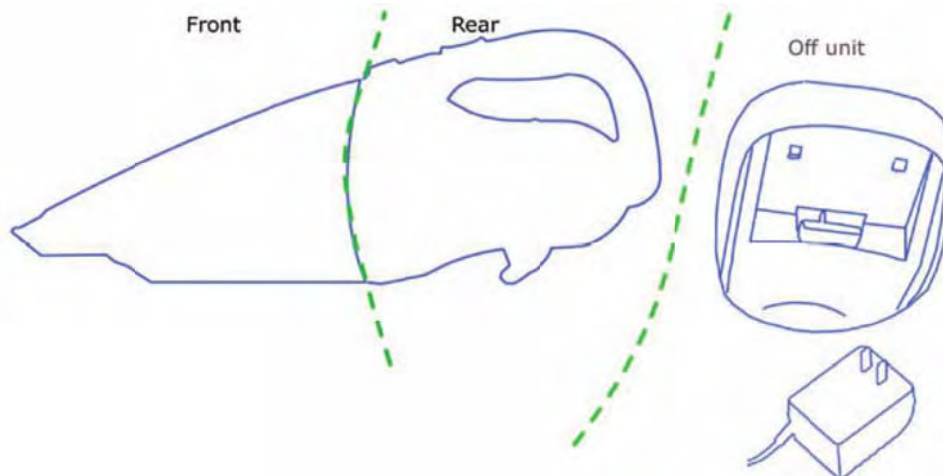


Figure 10. Geometrical properties describe the region where a TS belongs

Heuristic Properties

The dominant flow heuristic is particularly suitable in a handheld vacuum cleaner, where there are flows of air and energy. This heuristic predicts that any technical solution involved

in the main airflow should be part of one and the same module. We create a Heuristic Property called Dominant Air Flow, and tag all the involved technical solutions in the CPM. This holds them together in the clustering.

Driver Compatibility

In this example, two groups were introduced, Variance and Planned Change, defined as follows

- Variance: enforce distinction between Common Unit and either Styling or Different Specification
- Planned Change: enforce distinction between Carry-over and either Planned Design Change or Technical Evolution

If a Technical Solution had a MIM score on Styling or Different Specification, it would also receive a score on Variance. Similarly, Planned Design Change or Technical Evolution would give it a score on Planned Change.

Putting CPM to the test

We compared a traditional PMM with an ePMM, and examined the Dendrogram output. The Product Properties and Convergence Properties that were used are summarized in Table 4. The Module Drivers were used as well and they can be found in Table 2.

Table 4. Variables used in analysis

In MFD	Type	Name	Meaning
DPM	Product Property	Scratch resistance	High for outdoor applications
		Container volume	Dust storage volume
		Battery Capacity (mAh)	Battery capacity
		Particle size	Largest particle through filter
		Nozzle Angle	Pointiness of nozzle
		Output filter (mm^2)	Exhaust diffuser area
		Resolution of speed control	Type of movement of slider
		Average Charge Current	Charge current to batteries
		Voltage	Total voltage of batteries
		Primary voltage	Voltage from wall socket
		Blade height	Of impeller, impacts efficiency
		Impeller Inner Diameter	Drives flow through impeller
		Impeller Outer Diameter	Drives pressure from impeller
CPM	Option Property	Washable filter (y/n)	Whether filter is washable
		Wet application (y/n)	Whether unit has wet capability
		Charge Completed Indicator (y/n)	Indicates charge completed
		Charge in Progress Indicator (y/n)	Indicates charge in progress
		Charge Management option (y/n)	Has intelligent charge controller
		Powered attachment option (y/n)	Connects external rotating brush
		Adapter storage in base (y/n)	Stores adapters in base of unit
		Noise absorber option (y/n)	Has noise absorber on impeller
		Slider switch (y/n)	Slider switch for speed control
		Clean dial (y/n)	Filter agitator to unclog filter
		Presence of Electronic Speed Control (y/n)	Speed control is electronic
		Detachable battery option (y/n)	Connects external battery pack
	Geometrical Property	Brand ID	To drive shell and handle into one
		Off unit	Anything that is not on the unit
		Front	Located in the front
		Rear	Located in the rear
	Heuristic Property	Conversion - electrical to rotational	Hits motor and axle
		Conversion - HVAC to LVAC	Hits transformer, terminals and leads
		Conversion - pressure to flow	Hits nozzle or adapter
		Dominant - hold unit	Flow of holding force
		Dominant - air flow	Flow of air through unit
		Dominant - clean filter	Flow of dust when cleaning filter
		Dominant - liquid flow	Flow of liquid for wet & dry
		Dominant - attachment power	Flow of power to attachment
		Dominant - provision electricity	Flow of power from battery to motor
	Driver Compatibility	Variance	If Technical Spec or Styling was used
		Planned change (int or ext)	If Plan Des Change or Tech Push was used

Clustering was applied to three combinations of data as shown in Table 5. More tickmarks means more information was used. Full CPM uses all the Convergence Properties. No CPM uses none, which corresponds to clustering on a normal PMM. A hybrid scenario labeled Partial CPM was introduced to show how Option Properties alone also improve Dendrogram output.

Table 5. Clustering analysis was applied to three sets of data

Scenario	Data used					
	Prod.	Option	Geom.	Heur.	Driver	Module
	Prop.	Prop.	Prop.	Prop.	Compat.	Drivers
Full CPM	✓	✓	✓	✓	✓	✓
Partial CPM	✓	✓				✓
No CPM	✓					✓

The appearance of flat subtrees in Partial CPM and No CPM signifies lack of information, and this can clearly be seen in Figure 11. It is not necessary to read the Technical Solutions to see the subtrees.

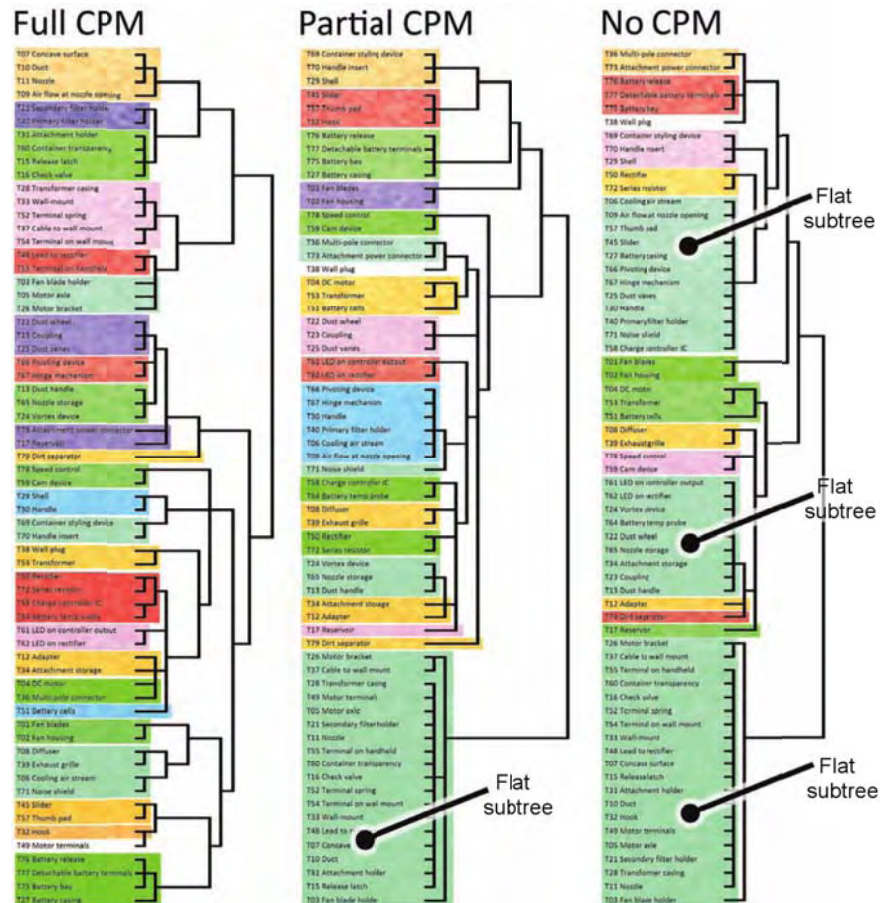


Figure 11. Dendrogram output from the three test scenarios in Table 5

Large flat subtrees *could* be module clusters, but large flat subtrees of *unrelated* technical solutions are caused by lack of relevant scoring. The Dendrogram labeled Full CPM does not show this symptom. The technical solutions of each flat subtree are listed in Table 6. The

reader will be able to confirm these technical solutions indeed are not related and do not represent useful modules.

Table 6. Unrelated technical solutions in large flat subtrees

<p>Full CPM</p> <p><i>No flat subtrees of unrelated technical solutions</i></p>
<p>Partial CPM</p> <p>Flat subtree 1</p> <p>Attachment holder, Cable to wall mount, Check valve, Concave surface, Container transparency, Duct, Fan blade holder, Lead to rectifier, Motor axle, Motor bracket, Motor terminals, Nozzle, Release latch, Secondary filter holder, Terminal on handheld, Terminal on wall mount, Terminal spring, Transformer casing, Wall-mount</p>
<p>No CPM</p> <p>Flat subtree 1</p> <p>Cooling air stream, Air flow at nozzle opening, Thumb pad, Slider, Battery casing, Pivoting device, Hinge mechanism, Dust vanes, Handle, Primary filter holder, Noise shield, Charge controller IC</p> <p>Flat subtree 2</p> <p>LED on controller output, LED on rectifier, Vortex device, Battery temp probe, Dust wheel, Nozzle storage, Attachment storage, Coupling, Dust handle</p> <p>Flat subtree 3</p> <p>Hook, plus all the technical solutions in Flat subtree 1 from scenario Partial CPM</p>

Observations from using ePMM

- The dendrograms obtained from two of the scenarios feature large flat subtrees of unrelated technical solutions, indicating these clusters are the result of insufficient information.
- The result obtained in scenario Full CPM is drastically better than from No CPM. The former corresponds to an ePMM and the latter a normal PMM without any Convergence Properties.
- With a couple of minor exceptions, the clusters made intuitive sense. This result was far better than from a typical first output of a normal PMM.
- Both the Dominant flow and Conversion-transmission heuristics were helpful, but in this particular product, the Branching-combining heuristic was not.

5 DISCUSSION

Convergence Properties in the context of other research

The proposed method presented in this paper is an attempt to improve one existing method for generation of modular architectures by including useful features of other methods while preserving the strengths of the original method upon which it is built. This approach has been used by other researchers, such as:

- Blackenfelt [2] who proposes improvements to DSM by incorporating both functional and strategic considerations
- Sellgren and Andersson [10] who incorporate interactive functions into the DSM, using a format similar to the PMM, but where the MIM is replaced by a DSM and functions take the role of properties

Conclusions

Module output varies greatly with the quality of the information provided to the clustering algorithm. The case study shows how Convergence Properties may be added to MFD in such a way that a matrix-based representation may still be used, which keeps one very important original feature intact: the possibility to apply MFD in very large projects where, for practical reasons, manual module generation simply is not possible. We have seen how the addition of four proposed new property types may raise the quality of the first output. The theory was tested on a product of low-to-medium complexity with about 60 technical solutions (a handheld vacuum cleaner), and yielded promising results.

Further research

More research and practical application is required to conclude whether the proposed modifications to MFD consistently improve output, in particular on more complex products with more interfaces or high innovation content. New types of Convergence Properties may be required in some cases. For example, how can Industrial Design and Manufacturing considerations be included? New types of heuristics may be required in products where no strong flows are present. That may be the case in modular storage systems (bookshelves, for example). Is there an underlying Theory of Convergence Properties, such as the Theory of Properties proposed in [11]?

References

- [1] Hölttä-Otto, K., 2005, Modular Product Platform Design, Doctoral Dissertation, Helsinki University of Technology, Department of Mechanical Engineering, Machine Design, Helsinki, Finland. ISBN 951-22-7766-2.
- [2] Blackenfelt, M., 2001, Managing complexity by product modularisation, TRITA-MMK 2001:1, ISSN 1400-1179, ISRN KTH/MMK/R--01/1--SE, Doctoral Thesis, Department of Machine Design, Royal Institute of Technology, Stockholm, Sweden.
- [3] Ulrich, K. T., Eppinger, S. D., 2008, Product Design and Development, fourth edition, McGraw-Hill Education Asia, Singapore.
- [4] Ericsson, A., Erixon, G., 1999, Controlling Design Variants: Modular Product Platforms, ASME Press, The American Society of Mechanical Engineers, New York, NY. ISBN 0-87263-514-7.
- [5] Akao, Y., Mizuno, S., 1994, The Customer-Driven Approach to Quality Planning and Development, Asian Productivity Organization, Tokyo.
- [6] Nilsson, P., Erixon, G., 1998, The Chart of Modular Function Deployment, Proceedings of 4th Workshop on Product Structuring, Delft University of Technology, Delft, The Netherlands.
- [7] Stake, R., 2000, On Conceptual Development of Modular Products, Doctoral Thesis, Department of Production Engineering, Royal Institute of Technology, Stockholm, Sweden.
- [8] Stone, R. B., Wood, K. L., Crawford, R. H., 2000, A heuristic method for identifying modules in product architectures, Des. Studies , 21 (1), 5-31.
- [9] Zamirowski, E. J., Otto, K. N., 1999, Identifying Product Portfolio Architecture Modularity Using Function and Variety Heuristics, Proceedings of the 1999 ASME Design Engineering Technical Conferences (DETC99/DTM-8760), Las Vegas.
- [10] Sellgren, U., Andersson, S., 2005, The Concept of Functional Surfaces as Carriers of Interactive Properties, Proceedings of International Conference on Engineering Design (ICED'05), August 15-18, Melbourne, Australia.
- [11] Hubka, V., Eder, W. E., 1996, Design Science, Springer-Verlag London Limited.

Contact: Fredrik Börjesson, Department of Machine Design, Brinellvägen 85, Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden

E-mail fborjesson0524@hotmail.com

A SYSTEMATIC QUALITATIVE COMPARISON OF FIVE APPROACHES TO MODULARITY

F. Borjesson

Keywords: Modular Function Deployment, Design Structure Matrix, Heuristics, Design for X, qualitative comparison

1. Introduction

The need to compare alternative approaches to modularity in a systematic way has arisen from the research idea that new hybrid approaches may be created to improve the main approaches on which they build. The purpose of the present paper is to answer the following two questions: (1) How may we compare the approaches in a consistent manner? (2) Do derived approaches improve on the main approaches they build on? In addition, we will discuss which of the main approaches seems most promising as a basis for new derived approaches.

Ulrich and Eppinger [Ulrich, Eppinger 2008] start with a simple idea, the chunk. Chunks are physical building blocks. Modular architecture has the following properties: (a) chunks implement one or a few functional elements in their entirety, and (b) the interactions between chunks are well defined and are generally fundamental to the primary functions of the product. We add one more point from [Erixon 1998]: A module is a physical building block with standardized interfaces selected for company-specific reasons. In the remainder of this paper, the phrase “approach to modularity” will be used to mean methods by which modular architectures are defined. We will look at five such approaches. According to [Höltkä-Otto 2005] there are three main approaches to modularity: (1) Heuristics, (2) Design Structure Matrix (DSM), and (3) Modular Function Deployment (MFD). In addition to these, we will look at two hybrid approaches: (4) Functional-Strategic DSM [Blackenfelt 2001], and (5) Extended Implementation Structure Matrix [Sellgren, Andersson 2005].

2. Overview of Methods

For the purpose of providing a brief overview, we will focus on five critical aspects of each method: how data is organized, data types that can be represented, relationships captured, type of interactions or dependencies, and how modules are generated.

Organization of data is particularly important in large projects and for selecting the right type of computer software.

Data type refers to one or several of the following: Technical Solutions (abbrev. TS; to describe how functions are realized), Customer Requirements (abbrev. CR; to describe the benefits customers are looking for in the product), Product Properties (abbrev. PP; to be able to make quantitative statements about the performance level of certain functions), Module Drivers (abbrev. MD; to describe the company specific strategy), Functional Requirements (abbrev. FR; statements about functions that must be performed by the product, used primarily in synthesis), and Functions (abbrev. FU; transformation of an input into an output, often expressed as verb plus noun, used primarily in analysis).

Relation is a reference to comparisons between pairs of data types. For example, QFD is a relation between Customer Requirements and Product Properties.

Interaction is the aspect of the Relation we are interested in. In a Task-based DSM, for example, the interaction is related to Design sequence, e.g., which Technical Solution of the two would be designed first. Other Interactions are Degree of causality (to what extent does changing one drive change in the other), Spatial (should Technical Solutions be close or far apart in the final product), and Flow (exchange of Energy, Matter, or Information).

Module generation refers to definition of groups of TSs into Modules. The Methods take three approaches to Module generation: Hierarchical Clustering creates groups where the TSs have similar PPs and MDs, Least interaction minimizes the dependency between groups of TSs, and Rule-based looks for certain patterns of Flow.

		Heur.	DSM	eISM	MFD	FS-DSM
Organi- zation of data	Matrix		✓	✓	✓	✓
	Func. Struct. diagram	✓				
Data types	Technical Sol'ns (TS)	✓	✓	✓	✓	✓
	Customer Req'ts (CR)				✓	
	Product Properties (PP)				✓	
	Module Drivers (MD)				✓	✓
	Functional Req'ts (FR)			✓		
	Functions (FU)	✓				

(Figure continues on next page)

		Heur.	DSM	eISM	MFD	FS-DSM
Relations	TS-TS		✓	✓		✓
	TS-MD				✓	✓
	TS-PP				✓	
	CR-PP				✓	
	CR-FR			✓		
	TS-FR			✓		
	TS-FU	✓				
	FU-FU	✓				
Inter-action	Spatial		✓	✓		✓
	Degree of causality			✓	✓	
	Design sequence		✓			
	Flow (Energy / Matter / Info.)	✓	✓			✓
Module generation	Hierarchical Clustering				✓	
	Least interaction		✓	✓		
	Rule-based	✓				✓

Figure 1. Overview of all five methods

In Figure 1, a black checkmark means the line item in question is applicable to the particular Method; a shaded checkmark means normally it is not, but with a minor modification it could be. DSM is a reference to Task-based and Component-based DSMs, see section 2.2.

2.1. Heuristics

Within modular architecture, heuristics try to capture how designers actually think. According to [Gilovich, Griffin, Kahneman 2002], heuristics are based on patterns of biased judgments, represent sensible estimation procedures, yield “quick and dirty” solutions, draw on underlying processes that are highly sophisticated, and are normal intuitive responses to even the simplest questions about likelihood, frequency, and prediction. The heuristics we look at here are based on flow of matter, energy, and information between functional elements in a function-structure diagram [Stone, Wood, Crawford 1998]. Table 1 summarizes the three rules. Readers interested in other heuristics (with some features similar to Module Drivers) may refer to [Zamirowski, Otto 1999].

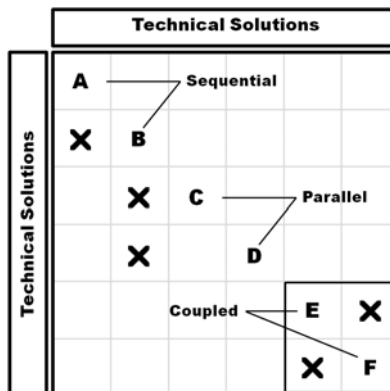
Table 1. Heuristics related to flow [Stone, Wood, Crawford 1998]

Heuristic	Description
Dominant flow	If the same flow of matter, energy, or information goes through a sequence of functions, they should form a module.
Branching flow	If a flow splits up into parallel function chains, the subfunctions that make up those chains should form modules.
Conversion-transmission	Functions that convert one type of flow into another should form modules. If the conversion is followed by transmission, that should be part of the same module.

[Hölttä-Otto 2005] compares all main approaches on their level of repeatability and offers a score based on the ratio of students that successfully apply each of the approaches. The heuristics in Table 1 scored quite high, in particular the application of Conversion-transmission. However, repeatability of a given flow heuristic might be high on a given function structure diagram, but in general the creation of the diagram itself is not a highly repeatable activity. This is supported by [Ulrich, Eppinger 2008] saying “There is no single correct way of creating a function diagram and no single correct functional decomposition of a product.” In contrast, [Kurfman et al 2003] achieved 80% repeatability in an experiment where groups of subjects analyzed a toy ball gun with 15-20 functions to create a functional model, using a particular method. In the author’s experience, repeatability would be lower for significantly more complex products.

2.2. Design Structure Matrix (DSM)

DSM may be thought of as a generic way of mapping interdependencies. Component-based DSM can be used to define modules in a product architecture [Hölttä-Otto 2005]. Task-based DSM may be used to determine the ideal sequence of development tasks in a project [Ulrich, Eppinger 2008]. The Component-based DSM in Figure 2 shows the task of developing B can only be completed once the task of developing A is complete: these are sequential. The tasks of developing C and D both depend on the task of developing B, but once it is concluded, C and D can be developed in parallel. Finally, the tasks of developing E and F are coupled. The best sequence is one that minimizes the number of coupled tasks. DSM predicts E and F should form a module [ibid.].

**Figure 2. DSM is based on mapping dependencies**

2.3. Modular Function Deployment (MFD)

MFD [Erixon 1998] is based on the idea of decomposing CRs into specific statements and linking them to measurable and controllable PPs, decomposing the product into TSs, describing how each TS impacts the performance on a particular PP, and grouping TSs carrying similar properties and strategic intent to define modules. Figure 3 shows how CRs, PPs, TSs, and MDs are visualized in MFD.

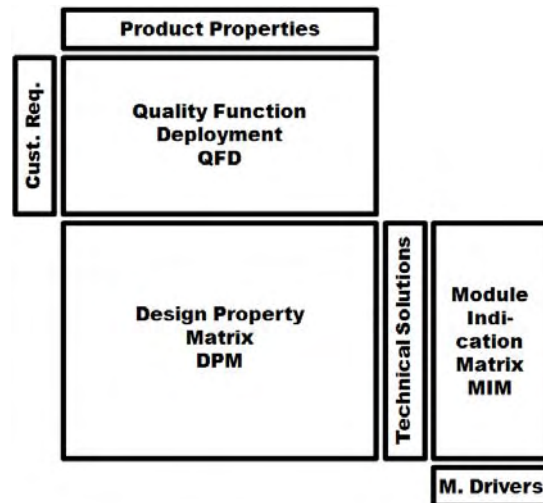


Figure 3. MFD uses three interlinked matrices

The grouping of Technical solutions by Product Property and Module Driver may be done manually or using statistical methods such as hierarchical clustering. Module Drivers are central and unique to MFD, and are presented below in Table 2.

Table 2. The Module Drivers

Module Driver	Strategy	Module Driver	Strategy
Common unit	Use solutions in many variants	Recycling	Simplify scrapping
Carry over	Use solutions in future generations	Strategic supplier available	Use external partner to develop, produce etc
Technical specification	Change specification level	Separate testability	Test separately before final assembly
Styling	Create styling variation	Upgrading	To increase after-sales
Planned design change	Allow for design changes	Process / organization	Protect scarce resources in production or design
Technology push	Incorporate new technology	Service / Maintenance	Easy field replacement

2.4. Functional-Strategic DSM

This method [Blackenfelt 2001] is a hybrid between DSM, MFD, and Heuristics. From DSM, it takes the format for describing dependencies. From MFD, it adds strategic considerations,

but using the Condensed module drivers in Table 3, instead of the original twelve (Table 2). From Heuristics, it adds flow of Matter, Information, and Energy. To this, Blackenfelt adds degree of Spatial interaction.

Table 3. Condensed module drivers [Blackenfelt 2001]

Condensed module drivers	Original twelve Module Drivers	
Commonality	Technical Specification, Styling	Common Unit
Carry Over	Technology Push, Planned Development	Carry Over
Make or Buy	Process/Organization	Strategic Supplier
Life Cycle		Separate Testability, Service/Maintenance, Upgrading, Recycling

Some Module Drivers are mutually conflicting. As an example, a conflict exists between Technical Specification (several performance levels) and Common Unit (one level only). Module Drivers belonging to the same Condensed module driver are said to be supporting if they appear on the same side of the dotted line in Table 3, conflicting otherwise.

For any pair of Technical Solutions, conflicting or supporting strategic objectives are indicated using a scale from -2 to +2 in the Strategic DSM. A similar scheme is used in the Functional DSM. For example, a score of +2 on Spatial would imply two Technical Solutions must be adjacent in space to function; a -2 would signify they absolutely may not be. Figure 7 shows the template for the two matrices. CO, C, MB, and LC refer to the Condensed drivers. S, M, I, E refer to Spatial, Matter, Information, and Energy, respectively.

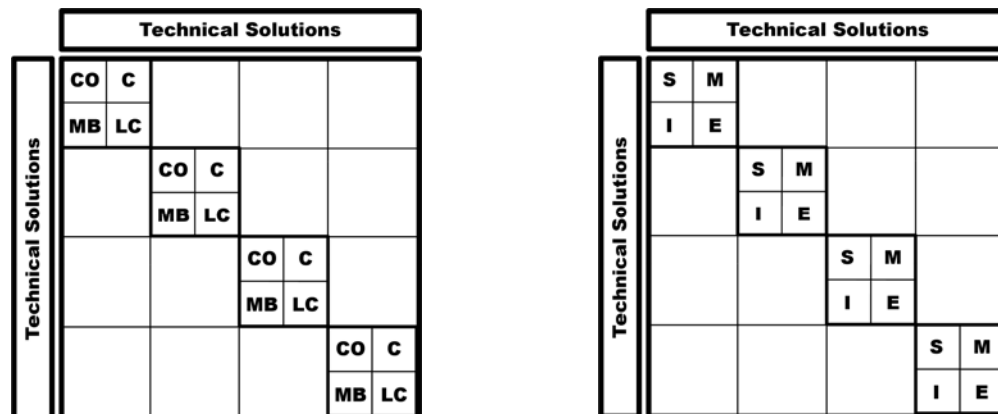


Figure 4. Strategic DSM (left) and Functional DSM (right)

The process of generating modules involves a three-step rule-based algorithm operating on both matrices.

2.5. Extended Implementation Structure Matrix (eISM)

eISM [Sellgren, Andersson 2005] is a hybrid of DSM and MFD. From DSM, it takes the format for describing dependencies. From MFD, it takes QFD and DPM, with FRs replacing PPs. Like MFD, the eISM uses three interlinked matrices, as shown in Figure 5.

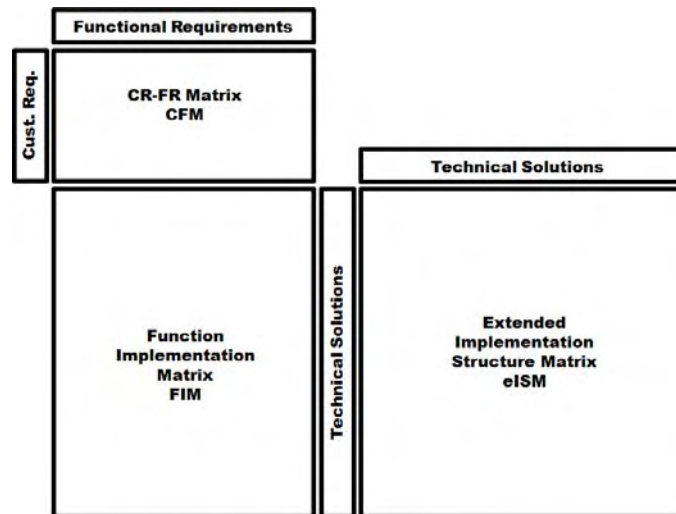


Figure 5. eISM uses three interlinked matrices

The creators of eISM, state the purpose of their approach is “to find a way to bridge the gap between the ‘hard’ technical requirements and the more ‘soft’ interactive requirements” [ibid.]. The FIM represents that bridge. Functional Requirements are stated as verb/noun combinations, which allows eISM to describe more easily how the product is used. A comparison of Figure 3 and Figure 5 shows both MFD and eISM translate CRs into TSs using an intermediate data type. The advantage of PPs is they can be measured, controlled, and assigned a goal value. This is not possible for FRs which may be a disadvantage in many practical applications. On the other hand, soft interactive requirements are harder to describe with PPs. In the end, the choice of PP or FR would depend on the application.

3. Method of Comparison

3.1. Initial set of criteria for comparison

The criteria used in this comparison are based on the works by [Huang 1996], [Hölttä-Otto 2005], [Keller, Binz 2009] and own project experience. The criteria are shown in summary format below.

Table 4. Summary of criteria used in analysis

Source	Category	Criteria
Huang	Functionality requirements	Gather and present facts, Measure performance, Evaluate whether design is good enough, Compare design alternatives, Highlight strengths and weaknesses, Diagnose why an area is strong or weak, Provide redesign advice, Predict what-if effects, Carry out improvements, Allow iterations to take place
	Operability requirements	Easy to learn or well-known concepts, Systematic (all relevant issues considered), Represent product and process data, Teaches good practice, Little effort for designer, Implementation cost and effort, Rapidly effective, Stimulates creativity
	Flexibility & Focus	Allows some degree of flexibility, Reasonably accurate

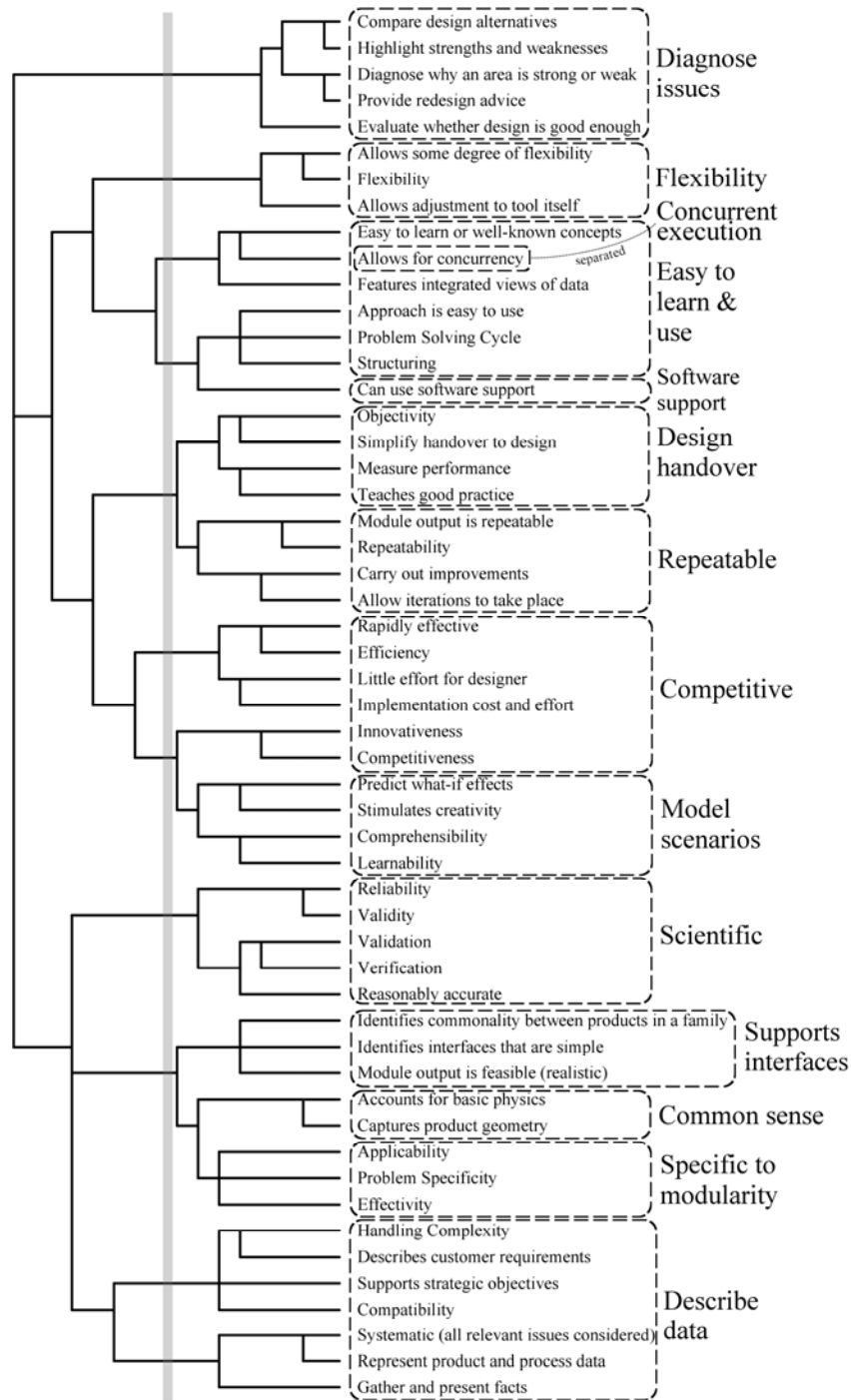
Source	Category	Criteria
Hölttä-Otto	Overall requirements	Identifies commonality between products in family, Identifies interfaces that are simple, Approach is easy to use, Module output is repeatable, Module output is feasible (realistic)
Keller/Binz	Revisability	Validation, Verification
	Pract. Relev. & Competitiveness	Innovativeness, Competitiveness
	Scientific Sound.	Objectivity, Reliability, Validity
	Comprehensibility	Comprehensibility, Repeatability, Learnability, Applicability
	Usefulness	Effectivity, Efficiency
	Prob. Specificity	Problem Specificity
	Struct. & Compatibility	Handling Complexity, Problem Solving Cycle, Structuring, Compatibility
	Flexibility	Flexibility
Experience	Overall requirements	Describes customer requirements, Allows for concurrency, Features integrated views of data, Accounts for basic physics, Captures product geometry, Supports strategic objectives, Can use software support, Simplify handover to design, Allows adjustment to tool itself

Huang's requirements are very focused on what-if-scenario modeling and ease of use. The Hölttä-Otto requirements are very much geared toward interface generation and usefulness of output. The Keller/Binz requirements are very comprehensive but not necessarily specific to architecture generation. The author's own requirements are architecture-specific and somewhat similar to the Hölttä-Otto requirements, but neither set is as comprehensive as either Huang or Keller/Binz. For these reasons, it was relevant to find one set of criteria taken from these models.

4. Results

Figure 6 is a dendrogram of the result, a hierarchical representation of the similarity between criteria.

Figure 6. Dendrogram of the criteria in Table 4.



A dendrogram shows the relative proximity of the different criteria. The basis of the Dendrogram in Figure 6 is a table of distances established by pairwise comparison. Where the dendrogram crosses the gray line, there are 11 subclusters. The final list includes a bias toward the criteria based on Experience, since several of those were not adequately captured in the group of 11 subclusters. Figure 6 indicates that both “Diagnose issues” and “Model

scenarios” are potential evaluation criteria. However, it was found that none of the methods really score on either of these. Therefore, those two were not included. Also, “Allows for concurrency” does not seem to belong in “Easy to learn & use” and was therefore separated and included as “Concurrent execution”. The result is shown in Table 5.

Table 5. Final list of 12 criteria

Criteria	Explanation
Flexibility	Method is flexible, allows adjustments
Concurrent execution	Promotes concurrent execution in groups
Easy to learn & use	Easy to learn, well-known concepts
Software support	Conducive to software support, including large projects
Design handover	Simplify handover from concept phase to detailed design
Repeatable	Method is repeatable and allows iterations
Competitive	Method represents an improvement over existing methods
Scientific	Based on science, valid, verifiable, accurate
Support interfaces	Supports generation of interfaces in modular architecture
Common sense	Allow common sense (physics & product geometry)
Specific to modularity	Specific to generation of modular architecture
Describe data	All data, including customer requirements and strategic intent

To determine whether the criteria seem relevant, the author made an experience-based assessment of each approach, using the 12 criteria. The result is shown in Figure 7.

	Heur.	DSM	eISM	MFD	FS-DSM
Concurrent execution					
Software support					
Describe data					
Support interfaces					
Specific to modularity					
Easy to learn & use					
Design handover					
Scientific					
Repeatable					
Competitive					
Flexibility					
Common sense					

	Strong support
	Medium support
	Weak support
	No support

Figure 7. Final scoring of the five methods

Concurrent execution is possible, to some extent, with matrix-based Methods. In MFD, for example, QFD and DPM scoring can be done by parallel teams, once the PPs are determined. Application of Heuristics depends on a function structure diagram. Before such a diagram is created, it is quite hard to apply any of the rules.

Software support is possible with all methods, but as discussed above, Heuristics relies on a function-structure diagram. Creating such a diagram involves manipulation of graphs, which may be particularly disadvantageous in large projects.

Describe data. MFD describes customer requirements and strategy, as we have seen. FS-DSM does capture strategy, but not requirements. eISM features neither but is unique with its “soft interactive” which increases its score somewhat. Heuristics and DSM are weaker for these three data types.

Support interfaces. MFD and FS-DSM have specific features to support interface generation (make sure modules are clean in terms of strategy and either properties or functions). FS-DSM is stronger in the way it models interactions, but it lacks Product Properties.

Specific to modularity. DSM grew out of a need to plan the sequence of design activities in large projects. The other four methods have specific features to support the generation of modular architecture.

Easy to learn & use. Heuristics relies on a set of rules. One study [Hölttä-Otto 2005] showed it is actually quite difficult to apply the Dominant flow and Branching/Combining rules consistently. In FS-DSM, the algorithm for generating modules is complex and not easy to understand.

Design handover. MFD is the only method that deals consistently with product property goal values, which is an important input in design. MFD and FS-DSM deal with strategic considerations which influences make/buy decisions, among other things. eISM captures “soft interactive requirements” which is shown to be important in design of certain types of products [Sellgren, Andersson 2005]. DSM naturally captures the design sequence.

Scientific. Heuristics is well supported by empirical research on hundreds of real products, but its theoretical foundation is not as clear as the other methods.

Repeatable. Because of its simplicity, DSM is the only method that receives a full score here. The other methods are believed to be roughly equal in terms of repeatability.

Competitive. Full score here indicates the method captures something that is unique to that method. Heuristics does a good job of describing the underlying physics of the product. MFD is unique in its treatment of customer requirements. eISM, as we have seen, is strong on “soft interactive”. FS-DSM is based on integration of MFD and DSM and offers nothing truly new (except the integration itself).

Flexibility is lower in DSM because of the fixed TS-TS format. Strategy may be incorporated, as in FS-DSM, but it requires an additional matrix, so it is no longer a pure Component-based DSM.

Common sense. Heuristics, DSM (Component-based) and FS-DSM naturally support underlying physics and spatial product considerations.

5. Conclusions and Discussion

This paper outlines a systematic approach to the comparison of methods across a consistent set of criteria, based on literature research and own experience. Subjectivity cannot be avoided, but the process of *first* determining criteria based on external sources, and *second*

scoring the methods on these criteria avoids the problem of a completely opinion-based analysis, which will only ever confirm what we knew from experience.

How did the Methods fare overall? For large projects, where *Describe data*, *Software support*, and *Concurrent execution* may be important, the matrix-based methods scored higher than Heuristics, which might be better suited to small projects where *Flexibility* is valued. Of the matrix-based methods, pure DSM seems to offer fewest advantages, apart from *Easy to learn & use*.

How did the hybrid Methods fare, specifically? Overall scoring indicates FS-DSM and eISM are *at least as strong* as either MFD or DSM, on which they are built. In fact, FS-DSM is stronger than MFD in the way it deals with spatial considerations (absent in MFD), and stronger than DSM in its integration of strategic considerations. However, module generation is more complex than in either of the original methods. Similarly, eISM successfully captures “soft interactive”, absent in both DSM and MFD, but sacrifices both property goal values and strategic considerations (present in MFD). Do hybrid methods offer improvement, then? Yes, but they seem to suffer from new disadvantages, absent in the original methods.

What would be the best method on which to base new hybrids? Several approaches are possible. It has been shown how DSM can be extended [Blackenfelt 2001] to cover degrees of strategic and spatial considerations (FS-DSM). One path would be to attempt to add property goal values, the absence of which is perhaps the main weakness in FS-DSM. MFD is open enough to accommodate new data types, so extending MFD with spatial or “soft interactive” properties might be another path. Finally, eISM can easily be extended to cover strategy, essentially by adding a MIM (see Figure 3) to the ISM (see Figure 5). Module generation may be similar to the approach in FS-DSM.

5. Acknowledgements

The author wishes to thank the following people for valuable feedback on this paper:

Dr. Ulf Sellgren (Dept. of Machine Design, Royal Institute of Technology, Stockholm), Dr. Mark W. Lange (Modular Management USA, Inc.), Dr. Gunnar Erixon (Sweden Modular Management AB), and Dr. Katja Hölttä-Otto (Dept. of Mechanical Engineering, University of Massachusetts Dartmouth).

6. References

- Blackenfelt, M. W., *"Managing complexity by product modularisation - Balancing the aspects of technology and business during the design process"*, Doctoral Thesis, Dept. of Machine Design, Royal Institute of Technology, Stockholm, 2001.
- Erixon, G., *"Modular Function Deployment - A Method for Product Modularisation"*, Dept. of Manufacturing Systems, Royal Institute of Technology, Stockholm, 1998.
- Gilovich, T., Griffin, D., Kahneman, D., *"Heuristics and Biases – The Psychology of Intuitive Judgement"*, Cambridge University Press, Cambridge, UK, 2002.
- Huang, G. Q., *"Developing Design For X Tools"*, Design For X - Concurrent Engineering Imperatives, Huang, G. Q. (ed), Springer, New York, 1996.
- Hölttä-Otto, K., *"Modular Product Platform Design"*, Doctoral Dissertation, Dept. of Mechanical Engineering, Helsinki University of Technology, Espoo, 2005.
- Keller, A., Binz, H., *"Requirements on Engineering Design Methodologies"*, Proceedings of the 2009 International Conference on Engineering Design (ICED09), Stanford, 2009.
- Kurfman, M., Stock, M. E., Stone, R. B., Rajan, J., Wood, K. L., *"Experimental Studies Assessing the Repeatability of a Functional Modeling Derivation Method"*, ASME Journal of Mechanical Design, 125(4), 2003, pp. 682-693.
- Sellgren, U., Andersson, S., *"The Concept of Functional Surfaces as Carriers of Interactive Properties"*, International Conference on Engineering Design (ICED 05), Melbourne, 2005.
- Stone, R. B., Wood, K. L., Crawford, R. H., *"A Heuristic Method to Identify Modules from a Functional Description of a Product"*, Proceedings of 1998 ASME Design Engineering Technical Conferences (DETC98), Atlanta, 1998.
- Ulrich, K. T., Eppinger, S. D., *"Product Design and Development"*, McGraw-Hill Education Asia, Singapore, 2008.
- Zamirowski, E. J., Otto, K. N., *"Identifying Product Portfolio Architecture Modularity Using Function and Variety Heuristics"*, Proceedings of the 1999 ASME Design Engineering Technical Conferences (DETC99/DTM-8760), Las Vegas, 1999.

Corresponding author Fredrik Borjesson, Ph.D. student, Department of Machine Design, Royal Institute of Technology (KTH), Brinellvägen 85, SE-100 44 Stockholm, Sweden
VP of Technology, Modular Management USA, Inc., 8030 Old Cedar Ave S, Suite 203, Bloomington, MN 55425-1215, USA. Telefax +1 952 854 5586
Email fredrik.borjesson@modularmanagement.com

Modularization of novel machines: motives, means and opportunities

Fredrik Börjesson^{1&2} & Ulf Sellgren²

¹*Modular Management USA, Inc., Bloomington, Minnesota, USA*

fredrik.borjesson@modular.se

²*Machine Design, Royal Institute of Technology (KTH), Stockholm, SWEDEN*

ulfs@md.kth.se

Abstract

Modularization approaches are often used to restructure mature products with known technical content, but not to assist new development of products with a high innovation content or soft interactive requirements. This paper investigates if various clustering techniques can be used to identify module candidates in matrix representations of evolving product properties, including interactive properties, and component architectures. The proposed approach is tested on the hybrid drive train of a novel forwarder. Forwarders are used in the forestry industry to transport logs from the felling area to a landing area close to a road accessible by trucks. Continuous efficiency improvements, new emission requirements, and the need to configure machine for different applications stresses the need for a modular product architecture.

Keywords: *DSM, forwarder, hybrid technology, new development, dendrogram.*

1 Introduction

To be able to meet the international competition, a sustainable productivity increase of 2 to 3 %, on an annual basis, and a significant improvement of the fuel economy is required [1]. At the same time, new legislation is significantly decreasing the allowed emission levels from the diesel engine. The modest size of the international forwarder market, about 3000 machines per year, combined with a relatively large product variety required to target very different tasks and conditions, stresses the need for a modular architecture with an integrated forwarder and harvester platform, a hybrid driveline, and all-wheel drive and steering.

First, the concepts of component-based Design Structure Matrix (DSM) [2] and function-structure heuristics [3] are shown, followed by Modular Function Deployment (MFD) [4] in an extended form that incorporates function-structure heuristics [5]. Second, a development case for a novel forestry machine is presented, where the results of clustering analyses based on MFD and DSM are compared. We show that when results differ, and argues that this may indicate that further decomposition of technical solutions is required. Finally, the ability of the methods to assist modularization of novel products is discussed.

2 Frame of reference

A DSM may be thought of as a generic method of mapping interdependencies. A (component-based) DSM can be used to identify modules in a product architecture [6]. In Figure 1 below, technical solutions E and F have a non-causal interaction, and thus form a

natural module candidate. The process of generating modules from a DSM is referred to as *clustering*.

		Technical Solutions					
Technical Solutions	A						
	1	B					
		1	C				
		1		D			
					E	1	
					1		F

Figure 1. DSM is based on mapping dependencies

Stone, Wood, and Crawford [3] have shown that three heuristics, shown in Figure 2 below, may be applied on function structure diagrams, to form modules.

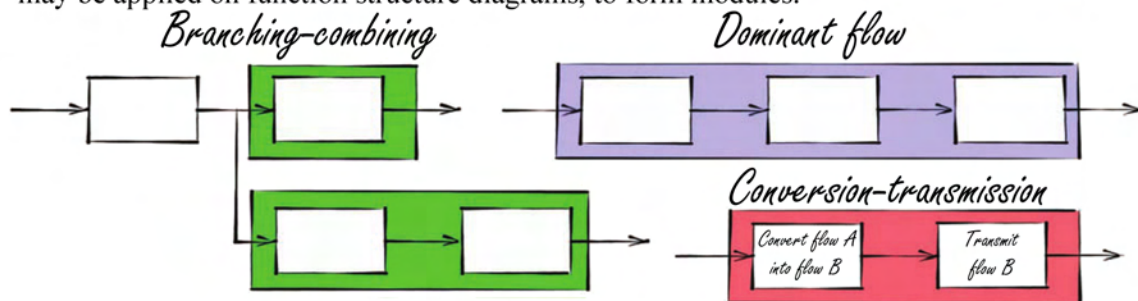


Figure 2. Three function structure heuristics

Modular Function Deployment (MFD), as proposed by Erixon [4], is a well established methodology for strategic modularization [7][8]. MFD builds on the three interlinked matrices; the Quality-function Deployment “House of quality” (QFD), the Design-property Matrix (DPM), and the Module-Indication Matrix (MIM), collectively referred to as Product Management Map (PMM). The MIM is the central central part of MFD, as proposed by Erixon [4]. In the MIM, the technical solutions are characterized by their *module drivers*, which support the grouping of technical solutions into a modular concept. Borjesson [5] extended MFD with a set of new property types, referred to as *convergence properties*. Figure 3 shows the property types used in this analysis. Interactive properties are based on Sellgren and Andersson [9]. Figure 4 shows a PMM extended with the convergence properties from Figure 3 and a component DSM to describe physical interactions between individual technical solutions. Figure 3 also shows how the proposed convergence properties are related to the technical solutions with a Convergence Property Matrix (CPM).

Class	Category	Sub-category	Purpose
Convergence Properties	Function-structure heuristics	Branching-combining	Configure system by application type
		Dominant flow	Apply function-structure heuristic
		Conversion-transmission	Not used
	Geometry of product		Traditional product geometry (usability and service)
Product Properties	Interactive property		Change soft interactive product aspects (feel)
	System		Selected by engineer subject to technical constraints
	Option		Customer selectable optional features (discrete choice)
	Variance		Customer selectable performance levels

Figure 3. Properties used in the present study

Hierarchical clustering has been proposed [7][10] as an alternative way of identifying module candidates. It may be applied using different algorithms, such as Ward and Centroid , see e.g. [11]. All algorithms rely on a distance metric, to determine whether technical solutions share similar scoring on product properties and module drivers. There are different distance metrics, such as Euclidean distance, Squared Euclidean, Manhattan city-block, and Pearson correlation, see e.g. [12]. Pearson is radically different from the other three in that it essentially looks at the angle between vectors instead of absolute distances between points.

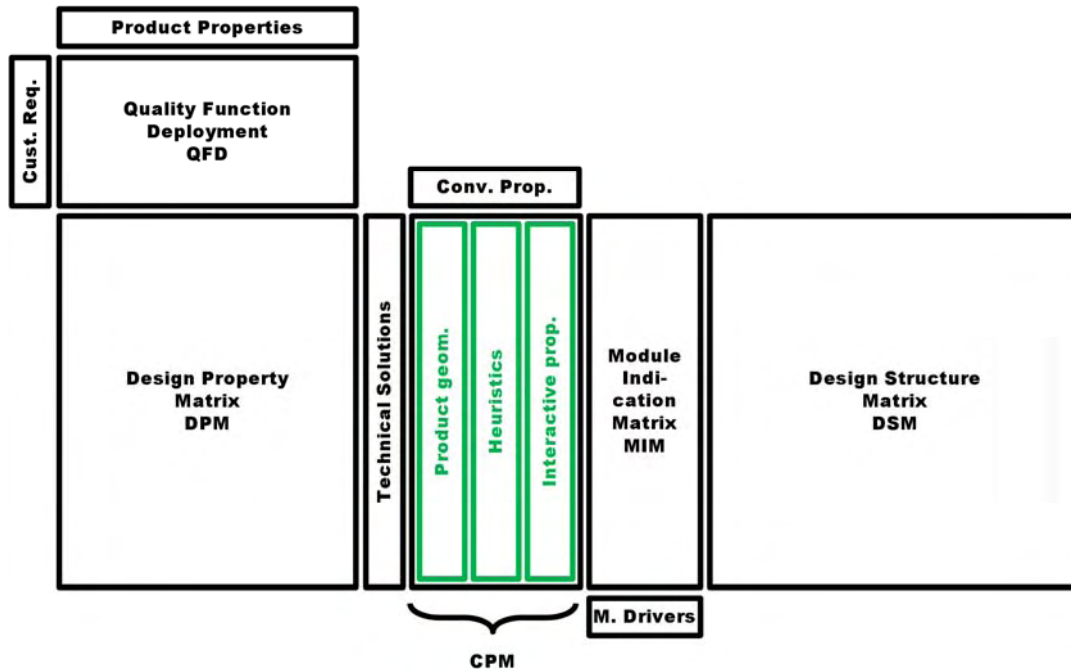


Figure 4. PMM extended with convergence properties and DSM

3 Case study – hybrid drive

3.1. Dealing with options in the QFD matrix

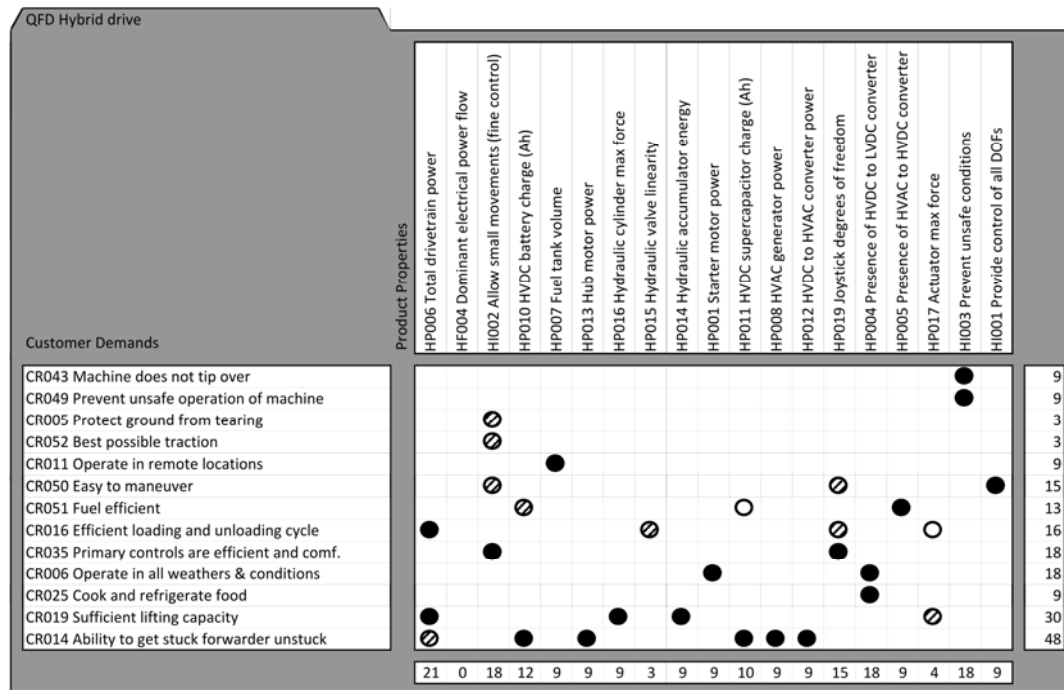


Figure 5. Quality Function Deployment matrix for hybrid drive

The QFD in Figure 5 shows two option properties, HP004 and HP005, that both dictate technical solutions to address two particular technical considerations: whether the vehicle uses an HVDC-LVDC converter to generate power for starter motor (and applications like cooking), and whether power is regenerated when vehicle brakes or moves downhill. Specifying specific solutions in the QFD, as a rule, is not a good idea. In this particular case, these two optional features were determined to be relevant in terms of project scope. The introduction of these option properties is a convenient shorthand for several properties such as starter motor power, current control, regeneration efficiency, maximum brake disc heat power etc that are solution-independent alternatives.

3.2. Enabling flexible configurations

Another high-level desire was that of configuring the machine flexibly for different applications. The branching-combining heuristic in Figure 2 is very useful in this particular case, because it helps us identify one of the key interfaces that allow flexible reconfiguration of the drive train, as shown on the left in Figure 6 below. A proposed DPM representation of the same relation is shown on the right.

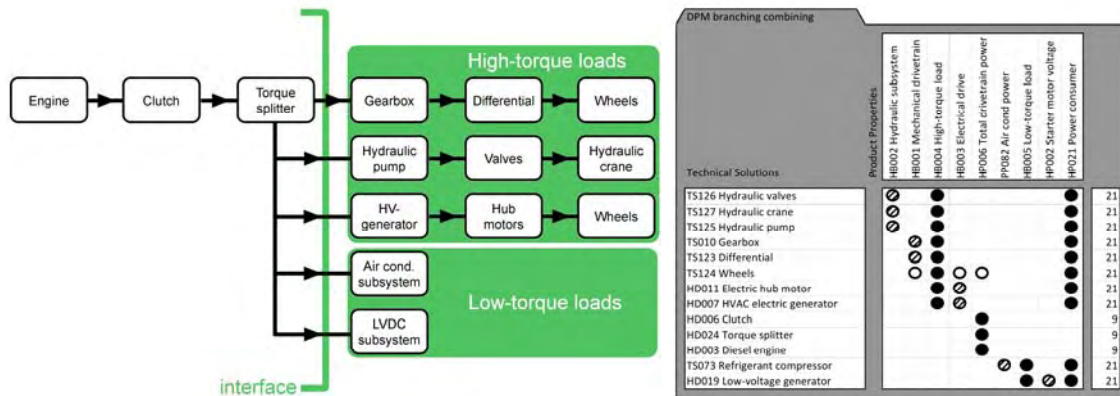


Figure 6. Branching-combining heuristic and proposed DPM representation

3.3. Function structure of hybrid drive

A conceptual function structure diagram was created for two reasons. First, it is a practical first step toward the technical solution decomposition. Technical solutions are used in both MFD and DSM. Second, to score a component DSM, we need to understand the flow of energy, matter, and information. A high level function structure diagram is shown in Figure 7.

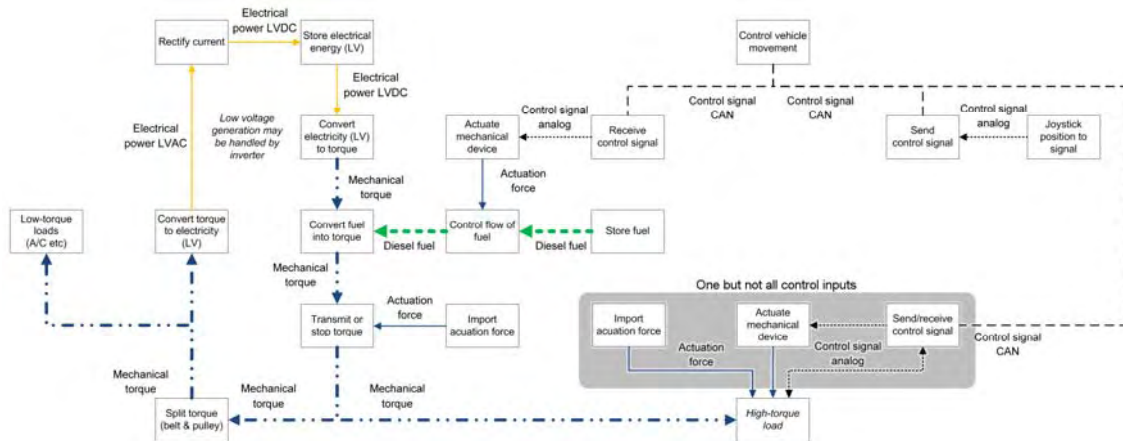
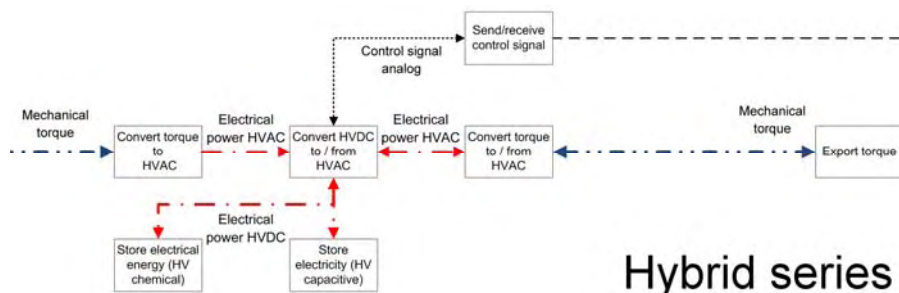


Figure 7. Function structure diagram of forwarder



Hybrid series

Figure 8. Hybrid series drive represented as a high-torque load

The forwarder has four main high-torque loads: three drives (series/parallel/mechanical) and the hydraulic crane. A series hybrid is the main option in the project, because of the improved ground clearance resulting from hub-mounted motors instead of a mechanical transmission. The series hybrid is shown in Figure 8. Power dissipation is not shown.

3.4. Decomposition of the inverter

The inverter has to support several hybrid operating modes, as shown in Figure 9.

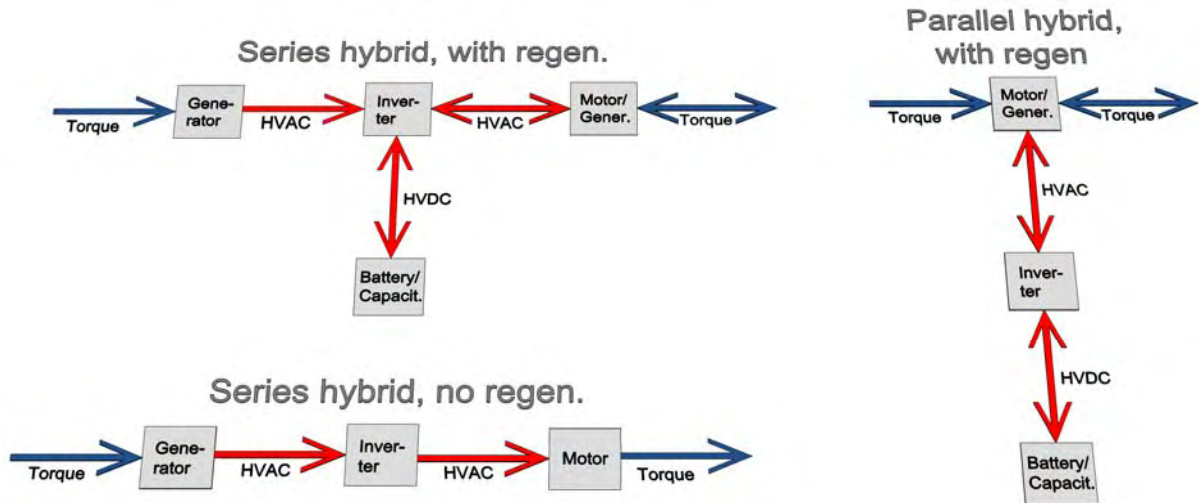


Figure 9. Inverter must support at least three hybrid configurations

In addition to these three operating modes, we wish to offer an optional HVDC to LVDC conversion feature. We clearly cannot view the inverter as a black box, even on this conceptual level. A proposed next level of disaggregation is shown in Figure 10.

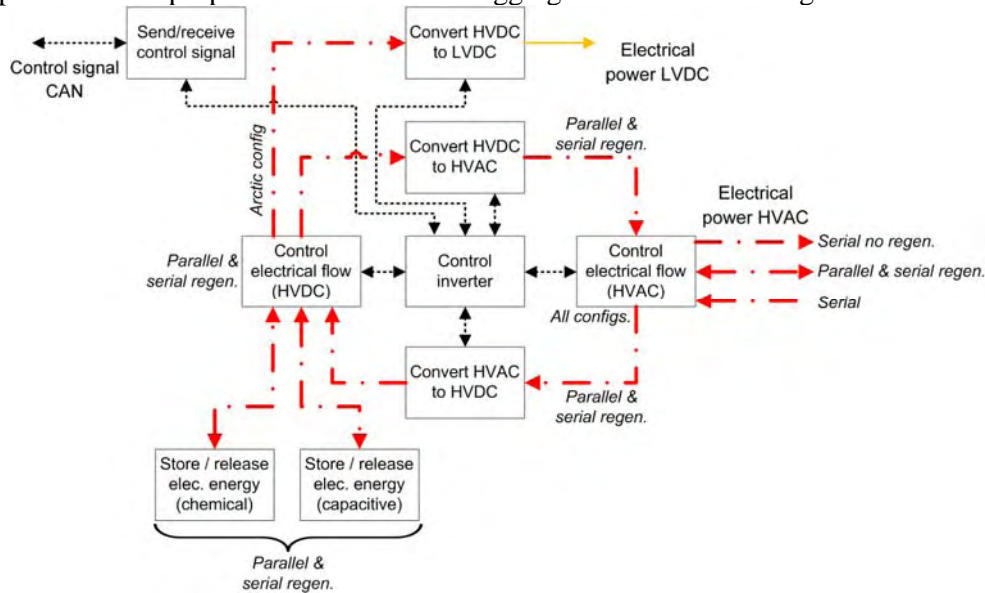


Figure 10. Proposed conceptual decomposition of inverter

Please note that this graphic does not necessarily reflect the inner workings of any actual inverter but is meant instead to illustrate the idea of further decomposition of selected system components. Three main functions are apparent here. First, conversion of electrical power from one form to another happens in one of three converters. Second, the flow of energy is controlled by some device we envision as a switching grid. These are the electrical flow controllers, which we assume must come in HVDC and HVAC versions. Third, there is a central intelligence that controls these devices, and makes instant decisions to reroute power when driving conditions change. The inverter does not rely on the vehicle computer for these decisions, to avoid some unsafe state. If it did, and communications failed, the inverter could conceivably end up in some unsafe state. Armed with this deeper understanding of the inverter, drive-related customer requirements and product properties from Figure 5, and the component-level interactions shown in Figure 7, Figure 8, and Figure 10 we may create a complete PMM with convergence properties and DSM, as shown schematically in Figure 4.

4 Results

4.1. Result of MFD clustering

The resulting dendrogram, from DPM clustering, is shown on the left in Figure 11 (generated using SPSS [13]). Note how the torque splitter ends up in a cluster with other communications-related or electronic devices. This is not caused by a simple error in scoring. Rather, the choice of distance metric is the main reason, and when we re-run the clustering with Centroid / Pearson correlation, we get the dendrogram shown on the right in Figure 11. Here, we see the torque splitter together with diesel engine and clutch. They are all mechanical and all related to the production or distribution of torque.

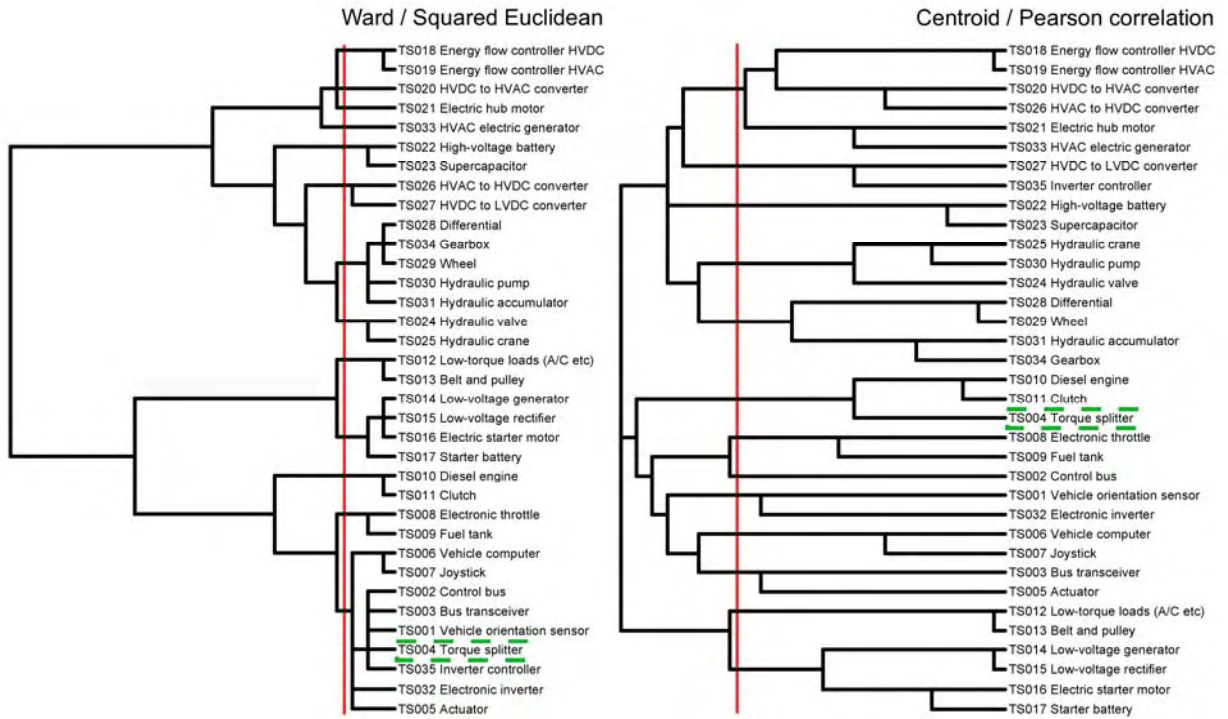


Figure 11. Dendrograms obtained by two algorithms / distance metrics

The two vertical lines intersect the dendrograms where there are exactly 13 subclusters. MFD-theory predicts [4] the ideal number of modules is equal to the square root of the number of technical solutions, which would be around 6 in this case. This is only true, however, if the decomposition is such that a technical solution corresponds to a simple component. For small to medium-sized systems on a high level of abstraction, the authors have found that a good guess is usually between one third and one half of the number of technical solutions. In this case, the vertical line makes it possible to compare the output of the two runs more easily.

4.2. Result of DSM clustering

A component DSM for the drive system is shown in the graphic below. The output was generated with the clustering feature of an Excel macro called Complex Problem Solver by RedTeam [14]. If we assume, as we did in the interpretation of the dendrograms (Figure 11), that a suitable module might have up to half a dozen technical solutions, we may add the boxes shown in Figure 12. When boxes overlap, we have to make a manual assessment. We can now compare the output of MFD with the output of DSM. The groupings in the column

labeled technical solution reflect the authors’ assessment of the output. The electronic inverter has been shaded, because the inverter was decomposed, as shown in Figure 10.

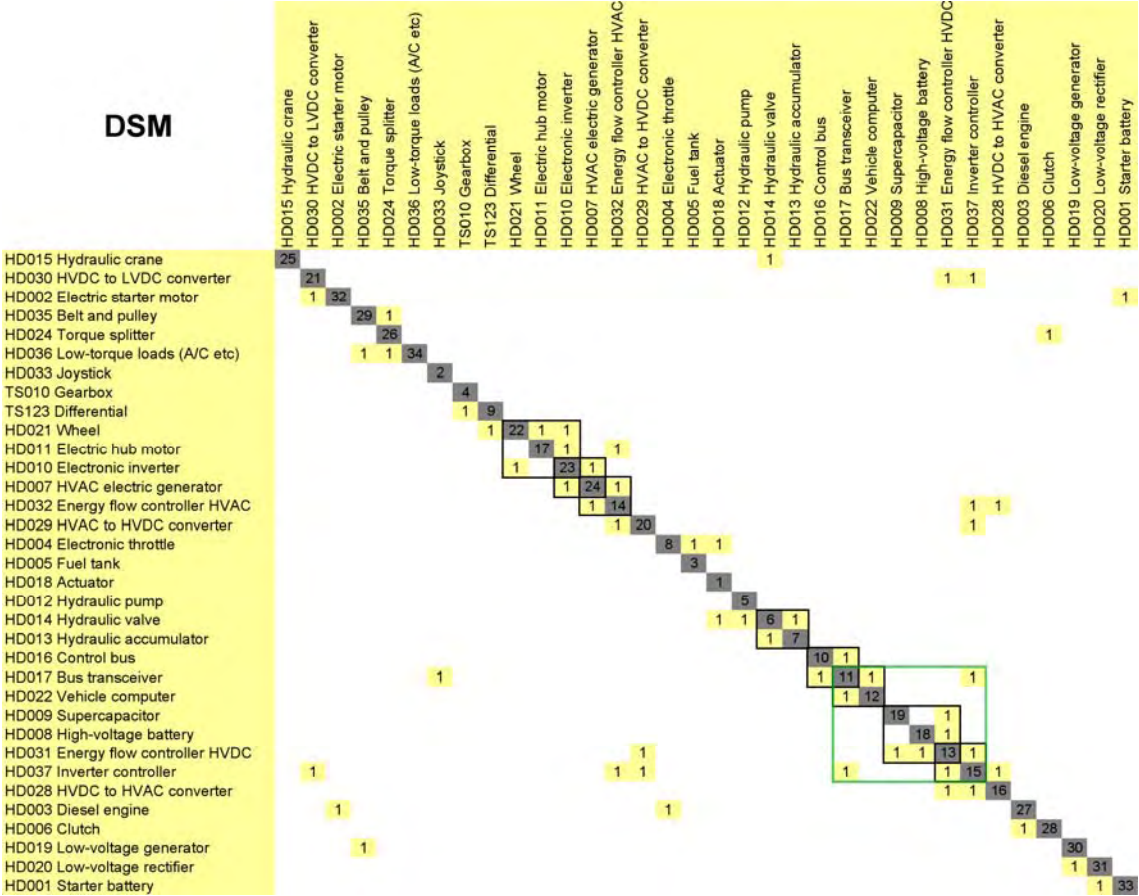


Figure 12. Design Structure Matrix for drive train system

4.3. Comparison of output from MFD and DSM

The outputs of MFD and DSM are compared in Figure 13. Where the predictions by MFD and DSM differ significantly, it is important to keep in mind that DSM deals with interactions only. MFD may capture some of those interactions (shared Product Properties), but more importantly, why things may need to change, for customer reasons (configurations and performance levels) or company-specific reasons (the drivers). Note how DSM predicts TS033 HVAC generator and TS019 Energy flow controller HVAC belong together. Figure 10 shows how TS019 changes depending on hybrid drive type and whether regeneration is enabled. For series hybrids, the HVAC generator might be the same with or without regeneration, so that might be an argument not to group it with the HVAC energy flow controller.

MFD with Convergence Properties			DSM					
Ward/ Squared Euclidean	Technical Solution	Centroid/ Pearson Correlation	D01	D02	D03	D04	D05	D06
W01	TS008 Electronic throttle TS009 Fuel tank	C01						
W02	TS012 Low-torque loads (A/C etc) TS013 Belt and pulley	C02						
W03	TS014 Low-voltage generator TS015 Low-voltage rectifier TS016 Electric starter motor	C03						
	TS017 Starter battery							

(figure continued from previous page)

W04	TS010 Diesel engine TS011 Clutch	C04			
W05	TS004 Torque splitter				
	TS006 Vehicle computer	C05		x	
	TS007 Joystick				
	TS005 Actuator	C06			
	TS003 Bus transceiver			x	
	TS002 Control bus	C07		x	
	TS001 Vehicle orientation sensor	C08			
W06	TS032 Electronic inverter				
	TS035 Inverter controller	C09			x
	TS027 HVDC to LVDC converter				
W07	TS026 HVAC to HVDC converter				
	TS018 Energy flow controller HVDC TS019 Energy flow controller HVAC	C10	x		x
W08	TS020 HVDC to HVAC converter				
W09	TS021 Electric hub motor		x		
W10	TS033 HVAC electric generator		x		
W11	TS022 High-voltage battery TS023 Supercapacitor	C11			x
					x
W12	TS029 Wheel	C12	x		
	TS028 Differential				
	TS034 Gearbox				
	TS031 Hydraulic accumulator			x	
W13	TS030 Hydraulic pump				
	TS024 Hydraulic valve	C13		x	
	TS025 Hydraulic crane				

Figure 13. Comparison of output from MFD and DSM

The final module proposal is shown below in Figure 14. For each module, one or several module variants are listed, as well as three basic configurations: one series hybrid without regeneration of power, one high-power series with regeneration for use in cold climates, and one parallel hybrid.

Module	Technical solutions	Comment / Explanation	Variants	Non-Arctic serial	Powerful Arctic serial	Standard parallel
M01	Electronic throttle, Fuel tank	Brought together by dominant flow (of fuel). Geometrical separation probably necessary. Fuel tank varies by volume. Could share space with M19.	M01A Std volume (250 liters), M01B Extra large (350 liters)	M01A	M01B	M01A
M02	Low-torque loads, Belt and pulley	Belt and pulley should provide a convenient way of connecting multiple low-torque loads.	M02A Standard arrangement for low-voltage generator and A/C, M02B for low-voltage gen only	M02A	M02B	M02A
M03	Low-voltage generator, Rectifier	Strong candidate for Common Unit, unless low-voltage DC level needs to be selectable (12 or 24V).	M03A Standard generator & rectifier for 24V	M03A	-	M03A
M04	Starter battery	De-facto module. Varies by capacity.	M04A Standard generator with 55 Ah battery, M04B Standard generator with 75 Ah battery	M04A	M04B	M04A
M05	Electric starter motor	Mandatory. May need higher power in cold climates.	M05A Standard starter motor, M05B Cold-climate starter motor (higher power)	M05A	M05B	M05A
M06	Torque splitter, Diesel engine, Clutch	Torque splitter should physically be located on the outgoing shaft of the engine. It enables the connection of high-torque load, plus the belt and pulley arrangement. Power variation expected.	M06A Deutz TCD 2012 6-cyl 110 kW at 1500 rpm, M06B Deutz TCD 1013 6-cyl 200 kW at 1500 rpm	M06A	M06B	M06A
M07	Vehicle computer, Joystick	Geometrical challenge. Could computer be co-located with joystick?	M07A Standard response, M07B Enhanced responsiveness	M07A	M07B	M07A
M08	Actuator	May come in a few variants (rotation/linear movement, weak/strong, slow/fast etc)	M08A Standard, M08B Enhanced linearity	M08A	M08B	M08A
M09	Bus transceiver	This module communicates with the Microprocessor, but can also act stand-alone, for example with an Actuator.	M09A Standard transceiver	M09A	M09A	M09A
M10	Control bus	Pre-wired to any location where access to control bus might be required.	M10A Standard bus	M10A	M10A	M10A
M11	Vehicle orientation sensor	Example of a sensor required for a particular option. All sensors should have the same interface if possible, electrically and mechanically.	M11A Sideways tilt only, M11B Full vehicle orientation	M11A	M11B	M11A
-	Electronic inverter	Inverter is not one single module. Decomposed into converters and controllers, see below.	-	-	-	-
M12	Inverter controller	Controls the inverter. Switches rapidly between motor mode and generator mode. Prevent unsafe conditions even if link to vehicle computer fails.	M12A Serial drive no regeneration, M12B Parallel or serial drive with regeneration	M12A	M12B	M12B
M13	HVDC to LVDC converter	Generate LVDC (12 or 24V) from HVDC. Would replace M03 (24V only). Useful option in cold climates where more energy may be required to start the Diesel engine.	M13A Standard power 12V, M13B Standard power 24V, M13C Extra power 24V	N/A	M13C	-
M14	HVAC to HVDC converter	Rectify HVAC from generator, to store in M19. Needs sensors and related circuitry so as not to overcharge batteries. Module enables regeneration of power.	M14A Standard converter	-	M14A	M14A

(figure continues on next page)

(figure continued from previous page)

Module	Technical solutions	Comment / Explanation	Variants	Non-Arctic serial	Powerful Arctic serial	Standard parallel
M15	Energy flow controllers (HVDC, HVAC)	This device controls the flow of energy going in and out of the inverter, both DC and AC. When HVDC flow controller is present, M13 may be connected (e.g., not available in series without regeneration).	M15A HVAC in plus HVAC out for series without regeneration; M15B HVDC and HVAC bidirectional for parallel drive; M15C HVDC and HVAC mono and bidirectional for series with regeneration	M15A	M15C	M15B
M16	HVDC to HVAC converter	Generate electrical power from stored (HV) energy. Both power level variation and possible Technology Push.	M17A Standard power, M17B Extra power	-	M17B	M17A
M17	Electric hub motor	Comes in different torques and power ratings. Hub motors also work as generators. Not available in parallel configuration.	M15A Standard hub motor, M15B Extra strong	M15A	M15B	N/A
M18	HVAC electric generator	Part of the serial hybrid drive. Converts torque to HVAC. Motor/generator variant for parallel has different mechanical interface (both ends of rotating shaft available).	M16A Generator only, M16B Motor/generator for parallel	M16A	M16A	M16B
M19	High-voltage battery, Supercapacitor	All energy storage conveniently focused in one single module. Could share space with M01.	M19A 4x supercapacitor Maxwell BMOD0063-P125-B14 (100 Wh at 450 V, total peak power 400 kW) plus 9x Effpower LIC Power battery 150V (total peak power 270 kW)	-	M19A	M19A
M20	Wheel	Mandatory. Power comes either from mechanical drivetrain or hub-mounted electric motors.	M20A Standard wheel, M20B Extra-wide	M20A	M20B	M20A
M21	Gearbox, Differential	This module is essentially the mechanical drivetrain. Same interface as all other high-torque loads.	M20A Mechanical drive front wheel-pair	-	-	-
M22	Hydraulic accumulator	Optional. May improve performance of hydraulic subsystem in terms of max power.	M22A Standard accumulator	M22A	M22A	M22A
M23	Hydraulic pump	Mandatory if vehicle has hydraulic subsystem. Connects as any other high-torque load. Power level variation expected.	M23A Standard pump, M23B Extra-powerful	M23A	M23B	M23A
M24	Hydraulic valve, Hydraulic crane	If you have a crane, you need a valve to control it. If valve linearity needs to change or improve, that would be an argument to place it in its own module.	M24A Crane standard responsiveness, M24B Crane enhanced responsiveness	M24A	M24B	M24A

Figure 14. Modules, variants, and three basic configurations

5 Conclusions and discussion

The following conclusions can be made:

- MFD can be extended to support non-customer driven configurations
- Real world properties are not ideal
- For a medium-sized conceptual system the ideal number of modules is between one third and one half of the number of principal technical solutions
- Disaggregate solutions further when MFD and DSM outputs are in conflict

The branching-combining heuristic proved particularly useful, in this case study. It required six additional properties: one for each configuration plus two to separate torque production and torque consumption. If you already know what you are driving towards, why not jump directly to the conclusion? Such an approach probably works well in small systems, but in very large systems, there is simply too much data to rely on an “intuitive” approach. The use of heuristics in conjunction with more traditional property types may be necessary, to promote convergence on certain solutions in a few subsystems, but allow the statistical processing to address the rest.

Ideal properties are measurable, controllable, and solution-free, but convergence properties typically are not. Does that mean the rigor associated with an appropriate selection of properties is out the door? No. A rigorous selection of properties based on customer requirements is a very good first step. Convergence properties, such as the ones shown in Figure 3, may be introduced successively, to address geometrical concerns, known features, spatial information et cetera. The hierarchical clustering should be re-performed each time a new property class is introduced.

This study has 34 principal technical solutions, but a detailed hybrid drive train has thousands, which is beyond the practical limit of a normal concept study. Project members must make some decisions about the initial level of disaggregation. A useful rule-of-thumb might be: *If most of your technical solutions come out as modules in their own right, the level of*

disaggregation is probably insufficient. Comparing the output of MFD and DSM might be a way of spotting areas where further disaggregation is needed. In this case, it pointed to the need for further disaggregation of the HVAC energy flow controller. MFD grouped it with the HVDC energy flow controller but DSM with the generator. Areas of commonality and variance within the energy flow controller itself must be understood further to make the call.

6 Acknowledgements

The authors wish to acknowledge the kind support of Dr. Gunnar Erixon (Modular Management AB, Sweden) and Dr. Katja Hölttä-Otto (University of Massachusetts Dartmouth).

7 References

- [1] Löfgren, B., “Kinematic Control of Redundant Knuckle Booms with Automatic Path-Following Functions”, Doctoral Thesis, Dept. of Machine Design, Royal Institute of Technology, Stockholm, Sweden, 2009.
- [2] Pimmler, T.U. and Eppinger, S.D., “Integration analysis of product decompositions”, ASME Design Theory and Methodology Conference, DE Vol. 68, 1994.
- [3] Stone, R. B., Wood, K. L. and Crawford, R. H. “A Heuristic Method for Identifying Modules for Product Architectures”, Design Studies, vol. 21, no. 1, 2000, pp. 5-31.
- [4] Erixon, G., “Modular Function Deployment - A Method for Product Modularisation”, Doctoral Thesis, Dept. of Manufacturing Systems, Royal Institute of Technology, Stockholm, Sweden, 1998.
- [5] Borjesson, F., “Improved Output in Modular Function Deployment Using Heuristics”, *Proc. of the 17th International Conference on Engineering Design (ICED '09)*, Vol. 4, ISBN 9-781904-670087, Stanford, 2009, pp 1-12.
- [6] Hölttä-Otto, K., "Modular Product Platform Design", Doctoral Dissertation, Dept. of Mechanical Engineering, Helsinki University of Technology, Espoo, Finland, 2005.
- [7] Stake, R., “On conceptual development of modular products”, Doctoral Thesis, Dept. of Production Engineering, Royal Institute of Technology, Stockholm, Sweden, 2000.
- [8] Blackenfelt, M., “Managing complexity by product modularization”, Doctoral Thesis, Dept. of Machine Design, Royal Institute of Technology, Stockholm, Sweden, 2001
- [9] Andersson, S. and Sellgren, U., “Representation and Use of Functional Surfaces”, *Proc. of 7th Workshop on Product Structuring – Product Platform Development*, Gothenburg, Sweden, 2004, pp 87-100.
- [10] Hölttä-Otto, K., Tang, V., and Otto, K. “Analyzing module commonality for platform design using dendrograms”, *Research in Engineering Design*, Vol. 19, No. 2, 2008, pp. 127-141.

(reference list continues on next page)

- [11] Hair, J. F., Black, B., Babin, B., Anderson, R. E., and Tatham, R. L., “Multivariate Data Analysis”, 6th edition, Prentice Hall, Upper Saddle River, 2005.
- [12] Rodgers, J. L. and Nicewander, W. A., “Thirteen Ways to Look at the Correlation Coefficient”, The American Statistician, Vol. 42, No. 1 (February 1988), American Statistical Association, Alexandria, pp. 59-66.
- [13] SPSS 12.0 by SPSS Inc., <http://www.spss.com/>.
- [14] Complex Problem Solver 1.2.2 by RedTeam, <http://www.redteam.se/>, downloaded March 03, 2010.

DETC2012-70076

IMPROVED CLUSTERING ALGORITHM FOR DESIGN STRUCTURE MATRIX

Fredrik Borjesson¹

VP of Technology,
Modular Mgmt USA, Inc.
Bloomington, MN 55425-
1215

Ph.D. stud., Dept. of
Machine Design
Royal Inst. of Tech.
Stockholm, Sweden

Katja Hölttä-Otto

Associate Professor of Mechanical Engineering
University of Massachusetts Dartmouth
North Dartmouth, MA 02747-2300

ABSTRACT

For clustering a large Design Structure Matrix (DSM), computerized algorithms are necessary. A common algorithm by Thebeau uses stochastic hill-climbing to avoid local optima. The output of the algorithm is stochastic, and to be certain a very good clustering solution has been obtained, it may be necessary to run the algorithm thousands of times. To make this feasible in practice, the algorithm must be computationally efficient. Two algorithmic improvements are presented. Together they improve the quality of the results obtained and increase speed by a factor of seven to eight for normal clustering problems. The proposed new algorithm is applied to a cordless handheld vacuum cleaner.

Keywords Design Structure Matrix; Clustering algorithm; Stochastic hill-climbing

¹ Address all correspondence to this author. Email: fredrik.borjesson@modularmanagement.com

NOMENCLATURE

Table 1 summarizes the nomenclature introduced in the present paper.

Table 1. Nomenclature

Term	Definition
Interaction	Exchange of energy, information, material or an association of physical space and alignment
Component	Simple physical entity which has Interaction with other simple physical entities
Component-DSM	Matrix of Interactions between pairs of Components
Element	(same as) Component
Cluster (<i>noun</i>)	Collection of Elements
cluster (<i>verb</i>)	generate a set of Clusters by means of an algorithm
IGTA	Idicula-Gutierrez-Thebeau Algorithm for clustering Component-DSM
Thebeau's algorithm	(<i>same as</i>) IGTA
ClusterSize	Number of Elements in Cluster
ClusterBid	Degree of fit between a selected Element and each of the existing Clusters; calculation includes a punishment for ClusterSize
Multiclusterallocation	Feature of IGTA where an element may be assigned to more than one cluster; occurs when more than one Cluster returns the highest ClusterBid
SMA	Suppressing Multiclusterallocation, allowing an Element to be assigned to one and only one Cluster
ITC	Improved Termination Criterion, selecting candidate Elements from a list, and subsequently deleting the Element from that list
IGTA-plus	Modification of IGTA that includes two algorithmic changes, SMA and ITC
Intra-cluster interaction	Interactions between Elements that belong to the same Cluster
Extra-cluster interaction	Interactions between Elements that belong to different Clusters
<i>TotalCost</i>	Sum of all Intra and Extra-cluster interactions, with an additional punishment for the latter
<i>DSM(i, j)</i>	Interaction between elements i and j
<i>ClusterSize_y</i>	Number of elements in cluster y
<i>DSMSize</i>	Number of elements in DSM
<i>powcc</i>	Exponent used to penalize the size of clusters in the formula for <i>TotalCost</i>

INTRODUCTION

Design Structure Matrix (DSM) [1][2][3] has been used for the purpose of generating product family architecture. Product families are based on the existence of modules: functional blocks with standardized interfaces that allow products with varying performance levels, features, or styling to be configured and manufactured efficiently [4]. Many researchers have identified product family architecture as a useful response to the need for increased variety, while con-

trolling complexity [5][6][7][8]. A particular type of DSM, Component-DSM, is concerned with interactions between individual Components [1]. The transformation of Component-DSM into proposed functional blocks of components is called Clustering. For small problems with perhaps up to 50 components, a Component-DSM may be sorted manually. For larger problems, this is not feasible, and at some point computer algorithms are absolutely necessary. Sharman [9] used a DSM to study the structure of a gas turbine. The turbine was divided into 31 “heterogeneous elemental sub-systems”, the relations between which were mapped in a Component-DSM of non-binary values. Automated Clustering was performed using the Thebeau Algorithm [10] (e.g., IGTA). Sharman performed twelve Clustering runs with four sets of parameter values and identified several problems with the output, one of the most important being the disappointing randomness of the output and the seeming need to run the algorithm many times.

In recent years, Genetic Algorithms (GA) have been used for the purpose of clustering Design Structure Matrices. Yu et al [11] used GA in conjunction with a Minimum Description Length metric. Their DSM has about 80 Elements. The runtime of their algorithm (compiled, written in C++) is stated as “about a day”.

Using GA and an objective function similar to the one used in IGTA, Whitfield et al [12] studied clustering arrangements for a Climate control system with 16 Elements. Since this is a relatively small matrix, runtime may not have been an issue, and no information is provided about that.

The algorithm proposed in the present paper grew out of research into a related but different topic: that of unifying DSM-based clustering with Hierarchical Clustering, often used in Modular Function Deployment, MFD [13][14] . It quickly became clear that to obtain reliably useful results, IGTA needs to complete thousands of runs. The application of IGTA to an example case with 57 Elements and performing 10 000 complete runs took about seven hours, which certainly does not invite “tinkering”. The need to improve IGTA so as to execute more quickly was identified as a priority. During this work, two improvements were identified, which improved speed almost by a factor of eight, reducing “night runs” to “lunch runs”.

There are a few of reasons IGTA was selected as the basis for this work. First, IGTA is freely available. It is written in Matlab and structured into program modules with clearly documented interactions, making it easy to modify. Second, it has been around for a decade and is known to be stable and bug-free. Third, IGTA has been used by others [9][15].

The present paper attempts to answer the following question: can IGTA be improved substantially so as to execute faster and improve quality of output?

IDICULA-GUTIERREZ-THEBEAU ALGORITHM (IGTA)

History of IGTA

IGTA is based on work by three researchers, John Idicula, Carlos Iñaki Gutierrez Fernandez, and Ronnie Thebeau. Idicula invented the original algorithm [16], which addressed the problem of organizing tasks in product development projects. The algorithm introduced the key concepts of ClusterBid and TotalCost, as well as the stochastic hill-climbing approach to avoid suboptimal solutions. Three years later, Carlos Iñaki Gutierrez Fernandez published C-code [17] which integrated the algorithm in an Excel environment, as well as additional user-control parameters. Finally, another three years later, Ronnie Thebeau translated the C-code into Matlab, and made experimental determinations as to the best value for several control parameters in the algorithm [18]. The code is freely downloadable on internet [10].

How IGTA works

For a complete explanation of IGTA, refer to [18]. We will focus on data representation, the overall structure, and the formula for calculating TotalCost. Figure 1 shows IGTA as a flowchart. IGTA moves Components from one Cluster to another, and tries to see if the overall change represents an improvement. The algorithm randomly picks a Component from an existing Cluster, and tries to determine if there is another Cluster where the Component represents a better fit. The fit is the ClusterBid. A ClusterBid is calculated for each Cluster. The highest ClusterBid “wins”, and the Component is moved to the corresponding Cluster. To determine whether the new configuration is better than anything found before, the algorithm calculates a quantity called TotalCost. The usage of two criteria, ClusterBid and TotalCost, reflects Idicula’s original optimization strategy [16][19], and a deeper discussion is outside the scope of the present paper.

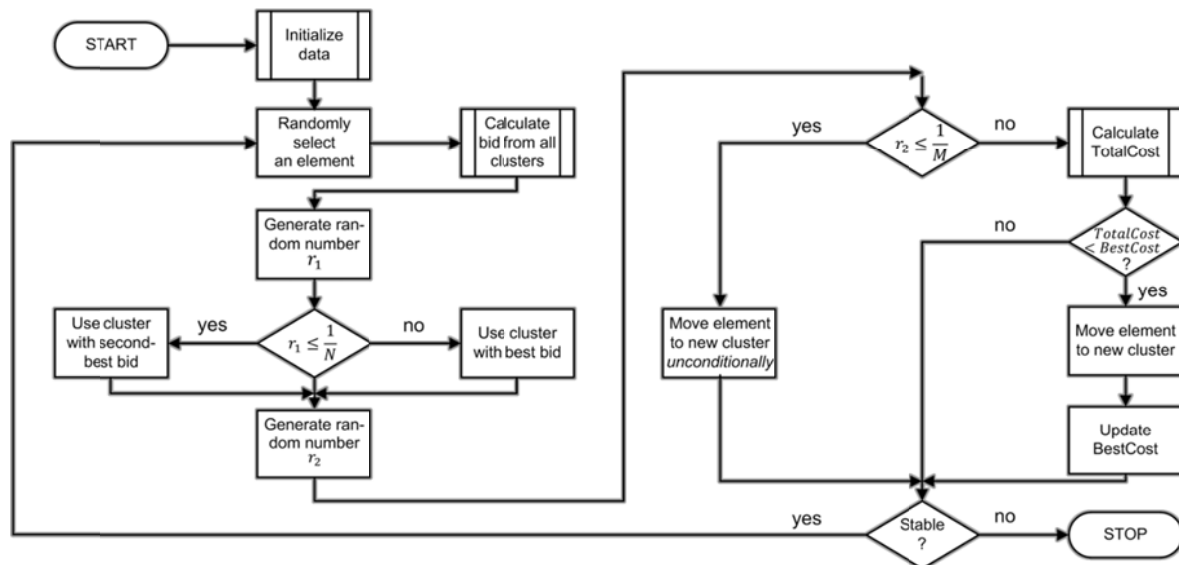


Figure 1. Flowchart of IGTA

The algorithm uses stochastic hill-climbing. It uses two random numbers to control the stochastic behavior. By allowing random changes to occur – according to pre-determined rules – the algorithm reduces the risk of getting stuck in local suboptimal solutions. As we shall see, that does tend to happen anyway.

IGTA allows a single Element to belong to more than one Cluster. IGTA uses a matrix called ClusterMatrix to keep track of the association between Elements and Clusters. Each row corresponds to a Cluster, and each column to an Element. If Element j belongs to Cluster i , then ClusterMatrix $_{ij}$ is set to one, otherwise zero. The number of Clusters is designated $n_{clusters}$. A one-dimensional matrix called ClusterSize counts the number of Elements of each Cluster. Table 2 shows the formula for calculating TotalCost [18]. The variables used are summarized in Table 1.

Table 2. IGTA equations for calculating TotalCost	
<i>No</i>	<i>Equations (IGTA)</i>
1	$IntraClusterCost = (DSM(j, k) + DSM(k, j)) \cdot (ClusterSize_y)^{powcc}$
2	$ExtraClusterCost = (DSM(j, k) + DSM(k, j)) \cdot DSMSize^{powcc}$
3	$TotalCost = IntraClusterCost + ExtraClusterCost$

Current challenges and opportunities with IGTA

IGTA allows an element to belong to more than one cluster. We refer to this as multicluster allocation. This is sometimes useful. In some cases we may prefer the algorithm to generate an output where each element belongs to one and only one cluster. It turns out that if we are willing to sacrifice multicluster allocation, it is possible to improve greatly the part of IGTA responsible for calculating TotalCost.

IGTA randomly picks elements and tries to see if they fit better in another cluster. When we are close to a very good solution, most elements are in their final cluster, and most such random attempts to move an element result in no move being made. IGTA keeps no record of the elements it has already tried, which means that it repeatedly evaluates moves that are known to be meaningless. IGTA has a cutoff point where it essentially says, “Nothing has happened for quite a while, I will assume I am done”. Depending on the settings of two parameters, it may take hundreds of iterations to come to that decision, which may be equivalent to half or two-thirds of the total number of iterations. The remedy we will explore involves simply keeping track of the elements that have already been tried, and not pick those again. This means the algorithm will never waste more iterations than there are elements in the input data.

We will refer to the first improvement as “suppressing multicluster allocation” (SMA) and the second as “improved termination criterion” (ITC). These two algorithmic improvements are independent of one another, meaning we can implement only ITC, if we want to allow multicluster allocation, for example.

PROPOSED IMPROVED ALGORITHM: IGTA-PLUS

Suppressing multicluster allocation (SMA)

The implementation of SMA consists of two parts: dealing with the multicluster condition and calculating TotalCost more efficiently.

Two or more ClusterBids could take on the same value. The multicluster condition occurs if that value happens to be the highest value, in which case IGTA allocates the selected element to all the Clusters with that ClusterBid value. This is the default behavior of IGTA, and we refer to it as multicluster allocation. When the SMA feature is enabled, IGTA-plus deals with the multicluster condition by randomly assigning the selected element to one of the Clusters with the highest ClusterBid value. When we have dealt with the multicluster condition, TotalCost may be calculated using the equations shown in Table 3.

Table 3. IGTA-plus equations for calculating TotalCost

<i>No</i>	<i>Equations (IGTA-plus)</i>
1	$\mathbf{IO} = \mathbf{ClusterMatrix} \times \mathbf{DSM} \times \mathbf{ClusterMatrix}^T$
2	$IO_{intra} = \sum_{k=1}^{n_{clusters}} \mathbf{IO}_{kk} \cdot ClusterSize_k^{powcc}$
3	$IO_{extra} = \left(\sum_{i=1}^{n_{clusters}} \sum_{j=1}^{n_{clusters}} \mathbf{IO}_{ij} - \sum_{k=1}^{n_{clusters}} \mathbf{IO}_{kk} \right) \cdot DSMSize^{powcc}$
4	$TotalCost = IO_{intra} + IO_{extra}$

The simplification comes from the insight that Equation number 1 in Table 3 collapses the Component-DSM into a Cluster-DSM, where elements on the diagonal are the sums of interactions within Clusters, any off-diagonal values are interactions between Clusters. The equations in Table 3 may be implemented very efficiently in Matlab with just a few lines of program code.

Improved termination criterion (ITC)

ITC involves keeping a list of elements that have not yet been tried. As elements are tried, they are removed from the list. When the list is empty, the algorithm terminates. If a move is made, the list needs to be reset. Figure 2 shows the principle of ITC.

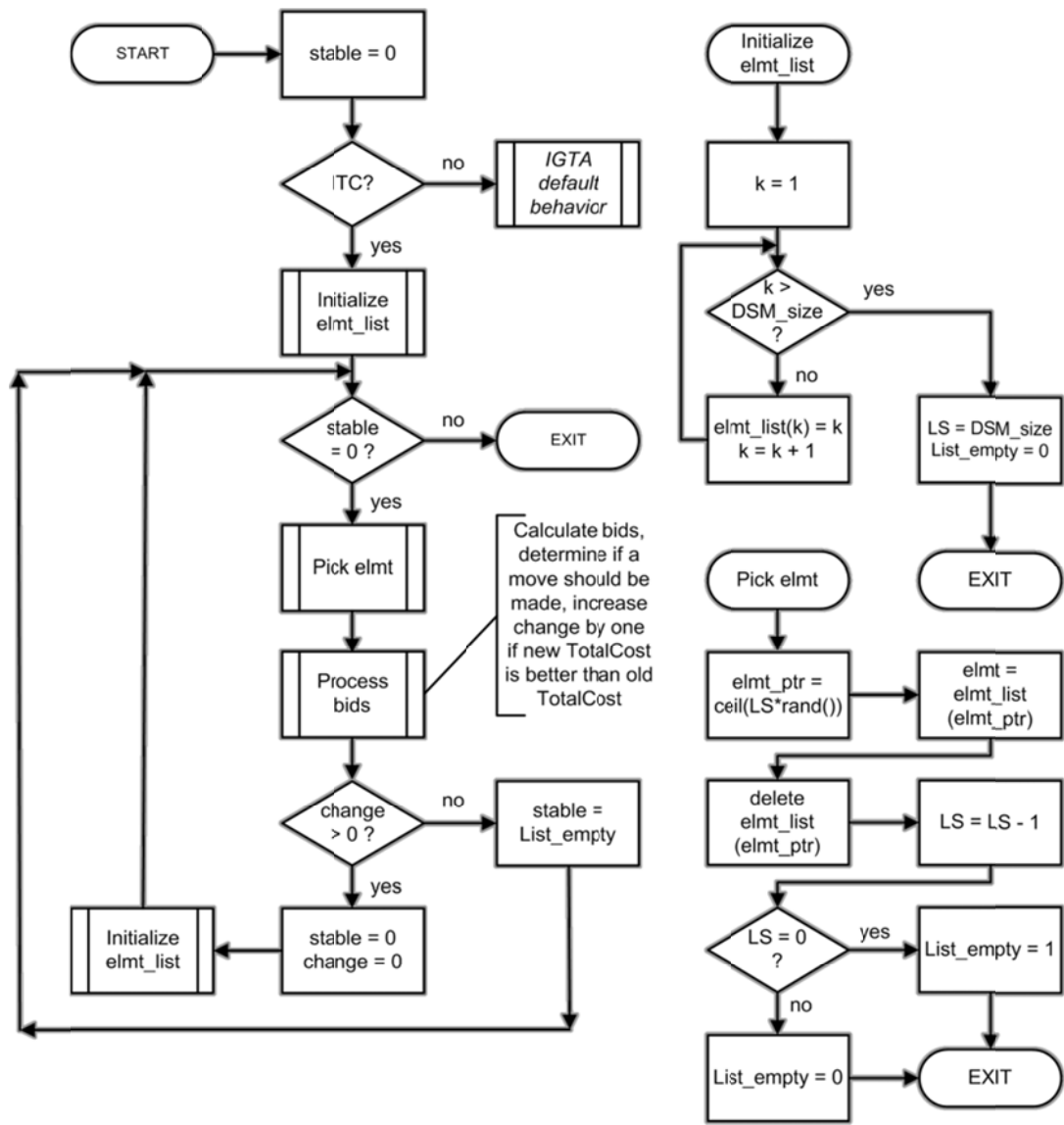


Figure 2. Flowchart of ITC

EVALUATING IGTA-PLUS

IGTA-plus does not purport to make better modules than IGTA, it just claims to do it faster. Thus, in evaluating IGTA-plus, we are interested in two questions.

1. Does IGTA-plus, in fact, generate solutions that are at least as good IGTA?
2. How much faster is it? And specifically, how much of the improvement is coming from SMA and how much from ITC?

Test case

To answer these two questions, IGTA and IGTA-plus will be applied to a test case, a Dustbuster from Black & Decker shown in Figure 3 and partially disassembled in Figure 4.



Figure 3. Dustbuster CHV1210 from Black & Decker

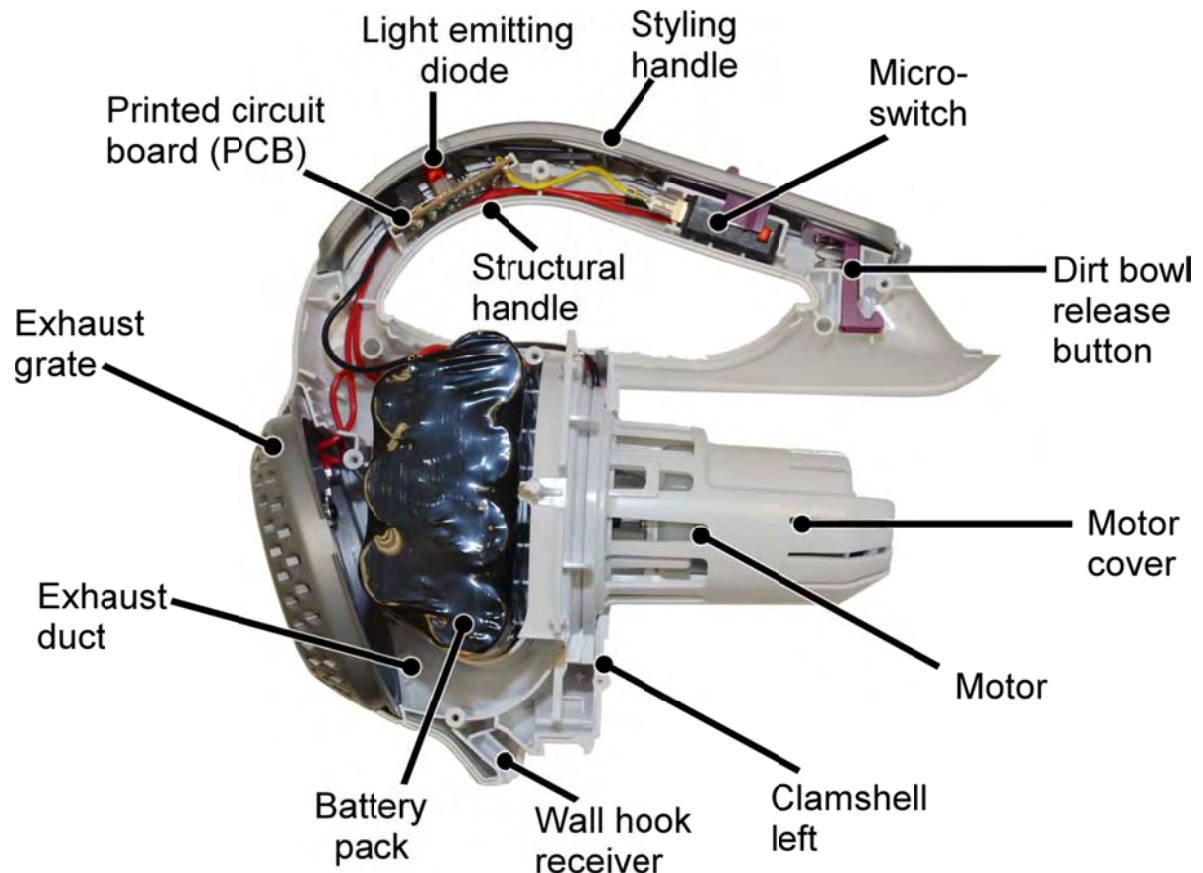


Figure 4. Dustbuster taken apart

IGTA-plus and IGTA each performed 10 000 complete runs of clustering. All interaction types were replaced with the numerical value of 1. The following section reports best solution found, distribution of solutions, a graph of TotalCost for the best solution, and finally the execution time.

RESULTS

Solutions found

Figure 5 shows the best solutions found.

IGTA-plus		IGTA		IGTA-plus	
14	Battery bay left	1	16	16	Low voltage AC connector male
14	Battery bay right	1	16	10	Leads
1	Clam shell left	1	2	10	Low voltage AC connector female
1	Clam shell right	1	2	4	Light Emitting Diode
1	Dirt bowl receiver	1	2	4	Rectifier diode
1	Escutcheon	1	2	4	Resistor
1	Exhaust duct	1	2	4	Transistor
1	Exhaust grate	1	2	4	Printed Circuit Board
1	Impeller housing	1	13	6	Motor terminals
1	Impeller	1	13	6	Vibration damper
1	Motor bracket	9	9	6	Electric motor
1	Wall hook receiver	10	10	6	Motor shaft
9	Wall hook	14	5	7	Motor cover
9	Wall mount	14	5	7	Filter media
9	Grooves	15	5	7	Filter media holder
2	Nozzle release spring	15	5	8	Nozzle release latch receiver
2	Nozzle release latch	3	5	8	Dirt bowl release latch
2	Nozzle	3	5	8	Dirt bowl
2	Vortex generator	3	12	5	Battery cell blank
2	Nozzle air duct	3	12	5	Shrink wrap
2	Dust flap	3	8	5	Battery pack terminals
2	Nozzle release holder	3	8	5	Battery cell rechargeable
3	Dirt bowl release spring	3	6	11	Transformer
3	Dirt bowl release button	3	6	11	Power plugs
3	Anti-theft device	4	6	11	Encapsulation
3	Structural handle	4	7	12	Button cam receiver
3	Styling handle	4	7	12	Button cam device
3	Power button	4	11	13	Microswitch bracket
		4	11	13	Microswitch

Figure 6. Clusters from Figure 5 shown as a list

TotalCost

Each algorithm was run 10 000 times, and a running tab was kept of the TotalCost of each solution. The curves in Figure 7 shows the distribution of the solutions. The vertical axis has been scaled to give an area of exactly one under each curve. Note the peak of the blue curve, IGTA-plus, is narrower than that of IGTA, and shifted to the left, meaning both average and standard deviation is lower.

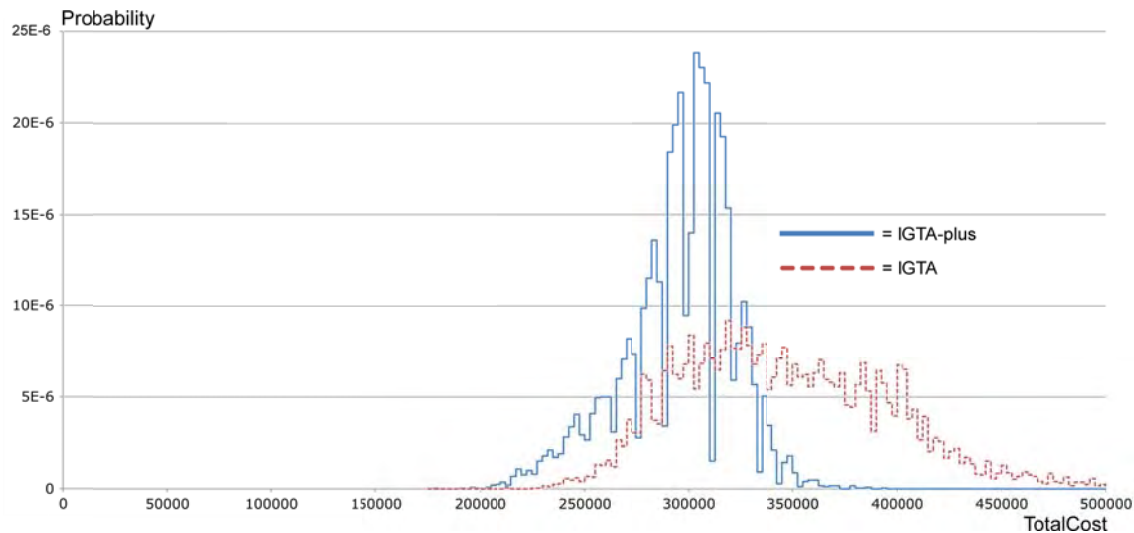


Figure 7. Distribution of solutions found

Cost history

The graph in Figure 8 shows the TotalCost as it changes for every iteration. This is referred to as cost history. One complete run consists of several hundred iterations, and the run shown here is the best solution found.

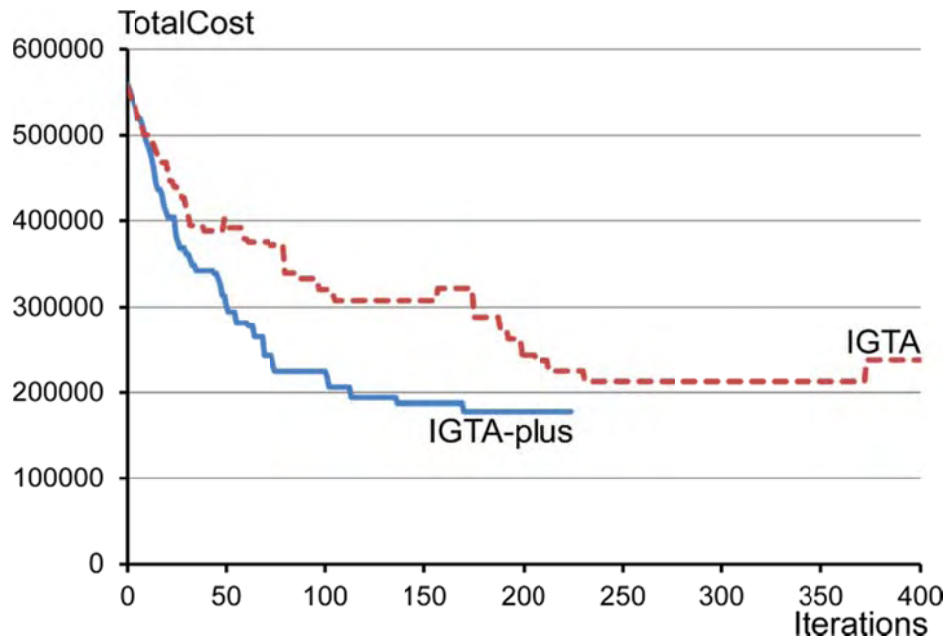


Figure 8. Cost history for both algorithms - best solution shown

IGTA terminates after 783 iterations. Only the first 400 are shown in Figure 8. IGTA-plus terminates after 225 iterations. Note wherever the cost history curve jumps up, that is a consequence of the stochastic behavior built into the algorithm. The best solution found by IGTA has a TotalCost of 212,744. The best solution found by IGTA-plus has a TotalCost of 178,127 which is 16% lower. In a test using 100 000 runs, the best solution still had TotalCost of 178,127 so it seems likely there really is not a better solution to be found.

Execution speed

The bar chart in Figure 9 shows the execution time per complete run.

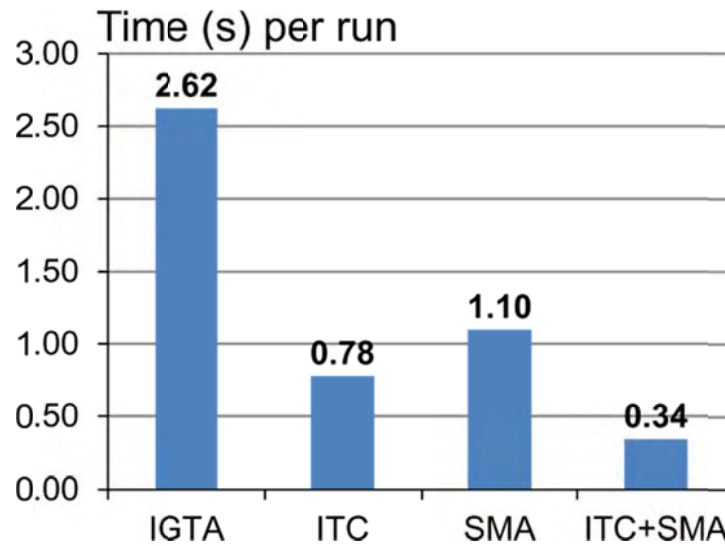


Figure 9. Execution time (seconds)

Expressed as percentages we get the following improvements

- ITC reduces time per iteration by 70% (e.g., about 3.3 times faster)
- SMA reduces time per iteration by 58% (e.g., about 2.4 times faster)
- ITC and SMA together reduce time per iteration by 87% (e.g., about 7.7 times faster), which is equivalent to the product of the contributions from ITC and SMA separately, indicating they are independent.

OBSERVATIONS

In the Dustbuster trial used to test IGTA-plus, we observe that:

- IGTA-plus finds a better solution. When both algorithms are allowed to run many times, IGTA-plus finds a solution with a lower TotalCost than IGTA.
- IGTA-plus is more consistent. Both average and standard deviation of TotalCost was lower, which means IGTA-plus is more consistent than IGTA.
- IGTA-plus is much faster. Using the same settings, IGTA-plus runs seven to eight times faster than IGTA.
- The improvements are independent. The two algorithmic improvements, ITC and SMA, are independent of each other.

CONCLUSION AND FUTURE RESEARCH

IGTA-plus represents a significant improvement in speed and quality of solution obtained. When both improvements are enabled, e.g., ITC and SMA operate in conjunction, the best speed increase is obtained. SMA may be thought of as a more efficient formula for calculating TotalCost. It does not alter the way the algorithm works. ITC is different, however. It changes the way the algorithm selects candidate elements, and the trial runs indicate this seems to have a very positive effect on the quality of the solution thus obtained. An algorithmic improvement originally devised to address computational efficiency ended up yielding better results (in addition to speed). It is not immediately obvious why that is the

case, but one way to think about it is ITC just does a better job of exhausting all possibilities before it “gives up”.

In DSM research using IGTA or IGTA-plus, the algorithm should be allowed to run many times, probably on the order of several thousand times, to ensure that a solution very close to the best possible is obtained.

It is probably possible to improve further on IGTA-plus. The basic algorithm still displays a tendency to get stuck in local optima. One way to alleviate this might be to introduce recurring disruptive changes. Another topic of research would be to add heuristics that recognize common structures such as a bus architecture.

NOTE ON MATLAB CODE

The Matlab code developed during this research is available for researchers interested in testing IGTA-plus or making further improvements. Please contact the corresponding author via email.

ACKNOWLEDGMENTS

The authors would like to thank Mr. John Idicula, original inventor of IGTA, for detailed information on the origins of the algorithm and the research context in which it came about [19].

REFERENCES

- [1] Browning, T. R., 2001, Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions, *IEEE Transactions on Engineering Management* , 48 (3), 292-306.
- [2] Eppinger, S., Whitney, D., Smith, R. & Gebala, D., 1994, A model-based method for organizing tasks in product development, *Research in Engineering Design*, 6, 1-13.
- [3] Steward, D. T., 1981, The Design Structure System: A Method for Managing the Design of Complex Systems, *IEEE Transactions on Engineering Management* , 28 (3), 71-74.
- [4] Simpson, T. W., 2004, Product Platform Design and Customization: Status and Promise, *Artificial Intelligence in Engineering Design, Analysis and Manufacture*, Special Issue: Platform Product Development for Mass Customization, 10 (1).
- [5] Dahmus, J. B., Gonzalez-Zugasti, J. P., 2001, Modular product architecture, *Design Studies*, 22(5), 409-424.
- [6] Martin, M., Ishii, K., 2000, Design for variety: Developing standardized and Modularized product platform architecture, *Research in Engineering Design*, 13 (4), 213-235.
- [7] Meyer, M. H., Lehnerd, A., 2000, *The power of product platforms*, New York, NY: The Free Press.
- [8] Simpson, T. W., Marion, T., de Weck, O., Holtta-Otto, K. & Shooter, S. B., 2006, Platform-Based Design and Development: Current Trends and Needs in Industry, *Proceedings of the ASME 2006 International design engineering technical conferences IDETC2006*, Philadelphia, PA.

- [9] Sharman, D. M., Yassine, A. A., 2007, Architectural Valuation using the Design Structure Matrix and Real Options Theory, *Concurrent Engineering* , 15 (2), 157-173.
- [10] Thebeau, R. E., 2001B, Matlab program code, available at <http://www.dsmweb.org/>.
- [11] Yu, T.-L., Yassine, A. A., Goldberg, D. E., 2007, An information theoretic method for developing modular architectures using genetic algorithms, *Res. Eng. Design* 18:91-109, DOI 10.1007/s00163-007-0030-1.
- [12] Whitfield, R. I., Smith, J. S., Duffy, A. H. B., 2002, Identifying Component Modules, *Proceedings of the 7th international conference on artificial intelligence in design AID'02*, pp. 571-592.
- [13] Ericsson, A., Erixon, G., 1999, *Controlling Design Variants: Modular Product Platforms*, ASME Press, The American Society of Mechanical Engineers, New York, NY. ISBN 0-87263-514-7.
- [14] Erixon, G., 1998, *Modular Function Deployment - A method for Product Modularisation*, PhD thesis, The Royal Institute of Technology, Stockholm.
- [15] Hölttä-Otto, K., 2005, *Modular Product Platform Design*, Doctoral Dissertation, Helsinki University of Technology, Department of Mechanical Engineering, Machine Design, Helsinki, Finland. ISBN 951-22-7766-2.
- [16] Idicula, J., 1995, *Planning for Concurrent Engineering*, Gintic Institute Research Report, Singapore.
- [17] Gutierrez Fernandez, C. I., 1998, *Integration Analysis of Product Architecture to Support Effective Team Co-Location*, Master's Thesis at Massachusetts Institute of Technology, Department of Mechanical Engineering.
- [18] Thebeau, R. E., 2001A, *Knowledge Management of System Interfaces and Interactions for Product Development Process*, Master's Thesis at Massachusetts Institute of Technology, System Design & Management Program.
- [19] Idicula, J., 2011, *A note on some computational problems in management of engineering projects*, Manuscript, Private Communication, February 2011.